Cooperation in Multi-hop Cellular Networks *

Naouel Ben Salem[†], Levente Buttyán[‡], Jean-Pierre Hubaux[†], Markus Jakobsson[§]

Technical Report Nr. IC/2003/55

Abstract

In multi-hop cellular networks, 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) cellular networks, this new generation can lead to a better usage of the available spectrum and to a reduction of infrastructure costs. However, these benefits would vanish if the mobile nodes do 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 analyse the robustness of our protocols against rational and malicious attacks. We show that our protocols thwart rational attacks and detect malicious attacks. We also show that our solution makes collaboration rational for selfish nodes.

Keywords: Cooperation, Multi-hop Cellular Networks, Hybrid Cellular Networks, Ad Hoc Networks, Packet Forwarding, Security, Pricing, Charging, Billing

1 Introduction

The geographic area covered by a conventional cellular network is populated with base stations 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 cellular network. For the operator, the usual solution to this problem is to increase the coverage of the network by adding antennas, and for the user to move until he reaches a covered region. An alternative solution would be to allow multi-hop communications in the cellular network, which makes it possible for the isolated node to ask other nodes to relay its traffic to or from a base station.

The resulting network [1, 34, 12, 2, 32], frequently called *multi-hop cellular network*, presents several benefits [24, 13, 25, 22]. First of all, the coverage of the network is increased while the number of fixed antennas is kept relatively small. 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, multi-hop cellular networks represent a new and promising paradigm. However, the proper operation of this new family of networks requires the mobile nodes to collaborate with

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

[†]Laboratory of Computer Communications and Applications (LCA), Swiss Federal Institute of Technology – Lausanne (EPFL), Switzerland ({naouel.bensalem, jean-pierre.hubaux}@epfl.ch)

[‡]Budapest University of Technology and Economics, Department of Telecommunications, Hungary (buttyan@hit.bme.hu)

[§]RSA Laboratories, 174 Middlesex Turnpike, Bedford, MA 01730, USA (mjakobsson@rsasecurity.com)

each other. This collaboration is not guaranteed in a civilian network, where each node represents a single authority. Indeed, forwarding packets is energy-consuming and a selfish user can tamper with his mobile device to remove the relaying functionalities 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 multi-hop aspect of the communications.

In this paper, we propose a technique to foster cooperation for the packet forwarding service in multi-hop cellular networks. This solution is based on a charging and rewarding system.

This paper extends and completes our previous treatment of the same problem [3]. This work is part of the Terminodes Project [19, 4, 15]. 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 analyse the robustness of our solution against rational and malicious attacks and we show that the charging and rewarding scheme indeed encourages cooperation in multi-hop cellular 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.

2 System model

The essence of our proposal is a set of protocols that encourages the nodes to cooperate in multi-hop cellular networks.

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. Communication between the mobile nodes and the base stations is based on wireless technology. We assume that all communication is packet-based and that the links are symmetric (i.e., the nodes and the base stations have the same power range). We also assume 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.

We call a *cell* [24] the geographical area under the control of a given base station. The power range of the base station is smaller than the radius of the cell, meaning that some nodes should rely on *multi-hop relaying* to communicate with the base station. We consider an ad hoc model in which mobile 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 (reactive or proactive) routing protocol. We also assume that all links are bidirectional.

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 node B (the correspondent), it first establishes an *end-to-end session* with B (as we will see in detail, in Subsection 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[1]. 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 [24] 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 only have to maintain a single route (to the closest[2] base station) instead of one route per potential correspondent. Of course, the base station has to maintain a route to every node in its cell.

In order to motivate the intermediate nodes to forward the traffic, we propose to charge the initiator A for the traffic in both directions and 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 Subsection 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 $\mathcal{A}$ is *rational* if it misbehaves only when this is beneficial in terms of remuneration, service provision or ressource saving. Otherwise, $\mathcal{A}$ is *malicious* (i.e., it misbehaves even if it loses money, energy, ... by doing so). The users are selfish and thus, each node in the network is potentially an attacker. We assume that several attackers can collude and share information 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 records and analyzes the exchanged data without altering them). 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 the different phases of the proposed protocols and we identify the following active attacks:

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

Secure routing protocol: Securing the routing protocol is beyond the scope of this paper; several solutions to secure routing in ad hoc networks have been proposed [17, 18, 29, 30, 6, 35, 16] and can be used as an underlying routing mechanism in our protocol. Therefore, we will not consider attacks such as abnormally long routes, routing loops, black or gray holes, wormhole attack etc.

Encouraging cooperation: We will show in Subsection 4.1 that the packet dropping attack is not rational. This means that cooperation is the best choice for a rational selfish node.

[1] Clearly, the communication ends are not necessarily both wireless. However, in this paper, we assume that to be the case as this corresponds to the most complete scenario.

