A Charging and Rewarding Scheme for Packet Forwarding in Multi-hop Cellular Networks*

Naouel Ben Salem Levente Buttyán† Jean-Pierre Hubaux
Laboratory of Computer Communications and Applications (LCA)
Swiss Federal Institute of Technology Lausanne (EPFL)
Switzerland

naouel.bensalem@epfl.ch buttyan@hit.bme.hu
jean-pierre.hubaux@epfl.ch

Markus Jakobsson
RSA Laboratories
174 Middlesex Turnpike
Bedford, MA 01730, USA

mjakobsson@
rsasecurity.com

ABSTRACT

In multi-hop cellular networks, data packets have to be relayed hop by hop from a given mobile station to a base station and vice-versa. This means that the mobile stations must accept to forward information for the benefit of other stations. In this paper, we propose an incentive mechanism that is based on a charging/rewarding scheme and that makes collaboration rational for selfish nodes. We base our solution on symmetric cryptography to cope with the limited resources of the mobile stations. We provide a set of protocols and study their robustness with respect to various attacks. By leveraging on the relative stability of the routes, our solution leads to a very moderate overhead.

Categories and Subject Descriptors

C.2.2 [**Computer Systems Organization**]: Computer-Communication Networks—*Network Protocols*

General Terms

Algorithms, Design, Security

Keywords

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

*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

†Now with Budapest University of Technology and Economics, Department of Telecommunications, Hungary.

MobiHoc'03, June 1–3, 2003, Annapolis, Maryland, USA.

1. INTRODUCTION

Multi-hop cellular networks combine the characteristics of both cellular and wireless mobile ad hoc networks. As with cellular networks, the geographic area covered by the network is populated with base stations that are connected to each other via a backbone. However, in multi-hop cellular networks, the existence of a communication link between the mobile station and the base station is not required; Indeed, in this new approach, the communication between the mobile station and the base station is, in general, relayed by a (usually small) number of other mobile stations. Multi-hop cellular networks can be perceived as an evolution of either voice-centric networks such as GSM or data-centric networks such as CDPD or IEEE 802.11.

Several benefits can be expected from the use of this new family of networks [23, 13, 24, 21]: (i) the number of fixed antennas can be smaller; (ii) the energy consumption of the nodes can be reduced, as the distance the signal has to cover is smaller[1]; (iii) the interference with other nodes diminishes as well; (iv) the coverage of the network can be increased.

The proper operation of the network requires the mobile nodes to collaborate with each other. Such a constraint does not exist in conventional cellular networks. In particular, it is clear that a node must be "encouraged" in some way to relay information for the benefit of other nodes. In the absence of such an incentive, a dishonest user could tamper with his mobile device to convert it into a purely "selfish one"; this tampering is maybe beyond the ability of the casual user, but certainly not of a dishonest "techie" or organization. In this paper, we provide an in-depth study of the problem and we propose a solution based on appropriate charging and rewarding mechanisms of the nodes.

The work presented in this paper is part of the Terminodes Project [18, 3, 15]. The rest of the paper is organized as follows. We describe the related work in section 2 and introduce our system and adversarial model in Section 3. In Section 4, we describe our approach and detail our requirements. In Section 5, we provide a detailed description of the protocols. In Sections 6 and 7 we discuss the security and the overhead of the solution, respectively. Finally, we present our conclusions and future work in Section 8.

[1]The transmission energy grows more than linearly with the distance at which the signal must be received.

2. RELATED WORK

In this section, we discuss some works related to the issues of multi-hop cellular networks, secure source routing and cooperation of nodes in (pure) ad hoc networks and in multi-hop cellular networks.

- **Multi-hop cellular networks:** In [1], Aggelou et al. describe the Ad Hoc GSM (A-GSM) system, which integrates relaying functions in GSM cellular network, focusing on the network layer, whereas in the SOPRANO project [33], Zadeh et al. investigate the self-organization aspect of such a network. Opportunity Driven Multiple Access (ODMA) allows terminals to use multi-hop communications to reach the base station if they are beyond the reach of the cell coverage [12]. In both [2] and [31], Bejerano and Wu et al. study the benefit of installing relaying stations (also called access points) to provide the nodes with an access to a backbone.

- **Secure Source Routing:** In [16], Hu et al. present a secure on-demand ad hoc network routing protocol (Ariadne) that uses symmetric cryptographic primitives to prevent attacks in mobile ad hoc networks. They present an attacker model and analyze the robustness of their protocol against various attacks. In [17], they consider the *wormhole attack* and propose a mechanism, called *packet leashes*, to detect this specific attack in mobile ad hoc networks. In [28], Papadimitratos and Haas propose a secure route discovery protocol that provides the nodes with accurate connectivity information, despite the existence of malicious nodes in the network. They study the robustness of the protocol against non-colluding attackers. In [29], Paul and Westhoff propose a mechanism to detect routing misbehavior in DSR based ad hoc networks and to inform the nodes about the identity of the attacker(s). An overview of security in wireless mobile ad hoc networks is presented in [7].

- **Cooperation in ad hoc networks:** Several research groups have considered the problem of selfishness and stimulation of cooperation in mobile ad hoc networks. In [25], 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 [27] approaches propose a reputation based solution to identify and punish misbehaving nodes. In [6, 8], Buttyán and Hubaux use a virtual currency called *nuglets* to charge and reward the provision of the packet forwarding service in ad hoc networks while in [34], Zhong et al. rely on a central authority that collects receipts from the forwarding nodes and charges/rewards the nodes based on these receipts.

- **Cooperation in multi-hop cellular networks:** In [22], Lamparter et al. propose a charging scheme to motivate cooperation in hybrid networks (i.e., mobile ad hoc networks with access to the Internet, what they call "stub ad hoc networks"). They assume the existence of an Internet Service Provider that authenticates the nodes involved in a given communication (using public key cryptography) and takes care of charging or rewarding them.

  In [19], 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 [19] in many aspects. First of all, in [19], we assume an asymmetric communication model, where the up-stream (mobile to base station) communication is potentially multi-hop and the down-stream (base station to mobile) communication is *always* single-hop, whereas in this paper, the communication model is symmetric, meaning that both the up-stream and the down-stream communications are potentially multi-hop. Second, in [19], 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 based on the received reports. The approach proposed in this paper does not rely on reports; instead, we use the concept of session during which each node involved in the communication (i.e., the initiator, the correspondent and the forwarding nodes) authenticates itself to the base station by altering the packet to be forwarded in a specific way. Finally, the protocol proposed in [19] includes routing decisions, whereas the protocols that we propose in this paper are independent of routing, and hence, can be used with existing (secure) source routing schemes.

3. MODEL

3.1 System and trust model

Our 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 initiator of the communication pays for the provided service[2]. We also assume that the links are symmetric (i.e., the nodes and the base stations have the same power range).

The geographical area under the control of a given base station is called a *cell* [23]. The power range of the base station is smaller than the radius of the cell. This means that some nodes in the cell may not be able to communicate with the base station directly; communication between these nodes and the base station is based on *multi-hop relaying* performed by intermediate mobile nodes.

When a mobile node A (the initiator) wants to establish a communication with another mobile node B (the correspondent), it first has to establish an *end-to-end session* with B (as we will see in detail in Subsection 5.2, 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* be-

[2]In this paper, we consider the case where the initiator is the main beneficiary of the service and thus should pay for it (e.g., e-mail). The proposed solution can easily be adapted to the case where the correspondent benefits from the service.

tween 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 can be A or B) and D its destination (which is therefore B or A, respectively). The base stations of the source and of the destination are called BS_S and BS_D, respectively. The packet is then sent, first from S to BS_S (if necessary in multiple hops using intermediate mobile nodes as relays). If S and D reside in two different cells, then the packet is forwarded by BS_S to BS_D via the backbone. Finally, the packet is sent from BS_D to D (if necessary in multiple hops again)[3]. The protocols remain the same if the packet is sent from initiator A to the correspondent B or from B to A, except for the charging and remuneration phase (see Subsection 5.7).

We assume an ad hoc model in which mobile nodes move. However, mobility should not be so high that the routing has to be performed on packet based granularity, but rather on a *session-to-session* basis. When A wants to communicate with B, two routes are established (using source routing): the *initiator route* (from A to BS_A) and the *correspondent route* (from BS_B to B). These routes are used either until the end of the session, or until one of them is broken by the movement of one or more mobile nodes on the path (including A and B). If one of these routes is broken during the session, then a new route is established and used in the same manner as the first one.