[2] The distance from a given node to the base station can be expressed in termes of number of hops, time, cost, etc....

3 Proposed solution

In this section, we will provide a detailed description of our solution. In particular, we will describe the protocols of the different phases sketched in Subsection 2.2.

3.1 Building blocks and notation

Our protocols use two cryptographic building blocks: a MAC (Message Authentication Code[3]) function and a stream cipher [27]. However, our use of these cryptographic primitives is unconventional:

- During the session setup phase (see Subsection 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 Subsection 3.2. Our solution achieves a similar effect to that of the classical MAC appending technique while keeping 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 Subsection 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 Subsection 4.3).

Notation: We denote the concatenation operator by $\mid$ 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 (i) 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 secure routing protocol; (ii) to authenticate all nodes belonging to these routes; and (iii) to inform these nodes about the traffic that will follow. A node can decide not to join the session, in which case the session setup fails and a new session is established using an alternative secure route. Successful completion of the session setup phase is a confirmation that both the initiator and correspondent routes are operational and that all the intermediate nodes on both routes accept to forward the traffic.

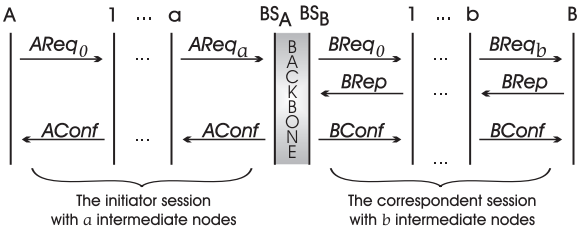

Figure 1: The session setup phase

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 *secure* initiator route $ARoute$, and some

[3]Throughout this paper, MAC will stand for Message Authentication Code and not for Medium Access Control.

information $TrafficInfo$ about the traffic to be sent[4]. 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 re-establish a broken session. This field is set to zero in case of a new session establishment. Finally, $AReq_0$ contains a MAC computed by A using its secret key K_A:

$$\begin{aligned} AReq_0 \quad = \quad & [\, AReqID \mid oldASID \mid ARoute \mid TrafficInfo \mid \\ & MAC_{K_A}(AReqID \mid oldASID \mid ARoute \mid TrafficInfo)\,] \end{aligned}$$

Each forwarding node i ($1 \leq i \leq a$) 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:

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

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 re-establish 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 $BReq_0$ towards B:

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

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 re-establish a broken session and $BRoute$ is the secure correspondent route.

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

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

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:

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

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 towards 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

[4]The initiator A may have no 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.

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

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

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

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

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., $K_i' = h_{K_i}(SID)$) 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 will discard all messages or packets of 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 Subsection 3.4).

Note that in case of initiator (respectively, correspondent) session re-establishment, it is not necessary to also re-establish the correspondent (respectively, the initiator) session if the latter is still valid. The broken session is re-established 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.

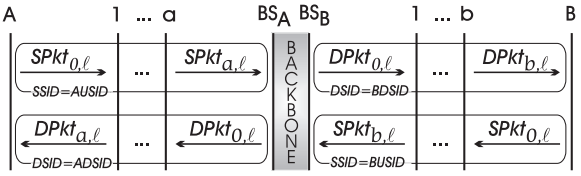

Figure 2: The packet sending phase

The ℓ-th packet $SPkt_{0,\ell}$ sent by S contains the session ID $SSID$ (which is $AUSID$ if $S = A$ and $BUSID$ if $S = B$), the sequence number ℓ, and the payload $Payload_\ell$. It also contains the receipt $SRcpt_{0,\ell}$ that can be used by the intermediate nodes, if needed, to provide the operator with a proof that they correctly received the packet (details about the computation and the use of the receipts are given in Subsections 3.4.4 and 3.4.1). 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 $PAD_{S,\ell}$:

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

The pads $PAD_{i,\ell}$ are generated by node i ($i = S$ for the source) as follows (see Figure 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 ℓ-th 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.

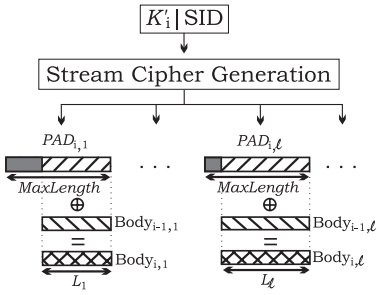

Figure 3: Encryption of the packets

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 $SRcpt_{i,\ell}$ 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. Then, it checks the MAC and checks that the packet is not a duplicate. If one of these verifications is not successful, then it drops the packet. Otherwise, it forwards it[5] to the base station of the destination BS_D. BS_D changes the up-stream session ID to the corresponding 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:

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

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. Upon reception of $DPkt_{j-1,\ell}$, each node j computes and stores the receipt $DRcpt_{j,\ell}$ for the packet (as explained in Subsection 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:

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

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

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 Subsection 3.4.2).

3.4 Payment Redemption

3.4.1 Charging

As we have already mentioned in Subsection 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 initiator A is charged a given amount $n(L_\ell)$ (depending on the packet size) and the up-stream forwarding nodes are credited $\alpha(L_\ell)$. The down-stream forwarding nodes are credited when Pkt_ℓ is acknowledged by D (see Subsection 3.4.2) because the operator may have no other reliable information about the delivery of the packet. The only motivation for D for not sending 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 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 cashes the charge ε if no acknowledgement arrives for the packet 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 Subsection 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[6]. A is charged $n'(L_{min}) = (k-1).\alpha(L_{min}) + \beta(L_{min})$. 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 $\beta \gg c'$). Note that α should be substantially larger than β to prevent nodes from systematically dropping packets ($\alpha \gg \beta$).

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 $n(L_\ell)$.

Note that the charges and rewards depend only 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 on average.

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[7]" and the packets received during each period are acknowledged in a single batch. The acknowledgment $DAck_t$ of the t-th time period of the session is formatted as the payload of a regular packet[8] and sent by D via the down-stream route to BS_D, some time during the $t+1$-th time period (e.g., at a randomly chosen

[6] The packet did not reach the base station, so the operator has no idea about its real length. We choose to give a reward equivalent to the one a node gets when it forwards the shortest possible packet in order to avoid that a forwarding node drops short packets it is asked to forward in order to get higher reward.

[7] We suppose that the nodes are at least loosely synchronized with their base station.

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

moment). The body of the acknowledgment packet is:

$$\begin{aligned} DAck_t &= [\, DBatch_t \mid DFirstPkt_t \mid DLastPkt_t \mid DLostPkts_t \mid \\ &\qquad MAC_{K'_D}(DBatch_t \mid DFirstPkt_t \mid DLastPkt_t \mid DLostPkts_t) \,] \end{aligned}$$

where $DFirstPkt_t$ and $DLastPkt_t$ are the sequence numbers of, respectively, the first and the last received packets during the t-th time period, $DLostPkts_t$ is the list of sequence numbers of the missing packets between $DFirstPkt_t$ and $DLastPkt_t$ and

$$DBatch_t = \bigoplus_{DFirstPkt_t \leq \ell \leq DLastPkt_t;\ \ell \notin DLostPkts_t} MAC_{K'_D}(DSID \mid \ell \mid Payload_\ell)$$

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 identified as being an acknowledgement. Then, BS_D verifies the MAC and checks $DBatch_t$ by XORing all the MACs of the packets from $DFirstPkt_t$ to $DLastPkt_t$, excluding those in $DLostPkts_t$ 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 $t+1$-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-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:

$$\begin{aligned} UAck_t &= [\, UFirstPkt_t \mid ULastPkt_t \mid ULostPkts_t \mid \\ &\qquad MAC_{K'_S}(UFirstPkt_t \mid ULastPkt_t \mid UDLostPkts_t) \,] \end{aligned}$$

When S receives $UAck_t$, it identifies it as being an acknowledgement and checks its validity by verifying its MAC. S can choose to re-establish 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 [36]. It does not represent a proof that the node forwarded the packet but rather that it received it correctly. As we will see in Subsection 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 Subsection 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 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 part of the packet $DPkt_{j,\ell}$ (i.e., if the MAC is encoded on 16 bytes, then $M_{j,\ell}$ represents the last 16 bytes 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) and because we use the *last* bytes of the pads $PAD_{j,\ell}$ to encrypt and decrypt the packets.

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:

$$Batch_{SID,i} = \bigoplus_{\ell \leq LastPkt \ ; \ \ell \notin LostPkts} Rcpt_{i,\ell}$$

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

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 involved in the communication about the rewards they received (typically when the node is within the power range of a base station). 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 Subsection 3.4.1. A node can ask for remuneration (by sending the receipt) even if it did not provide the service. Note that a single receipt is sent to ask remuneration for several packets.

The format of the packet receipt sent to the operator is the following:

$$\begin{aligned} Rcpt_{SID,i} = \ & [\ SID \mid Batch_{SID,i} \mid LastPkt \mid LostPkts \mid \\ & MAC_{K'_i}(SID \mid Batch_{SID,i} \mid LastPkt \mid LostPkts)\ ] \end{aligned}$$

Upon reception of this message, the operator verifies the MAC and if the verification is positive, it remunerates the node according to the charging scheme (see Subsection 3.4.1).