Note that our system model described above is similar to that of [23] with the difference that we require *all* communication to pass through a base station. Although this may lead to suboptimal routes when the initiator and the correspondent are not neighbors but are close to each other, 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 (the route to the base station) instead of one route per potential correspondent. In other words, if the correspondent B is not a neighbor, then the initiator A simply sends the packet towards the base station and does not need to run route discovery to find a route to B. Of course, a base station has to maintain a route to every and every node in its cell. This can be based on a traditional source route discovery protocol such as DSR [20] or alternatively, the base station can obtain down-link routing information from up-link packets (packets sent by the source S to BS_S) assuming that those packets carry the route (i.e., source routing is used) and that the links are bidirectional.

In order to make the presentation easier, we assume that all the base stations and the backbone are operated by a *single operator*, although our proposal can easily be extended to the general case where the base stations and the backbone(s) are under the control of several operators. We further assume that the network operator is fully trusted by all mobile nodes, be it for charging, for route setup, or for packet forwarding. The network operator is the authority who tries to make sure that all users obey the rules, and in particular that they respect the charging mechanism described hereafter.

[3]Clearly, the communication ends (on the initiator side and on the correspondent side) are not necessarily both wireless. However, in this paper, we assume that to be the case as this corresponds to the most complete scenario.

3.2 Adversarial model

A coordinating adversary: Although our system must tolerate the existence of many independent attackers, we use the common cryptographic approach of considering all of these attackers to be controlled by one and the same adversary. Note that this is a pessimistic assumption, as it increases the power of the attackers; seen another way, it is clear that any security guarantees that are possible in such a model can be transferred to a model in which individual attackers work independently of each other.

A rational adversary: As we will consider and secure our protocols against very strong attacks, it should be emphasized that our protocols will not be secure against arbitrary malicious behavior. Instead, a principal assumption we will make is that the adversary is *rational*, in that it will only attempt to cheat if the expected benefit of doing so is greater than the expected benefit of acting honestly. This is reasonable, in particular, given that the underlying routing structure (given any existing secure source routing protocol) is not believed to withstand arbitrary attacks. An adversary will determine the expected monetary benefit of a certain set of actions, along with the expected degree of service (namely of being able to use the communication infrastructure to send messages) and the cost of his actions (measured in the required battery and CPU resources). Rational behavior requires him to select the behavior that minimizes his costs and maximizes his benefit.

Incentives – a double-edged sword: It should be noticed that the introduction of monetary rewards in the system may act not only as an incentive for collaboration, but also (for a poorly designed system) as an incentive for cheating. Note that this applies to the underlying routing scheme as well: if it is rational for an adversary to corrupt routing tables (both his own and those of others), then he will attempt to do just that. However, securing the routing protocol is beyond the scope of this paper and thus we assume that routing tables are not corrupted. As we have seen in Section 2, several solutions to secure source routing in ad hoc networks exist (e.g. [16], [28] or [29]) and can be used as underlying routing mechanism in our protocol.

A two-tiered model: As in many other systems needing to protect themselves against theft of service, it makes sense to consider two types of attacks, which we will refer to as *filtering* and *invasive* attacks.

- **Filtering attack:** In this type of attack, the adversary can observe and modify both the input and the output of each device he controls, but he cannot extract the secret information from these devices. One can think of this as an external filtering of the data, but it should be made clear both that it does not have to be external (but may correspond to a change of software, where applicable), and that filtering does not mean a simple removal of data, but that the adversary may *add* and *modify* data as well. The functional description of the adversarial behavior corresponds closely to the common assumption of tamper-proofness.
- **Invasive attack:** In this type of attack, the adversary may – in addition to what he can do in the filtering attack – extract all secret information of the controlled devices and distribute this information to the individual devices, incorporating this into the filtering mechanisms of these.

For both of these attack models, we assume that the network operator may obtain access to one or more of the corrupted devices, allowing them to determine its exact contents, i.e., the filter algorithm and its data (it is important to notice that the security of our scheme does not rely on this assumption; although, it does benefit from it).

Both of the above attack models are worthwhile to consider. To allow the best possible analysis of our proposed system, we consider a *two-tiered* adversarial model, corresponding to these two types of attack. The development and use of such a two-tiered model allows us to produce and present very precise security statements. Due to space limitations, we place an emphasis on the filtering attack herein and only briefly describe the tools needed to obtain security against the invasive attack.