4 Security Analysis

In this section, we study the robustness of our protocols against the attacks identified in Subsection 2.3. We do not consider attacks against the routing protocol or denial of service (DoS) attacks based, for example, on jamming.

4.1 Packet dropping

In this attack, an attacker $\mathcal{A}$ 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 is interesting, particularly for the packet sending phase, because it proves that our solution fosters cooperation.

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

- The request message: the sender of the request (which is A or BS_B) would not receive the confirmation or the reply message, respectively. It would then establish 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 not to participate in a given session.

- The reply message: BS_B would never receive the reply and the correspondent session setup would fail. BS_B would then use another route to establish the correspondent session.

- The confirmation message: some of the nodes involved in the communication would not be 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: (i) $\mathcal{A}$ is in the initiator route, therefore A would not receive the confirmation message and would consider that the session setup failed; It would then establish a new session using another route. (ii) $\mathcal{A}$ is in the correspondent route; The session would be active for all the nodes, except for those that are after $\mathcal{A}$ in the correspondent route (including B); These nodes would thus discard all the packets sent by A during the session. The destination (which is B in this case) would not be able to send the periodic acknowledgment to BS_B and the session would be re-established.

  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 Subsection 3.4.2, 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 Subsection 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 (i) a forwarding node cannot know if the receipt is valid or not before sending it to the operator, (ii) the cost of sending the receipt is negligible and (iii) $\alpha \gg \beta \gg c$, 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 Subsection 4.3, the filtering attack is malicious. The nodes involved in the communication being rational, they will not perform this attack on the packets they are asked to forward and thus the receipts produced by the intermediate nodes would 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 would be $\alpha - c$,
- if on the contrary Pkt_ℓ does not reach its target, then at least one node j ($j > i$) would send a receipt for it (according to Proposition 1) and the payoff of i would also be $\alpha - c$.

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 $\mathcal{A}$ that drops it. The rational nodes that are before $\mathcal{A}$ in the path will then send valid receipts for Pkt_ℓ (according to Proposition 1). The operator would identify the last node k in the path that sent a valid receipt, which is $\mathcal{A}$ or the rational node that is before it on the route (because $\mathcal{A}$ is also able to generate a valid receipt for the packet). The operator would then suspect 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 $\mathcal{A}$ performs this attack only 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 $\mathcal{A}$ 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 $\mathcal{A}$ has thus no way to distinguish a packet containing an acknowledgement from a data packet[9]. A brute force attack would be for $\mathcal{A}$, in order to specifically drop the t-th acknowledgement, to drop all the packets sent during the $t + 1$-th time period. The consequence of this attack would be the re-establishment of the session using another route.

4.2 Replay attack

We will consider that a replay attack performed by an attacker $\mathcal{A}$ is successful if the replayed message or packet is considered as valid by *all* the parties involved in the communication (including the operator)[10].

[9] It is possible for the attacker to identify the acknowledgement packet if it has a fixed and predefined length. If we want to remedy this, the generator of the acknowledgement should insert some padding.

[10] Fooling just one or several relay nodes does no more harm than a DoS attack based, for example, on jamming.

Note that $\mathcal{A}$ is not necessarily part of the network. In this paragraph, 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) would thus be detected by the first base station that receives it.

However, a detection at the nodes is also possible. Indeed, 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
- or the session to be established is already active or it is closed but still in memory[11]
- or i is supposed to be the initiator of the communication

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. Indeed, each forwarding node maintains the list of all packets it has received so far (for the computation of the receipt, see Subsection 3.4.4). The sequence number of the packet to forward would then correspond to the identifier of an already handled packet and the duplicate would be 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 Subsection 4.1, it is difficult to identify the packets containing the acknowledgements and thus to replay them specifically.

4.3 Filtering attack

An attacker $\mathcal{A}$ that performs a filtering attack modifies one or several fields of the packet it is asked to forward. In this Subsection, we will analyse the effect of this attack on our protocols. We also consider the *free-riding* attack where two colluders $\mathcal{A}_1$ and $\mathcal{A}_2$ 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: $\mathcal{A}$ can tamper with:

- The request or the reply messages: the verification of the "layered" MAC will fail and the base station (BS_A or BS_B) will discard the message. A new session will then be established using an alternative route.
- The confirmation message: the first node that will receive the tampered message will discard it because the verification of the MAC will fail. If $\mathcal{A}$ tampers with one (or more) MAC(s) in the message, the first node whose MAC was modified and that will receive the message will discard it. This attack has the same effect as dropping the confirmation message (see Subsection 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 $\mathcal{A}_1$ and $\mathcal{A}_2$ 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: $\mathcal{A}$ can tamper with the different fields of the packet Pkt_ℓ.

- Modifying SID, ℓ or $Body_{i,\ell}$ would be 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.