For concreteness, let us mention a few attacks that the adversary may try to perform and that will be covered by our security model:

- **Refusal to pay:** A party may claim that he did not initiate some communication, and therefore should not pay for it.
- **Dishonest rewards:** A party may try to make it appear that he was involved in the forwarding of certain packets – potentially with the help of a colluder that really did handle the packets – and therefore demand to be remunerated.
- **Free riding:** Two dishonest parties on the route between two honest parties may attempt to piggyback data on the packets sent between the honest parties, with the goal of not having to pay for the communication. This attack is not limited to the appending of data, but may also involve substitution. In view of our approach to charging (see Subsection 5.7), we can also identify an even more serious form of free riding in which the initiator passes information free of charge to a colluding intermediate node F on the up-stream route to the base station by sending packets to a fictive destination. The packets are routed through F, who simply drops them after extracting the payload. Since the base station received no packet, the initiator will not be charged and the forwarding nodes will thus not be remunerated.

We propose a protocol that provides security against the above attacks, and at the same time, offers incentives for collaboration.

4. APPROACH AND REQUIREMENTS

In order to motivate the nodes to forward traffic for the benefit of other nodes, we propose to remunerate the forwarding nodes. Remuneration is made possible by charging the initiator A of the communication (for the traffic in both directions, i.e., from A to the correspondent B or from B to A) and rewarding the forwarding nodes (and the operator). 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 is then implemented by manipulating the appropriate billing accounts.

A question that arises at this point is "Under what circumstances exactly should the initiator A be charged for a packet sent by a source S to a destination D (where one of these entities is the initiator A and the other is the correspondent B)?". One option would be to charge A only if the packet has successfully been delivered to D. The problem here is that due to the multi-hop nature of the communication, the packet may be delivered to D by an intermediate mobile node instead of the base station BS_D, which means that the operator may have no reliable information about the delivery. This may require D to acknowledge the packet, but the source and the destination can collude and agree not to send acknowledgments. In this way, the initiator would escape from being charged, and if it provides the correspondent with a fraction of the saved charge, or it returns the favor to the correspondent when they play opposite roles, then they are both better off by cheating than behaving honestly. Therefore, we decided to charge the initiator before the packet is delivered to the destination. Since we require every packet to pass through a base station, it is convenient to charge the initiator when the base station of the source BS_S receives the packet.

As for crediting the forwarding nodes, the up-stream forwarding nodes (i.e., the nodes that are in the route between S and BS_S) are remunerated when the packet is received by the base station BS_S, whereas the down-stream forwarding nodes (i.e., the nodes that are in the route between BS_D and D) are remunerated only if the packet is delivered to the destination D. As we saw in the previous paragraph, the operator may not have reliable information about the delivery of the packet, therefore we require the destination to acknowledge the reception of the packet. Note, however, that in this case, not sending the acknowledgement does not help the initiator to escape from being charged, since it has already been charged. Therefore, the only reason for not sending the acknowledgement is for D to save resources. In order to overcome this problem and motivate the destination to send acknowledgements, the destination of the packet is charged for a small amount ε when the base station BS_D injects the packet in the down-stream route and reimbursed when the acknowledgment is received by a base station. The acknowledgement scheme that we propose is lightweight and off-line (as explained in Subsection 5.6).

The essence of our proposal is a set of protocols that allow the operator to collect information based on which it can decide which accounts should be charged and which accounts should be credited. Our design seeks to satisfy the following requirements:

- **Authentication of the source and the destination.** The operator should be able to reliably identify the source and the destination of every packet received by the base stations in order to know which account to charge. By satisfying this requirement, we want to prevent *refusal to pay* attacks.
- **Authentication of the forwarding nodes.** The operator must be able to reliably identify the forwarding nodes (both in the initiator and in the correspondent routes) during the session setup phase, and later verify that this exact set of nodes participated in the forwarding of each packet. The latter may be done either using standard authentication techniques, or, as we propose in our scheme, using what one may think of as an "implicit" authentication mechanism (see the usage of the cipher stream encryptions/decryptions in Subsection 5.3).

For efficiency reasons, our protocols are entirely based on symmetric key cryptography. Although asymmetric cryptographic primitives, especially digital signatures, 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.

We require each node to register with the operator in order to be able to use the system. When a node i registers, it receives a long-term symmetric key K_i, which is shared between i and the operator (i.e., all base stations). K_i is the only long-term cryptographic material stored in the node; it is used to authenticate i either as a source, as a destination, or as a forwarder of packets.