[11] The mobile nodes do not keep trace of all the messages and packets they received. Rather, they maintain a short-term history (i.e., on-going sessions and session that are not acknowledged yet).

- We hereafter define the *early duplicate* attack, a malicious attack where $\mathcal{A}$ 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 handled 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" ℓ-th packet, the forwarding nodes would consider it as a duplicate and thus would discard it.

  Our protocols, as presented so far, are vulnerable to the *early duplicate* 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[12].

  Let us first consider the solution, for the *early duplicate* attack, 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 ℓ-th packet it generates[13]. BS_A sends the hash value ADw_ℓ with the ℓ-th packet it sends toward A. The intermediate nodes can verify the validity of the hash values by checking[14] that $w_0 = h^\ell(w_\ell)$ ($w = AUw$ or ADw). The packets containing invalid hash values would be 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 session. This means that packets 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 would be detected at the first node that will receive the modified packet because the checking of the hash value will fail.

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

- The free-riding attack is not rational during the packet sending phase. Indeed, the data sent by $\mathcal{A}_1$ cannot be interpreted by $\mathcal{A}_2$ because it was encrypted at least by one intermediate node[15]. If this attack is performed anyway, it is detected as a "regular" filtering or packet dropping attack (depending whether $\mathcal{A}_2$ forwarded the tampered packet or not).

- Modifying only the receipt 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 should re-establish the session (if $S = B$) or ask the initiator

[12] A hash chain is a chain of N hash values where $w_{N-i} = h(w_{N-i+1})(0 < i \leq N)$ and h is a one-way hash function.

[13] 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.

[14] The verification of the hash value can be optimized if we use mechanisms such as [10] for example.

[15] 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.

to do it (if $S = A$). Such a radical solution is needed because, as explained in Subsection 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 will not verify correctly at the operator.

- The attacker $\mathcal{A}$ 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 would forward the modified packet. When the target (BS_S or D) receives it, it detects the attack and re-establishes the session.

Payment redemption phase: The same analysis as for the packet dropping attack during the payment redemption phase holds.

4.4 Emulation attack

This attack is equivalent to the cloning of a SIM card in a GSM network. It is detected in the same way.

4.5 Hybrid attacks

So far, we analyzed the effect of each of the four active attacks we identified in Subsection 2.3. However, more sophisticated attacks can combine two or more of the attacks described so far. The description and analysis of one of these attacks can be found in Appendix II (Supplemental material).

5 Overhead

In this section, we will 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.

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

Table 1: Size of the fields used in our protocol (for both up and down streams)

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 *Receipt* 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 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.

In order to estimate this overhead, we conducted a set of 100 simulations with a network of 100 nodes and one base station. The nodes are randomly laid out on a 500x500 m^2 single cell[16] and the base station is 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 [21] and we discard the first 1000 seconds of simulation time to remove the initial transient phase [9]. The speed is uniformly chosen between 3 and 23 m/s [33], which corresponds to an average speed of 10.56 m/s at t = 1000 s. The pause time is 0 s.

In our simulations we are interested in the average lifetime of a route (*AvrLT*) and the average number of forwarding nodes (*NbFwdrs*). After the initial transient phase of each simulation, we randomly choose a node that has a route to the local base station and we observe the lifetime of this route. The simulation ends when the route is broken (i.e., at least one link is broken). *AvrLT* represents the average value of all these lifetime values over the 100 simulations. Note that we consider the route on only one side of the communication (the initiator or the correspondent route) and not the end-to-end route. *NbFwdrs* is computed for the node we consider for the *AvrLT*. We do not expect this number to be large for multi-hop cellular networks.

The results are the following: the route remains stable for an average of *AvrLT* = 6.58 s ($\pm$1.66 s with 95% confidence interval), with an average number of forwarding nodes of *NbFwdrs* = 1.36. In order to estimate the amount of information that a node can send during this period of time, we 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 [11]. It is possible during 6.58 s to send 52.64 kbytes of data.

Using the numbers of Table 1, we estimate the overhead of the end-to-end session setup, with the protocol proposed here, to be around 243 bytes,which represents less than 0.5% of the payload. Moreover, as explained in Subsection 3.2, it is possible to re-establish only the broken initiator or 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 re-establishment (see Section 4). However, if the attacks are repeated often, the operator can collect information about the problematic sessions, identify the suspected nodes and statistically identify the attacker(s) (details about the detection are in Appendix I - supplemental material). Identifying and punishing the attackers can represent 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% of the packet size. This overhead is reduced if we use larger packets. An additional (and substantial) reduction of the overhead consists in combining the protocol we propose here with the routing protocol.

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, as explained in 3.4.2 and 3.4.3, the session is re-established because the throughput is not satisfactory.

5.2 Computation overhead

In this subsection, 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 [31], we can consider the battery consumption due to cryptographic computations as negligible compared to the energy needed for data transmission.

[16]The shape of the simulated cell is therefore a square; in fact, the specific shape does not significantly affect the results of the presented simulations.

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

Packet Sending Phase: 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 [14]. Moreover, for each packet, the source has to perform one MAC computation and the destination has to perform one MAC verification.

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 Panama Cipher (little endian) algorithm at 120.301 Mbytes/s [14]. These speeds are to exemplify the range; if slower (or faster) processors are used, it 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 multi-hop cellular networks.

Cooperation in multi-hop cellular networks: In [23], Lamparter et al. propose a charging 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, [23] and our current approach present two main differences. First of all, in [23], the authors analyse 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 [23] are based on public-key cryptography, whereas our solution is based entirely on symmetric key cryptography, which is more suitable for resource constrained mobile devices.

In [20], we have proposed a micro-payment scheme for multi-hop cellular networks that encourages collaboration in packet forwarding. However, our current proposal significantly differs from [20] in many aspects. First of all, in [20], we assume an asymmetric communication model, where the up-stream communication is potentially multi-hop and the down-stream communication is *always* single-hop, whereas in this paper, both the up-stream and the down-stream communications are potentially multi-hop. Second, in [20], the nodes report a fraction of their packet forwarding actions (on a probabilistic basis) to an accounting center that determines how the nodes are remunerated. The approach we propose 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 [20] includes routing decisions, whereas the protocols that we propose in this paper are independent of routing.

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 [26], 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 [28] approaches propose a reputation based solution to identify and punish misbehaving nodes. In [36], 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 [7, 8], Buttyán and Hubaux use a virtual

currency (nuglets) to charge/reward the packet forwarding service provision in ad hoc networks.

7 Conclusion

In this paper, we proposed a set of protocols that fosters cooperation for the packet forwarding service in multi-hop cellular networks. Our solution is based on the charging and rewarding of the nodes and relies exclusively on symmetric cryptography techniques 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 multi-hop cellular networks. Finally, we have analyzed the robustness of our protocols against rational and malicious attacks and have shown that our solution thwarts rational attacks and detects malicious attacks.

As future work, we intend to extend our protocols to include routing misbehavior. We will consider techniques that aim at the calibration of the relevant parameters, and 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 case where several operators exist in the network.

A Identification of the attackers

In this paragraph, we give an idea about how the operator can identify an attacker $\mathcal{A}$ that performed one of the four attacks analyzed in Section 4.

Packet dropping attack

- *Session setup phase*: As explained in Subsection 4.1, dropping the request message is not considered as an attack, so there is no need for the operator to identify the "misbehaving" node. The same reasoning can be applied to the dropping of the reply message because the operator cannot distinguish it from the dropping of the request message. If the attacker $\mathcal{A}$ drops the confirmation message, it can be identified in the same way as for the dropping of a packet during the packet sending phase (see the proof of the Proposition 3). Note that the effect of this last attack lasts for two time periods at most.
- *Packet sending phase*: If $\mathcal{A}$ repeatedly drops packets it is asked to forward, it can be identified as explained in the proof of Proposition 3.
- *Payment redemption phase*: As explained in Subsection 4.1, this attack is inefficient and thus there is no need to identify the attacker.

Replay attacks

As explained in Subsection 4.2, this attack is inefficient.

Filtering attack

- *Session setup phase*: As for the packet dropping attack, an attacker $\mathcal{A}$ that tampers with the request message or the reply message causes the re-establishment of the session using an alternative route. However, the operator may want to identify $\mathcal{A}$ because, contrarily to the packet dropping attack that is part of the decision making of the nodes, tampering with one of these messages before forwarding it is purely malicious. Details about the statistical identification of the attacker are in Appendix C. Tampering with the confirmation message is equivalent to the packet dropping attack and $\mathcal{A}$ can be identified in the same way.
- *Packet sending phase*: If $\mathcal{A}$ tampers with a packet Pkt_ℓ, we can have one of the following cases:

- $\mathcal{A}$ tampers with one of the fields used by the intermediate nodes to generate the receipts ($SSID$ or ℓ for the up-stream and $DSID$, ℓ or $Body_{j,\ell}$ for the down stream). The attacker $\mathcal{A}$ can be identified as explained in Appendix D.
- The *early duplicate* attack can be performed by an attacker that is not even part of the network. However, if the solution proposed in Subsection 4.3 is applied, the attack becomes inefficient.
- In Subsection 4.3, we proved that the *free-riding* attack is malicious. However, if two colluders $\mathcal{A}_1$ and $\mathcal{A}_2$ decide to perform it anyway, $\mathcal{A}_1$ can be identified as explained in Appendix D (the case where the packet is dropped by $\mathcal{A}_2$ is presented in Subsection B).
- An attacker $\mathcal{A}$ that modifies only the receipt field in an up-stream packet can be identified by the operator as explained in Appendix D.
- An attacker $\mathcal{A}$ that modifies a packet without altering the information used to generate the receipts can be statistically identified by the operator as explained in Appendix C.