We assume that source routing is used in the network to determine a packet's path from the source to the base station BS_S and from the base station BS_D to the destination, otherwise our protocols are independent of the routing protocol. We note that with minor modifications, our proposal can be extended to work with other types of routing algorithms as well (e.g., distance vector routing). This means that our scheme could potentially be used with any existing and future ad hoc routing protocols.

5. PROTOCOLS

5.1 Building blocks and notation

Our protocols use two cryptographic building blocks: a MAC (Message Authentication Code[4]) function and a stream cipher.

- **Message Authentication Codes:** A MAC function is a one-way function that takes as input a message m and a key K, and outputs a digest $\mu = MAC_K(m)$. Here, μ depends on both m and K; does not reveal any information about the key K; and cannot be computed without knowledge of the same. MACs are used for allowing the integrity and authenticity of data.

- **Stream cipher:** A stream cipher function is a one-way function that takes as input a short seed σ and produces an arbitrarily long sequence of pseudo-random bits from this seed. By XORing a segment of the output of a stream cipher to a plaintext, one obtains a ciphertext; this can be decrypted (to obtain the initial plaintext) by XORing the same segment of the output to the ciphertext. Note that if two parties both know the seed σ (as is common for this application), this technique can be used to encrypt – and decrypt – messages of arbitrary and variable length. If, in addition, the output stream is *indexable* by means of a position *pos*, it is – given this indexing position – easy to decrypt ciphertexts out of order, even if some are lost.

Further notation: In the following description, we will denote the concatenation operator by $\mid$ and the XOR operator by $\oplus$. In the session setup phase (see Subsection 5.2), we denote the initiator and the correspondent of a given communication by A and B, respectively. We call BS_A the base station of the initiator and BS_B the base station of the correspondent. The number of relaying nodes in the *initiator route* (from A to BS_A) and in the *correspondent route* (from BS_B to B) are denoted by a and b, respectively. In the packet sending phase (see Subsection 5.3), we denote the source and the destination of a given packet by S and D, respectively. We call BS_S the base station of the source and BS_D the base station of the destination. The number of relaying nodes in the *up-stream route* (from S to BS_S) and in the *down-stream route* (from BS_D to D) are denoted by s and d, respectively.

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

5.2 Session setup

When a node A wants to communicate with another node B, it first has to set up an *end-to-end session*. The goal of the session setup is to test the initiator and the correspondent routes obtained from the underlying source routing protocol, to inform the nodes on these routes about the traffic that will follow and to let the base stations authenticate all nodes belonging to these routes. Any node on the routes from A to BS_A and from BS_B to B may decide not to join the session, in which case the session setup fails. If A or BS_B has an alternative route to the target (BS_A for the initiator route and B for the correspondent route), it may try to establish the session using that route. Successful completion of the session setup phase is a confirmation that both the initiator and the correspondent routes are available and that all the intermediate nodes on the routes accept to forward the traffic.

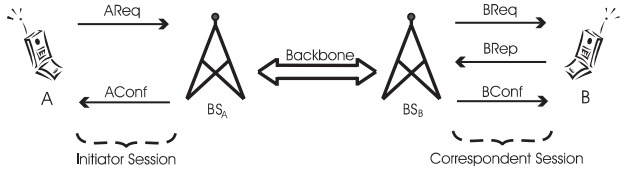

Figure 1: The session setup phase

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 initiator route *ARoute* from A to BS_A (obtained from the secure source routing protocol), and some information about the traffic[5] to be sent (*TrafficInfo*). 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 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 on the initiator route checks the traffic information *TrafficInfo*. If i decides to participate in

[5]The initiator A may have no precise information about the traffic the correspondent B intends to generate. *TrafficInfo* is thus an estimation of the expected traffic in both directions. Note that it is not to the benefit of A to underestimate the traffic, because the nodes will most likely interrupt the packet forwarding service provision if the amount of data to forward is much larger than expected.

the forwarding of the specified traffic, then it computes a MAC on the whole request (including the MAC already in the request) using its own key K_i, replaces the MAC in the request with the newly computed MAC, and forwards the request. In other words, each node i ($1 \leq i \leq a$) on the initiator route sends the request message $AReq_i$ to the next intermediate node on the route (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 MAC that was computed by A and all the nodes on the route in an iterative manner. From a cryptographic point of view, our "MAC layering" technique achieves a similar effect to that where each node on the route attached its own MAC to the request. But the traditional solution of attaching a new MAC would increase the size of the request, whereas in our case, the size of the request remains constant. Therefore, our technique is more efficient in terms of bandwidth. To the best of our knowledge, such a scheme has not been proposed yet for ad hoc networks.