- *Payment redemption phase*: As explained in Subsection 4.3, this attack is inefficient.

Emulation attack

The emulation attack can be detected by the operator if the compromised node seems to be in two different locations at the same time (e.g., in two non-adjacent cells). Note here that the operator does not identify the attacker but the compromised node. As the operator cannot distinguish between the emulation attack and the case where colluders exchanged their secret information, it is reasonable to assume that the operator will give the node the benefit of the doubt and change its secret key. However, if the same problem happens again, the legitimate subscriber becomes suspect and the operator can decide to terminate his contract and to exclude him definitely from the network.

B Description of a hybrid Attack

Two colluders $\mathcal{A}_1$ and $\mathcal{A}_2$ are on the same route and perform the filtering attack and the packet dropping attack, respectively.

- 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 $\mathcal{A}_2$ as being the attacker (see the proof of Proposition 3).
- If, on the contrary, the nodes that are between $\mathcal{A}_1$ and $\mathcal{A}_2$ are not able to generate valid receipts, then $\mathcal{A}_1$ will be identified by the operator as an attacker that performed a filtering attack (see Appendix A).

The same reasoning can be applied to the case where there are more than two colluders.

C Statistical identification of an attacker

As the operator does not know the identity of the attacker $\mathcal{A}$, it will suspect all the nodes in the route. The operator keeps the information about the suspected nodes and identifies the nodes that are the most frequently suspected. $\mathcal{A}$ will thus eventually be identified as being an attacker and probably be punished. Note here that the identification of $\mathcal{A}$ is not possible if the attack is not repeated a sufficient number of times; but the attack is less harmful if performed only a few times.

D Identification of an attacker using the receipts

The nodes that are before the attacker $\mathcal{A}$ in the route are able to provide the operator with valid receipts for the packet, whereas the nodes that are after $\mathcal{A}$ will provide the operator with invalid receipts. $\mathcal{A}$ and the node after it in the route are thus suspected by the operator and if $\mathcal{A}$ repeats the attack a substantial number of times (when belonging to different routes), the operator will identify it as a malicious node.

References

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

[2] Y. Bejerano. Efficient Integration of Multi-Hop Wireless and Wired Networks with QoS Constraints. In *Proceedings of Mobicom*, pages 215–226. ACM Press, September 2002.

[3] N. Ben Salem, L. Buttyán, J.-P. Hubaux, and M. Jakobsson. A Charging and Rewarding Scheme for Packet Forwarding in Multi-hop Cellular Networks. In *Proceedings of MobiHOC*, Annapolis, USA, 2003.

[4] L. Blazevic, L. Buttyán, S. Capkun, S. Giordano, J.-P. Hubaux, and J.-Y. Le Boudec. Self-Organization in Mobile Ad-Hoc Networks: the Approach of Terminodes. *IEEE Communications Magazine*, 39(6), June 2001.

[5] S. Buchegger and J.-Y. Le Boudec. Performance Analysis of the CONFIDANT Protocol: Cooperation Of Nodes - Fairness In Distributed Ad Hoc NeTworks. In *Proceedings of MobiHOC*, Lausanne, CH, June 2002.

[6] L. Buttyán and J.-P. H. (Eds). Report on a Working Session on Security in Wireless Ad Hoc Networks. *ACM Mobile Computing and Communications Review (MC2R)*, 7(1), January 2003.

[7] L. Buttyán and J.-P. Hubaux. Enforcing Service Availability in Mobile Ad Hoc WANs. In *Proceedings of MobiHOC*, Boston, MA, USA, August 2000.

[8] L. Buttyán and J.-P. Hubaux. Stimulating Cooperation in Self-Organizing Mobile Ad Hoc Networks. *ACM/Kluwer Mobile Networks and Applications (MONET)*, 8(5), October 2003.

[9] T. Camp, J. Boleng, and V. Davies. A Survey of Mobility Models for Ad Hoc Network Research. *Wireless Communication and Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, 2(5):483–502, 2002.

[10] D. Coppersmith and M. Jakobsson. Almost Optimal Hash Sequence Traversal. In *Proceedings of Financial Cryptography*, 2002.

[11] B. Goode. Voice Over Internet Protocol (VoIP). *Proceedings of the IEEE*, 90:1495–1517, September 2002.

[12] H. Holma and A. Toskala. *WCDMA for UMTS: Radio Access for Third Generation Mobile Communications*. Wiley, 2002.

[13] H.-Y. Hsieh and R. Sivakumar. Towards a Hybrid Network Model for Wireless Packet Data Networks. In *Proceedings of ISCC*. IEEE, 2002.

[14] http://www.eskimo.com/˜weidai/benchmarks.html.

[15] http://www.terminodes.org.

[16] Y.-C. Hu, D. B. Johnson, and A. Perrig. SEAD: Secure Efficient Distance Vector Routing for Mobile Wireless Ad Hoc Networks. In *Proceedings of WMCSA 2002*, Portorosz, Slovenia, June 2002.

[17] Y.-C. Hu, A. Perrig, and D. B. Johnson. Ariadne: A Secure On-Demand Routing Protocol for Ad Hoc Networks. In *Proceedings of Mobicom*. ACM Press, September 2002.

[18] Y.-C. Hu, A. Perrig, and D. B. Johnson. Packet Leashes: A Defense against Wormhole Attacks in Wireless Ad Hoc Networks. In *Proceedings of INFOCOM*. IEEE, 2003.

[19] J.-P. Hubaux, T. Gross, J.-Y. Le Boudec, and M. Vetterli. Towards Self-Organizing Mobile Ad-Hoc Networks: the Terminodes Project. *IEEE Communications Magazine*, 39(1):118 –124, January 2001.

[20] M. Jakobsson, J.-P. Hubaux, and L. Buttyán. A Micro-Payment Scheme Encouraging Collaboration in Multi-Hop Cellular Networks. In *Proceedings of Financial Cryptography*, 2003.

[21] D. B. Johnson and D. A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In Imielinski and Korth, editors, *Mobile Computing*, chapter 5, page 153181. Kluwer Academic Publishers, 1996.

[22] M. Kubisch, S. Mengesha, D. Hollos, H. Karl, and A. Wolisz. Applying Ad-hoc Relaying to Improve Capacity, Energy Efficiency, and Immission in Infrastructure-based WLANs. In *Proceedings of Kommunikation in Verteilten Systemen (KiVS 2003)*, Leipzig, Germany, February 2003.

[23] B. Lamparter, K. Paul, and D. Westhoff. Charging Support for Ad Hoc Stub Networks. *Journal of Computer Communication, Special Issue on Internet Pricing and Charging: Algorithms, Technology and Applications, Elsevier Science*, Summer 2003.

[24] Y.-D. Lin and Y.-C. Hsu. Multihop Cellular: A New Architecture for Wireless Communications. In *Proceedings of INFOCOM*, 2000.

[25] O. C. Mantel, N. Scully, and A. Mawira. Radio Aspects of Hybrid Wireless Ad Hoc Networks. In *Proceedings of VTC*. IEEE, 2001.

[26] S. Marti, T. Giuli, K. Lai, and M. Baker. Mitigating Routing Misbehavior in Mobile Ad Hoc Networks. In *Proceedings of Mobicom*, 2000.

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

[28] P. Michiardi and R.Molva. Core: A Collaborative Reputation Mechanism To Enforce Node Cooperation In Mobile AD HOC Networks. In *Proceedings of The 6th IFIP Communications and Multimedia Security Conference*, Portorosz, Slovenia, September 2002.

[29] P. Papadimitratos and Z. J. Haas. Secure Routing for Mobile Ad Hoc Networks. In *Proceedings of CNDS*, January 2002.

[30] K. Paul and D. Westhoff. Context Aware Inferencing to Rate a Selfish Node in DSR based Ad-hoc Network. In *Proceedings of GLOBECOM*, November 2002.

[31] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. D. Tygar. SPINS: Security Protocols for Sensor Networks. In *Proceedings of Mobicom*, Rome, Italy, July 2001.

[32] H. Wu, C. Qios, S. De, and O. Tonguz. Integrated Cellular and Ad Hoc Relaying Systems: iCAR. *IEEE Journal on Selected Areas in Communications*, 19(10), October 2001.

[33] J. Yoon, M. Liu, and B. Noble. Random Waypoint Considered Harmful. In *Proceedings of INFOCOM*. IEEE, 2003.

[34] A. N. Zadeh, B. Jabbari, R. Pickholtz, and B. Vojcic. Self-Organizing Packet Radio Ad Hoc Networks with Overlay (SOPRANO). *IEEE Communications Magazine*, June 2002.

[35] M. G. Zapata and N. Asokan. Securing Ad-Hoc Routing Protocols. In *Proceedings of WiSe 2002*, Portorosz, Slovenia, September 2002.

[36] S. Zhong, Y. R. Yang, and J. Chen. Sprite: A Simple, Cheat-Proof, Credit-Based System for Mobile Ad Hoc Networks. In *Proceedings of INFOCOM*. IEEE, 2003.