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 these verifications are not successful, then the base station BS_A drops the request. Otherwise, it sends the request, via the backbone, to the base station BS_B that generates and sends a correspondent session setup request $BReq_0$ towards the correspondent where:

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

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 (set to zero in case of a new session establishment), and *BRoute* is the source route between BS_B and the correspondent B.

In case of initiator session re-establishment, it is not necessary to also re-establish the correspondent session if it is still valid. BS_A then generates and sends an initiator session setup confirmation message *AConf* towards the initiator (as explained hereafter) and both base stations (BS_A and BS_B) link the new initiator session ID to the existent correspondent session ID by updating the information about the session in the operator's database.

The correspondent session setup request is processed by the forwarding nodes on the correspondent route in the same way as for the initiator session setup request. Each downstream forwarding node j ($1 \leq j \leq b$) computes and sends $BReq_j$ where:

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

When the correspondent 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:

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

Thus, the reply *BRep* contains a single MAC that was computed by all the nodes on the correspondent route and B in an iterative manner. The reply is relayed back without any modification to BS_B on the reverse route of the request. The base station BS_B checks the "layered" MAC in the reply, and if it verifies correctly, BS_B informs BS_A that the session is valid and both BS_A and BS_B send an initiator (respectively correspondent) session setup confirmation message towards A (respectively B). The initiator session setup confirmation message *AConf* contains the initiator request ID *AReqID*, a freshly generated random number representing the initiator session ID *ASID* and a series of MACs where each MAC is intended for one of the nodes on the initiator route and one MAC is for A:

$$AConf = [\ AReqID \mid ASID \mid AMAC_A \mid AMAC_1 \mid \ldots \mid AMAC_a\ ]$$
$$AMAC_i = MAC_{K_i}(AReqID \mid ASID \mid oldASID \mid ARoute \mid TrafficInfo)$$

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

$$BConf = [\ BReqID \mid BSID \mid BMAC_1 \mid \ldots \mid BMAC_b \mid BMAC_B\ ]$$
$$BMAC_j = MAC_{K_j}(BReqID \mid BSID \mid oldBSID \mid BRoute \mid TrafficInfo)$$

Each node i on the initiator and correspondent routes (including A and B) verifies its own AMAC or BMAC and stores the initiator or correspondent session ID, respectively. The state information related to the established initiator and correspondent sessions (including session IDs, routes, and cryptographic parameters) is stored in the operator's database.

In case of correspondent session re-establishment, it is not necessary to also re-establish the initiator session if it is still valid. BS_B thus establishes the correspondent session as explained previously and base stations BS_A and BS_B link the new correspondent session ID to the existent initiator session ID.

5.3 Packet sending

Once the session has been set up, the source S (who can be interchangeably A or B) starts sending packets to the destination D (who is therefore B or A, respectively). The ℓ-th packet $SPkt_{0,\ell}$ sent by S contains the session ID *SSID* (which is *ASID* if $S = A$ and *BSID* if $S = B$), the sequence number ℓ, and the payload $Payload_\ell$. In addition, the integrity of the packet is protected and authentication of its source is provided by a MAC computed on the packet using the secret key K_S:

$$SPkt_{0,\ell} = [\ SSID \mid Body_{0,\ell}\ ]$$
$$Body_{0,\ell} = \ell \mid Payload_\ell \mid MAC_{K_S}(SSID \mid \ell \mid Payload_\ell)$$

The MAC in $SPkt_{0,\ell}$, however, can and will only be verified by BS_S; intermediate nodes will simply ignore it. Instead of verifying the MAC, each node i ($1 \leq i \leq s$) on the up-stream route encrypts the body of the packet (including the MAC) by XORing it with the pad $PAD_{i,\ell}$:

$$SPkt_{i,\ell} = [\ SSID \mid Body_{i,\ell}\ ]$$
$$Body_{i,\ell} = PAD_{i,\ell} \oplus Body_{i-1,\ell}$$

The pads $PAD_{i,\ell}$ are generated from the session identifier and the secret key K_i of node i in the following way (see Figure 2). The up-stream session identifier *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 actual length L_ℓ of the packet to be encrypted is smaller than *MaxLength*, then only the first L_ℓ bytes of $PAD_{i,\ell}$ are used, the rest of $PAD_{i,\ell}$ is thrown away.

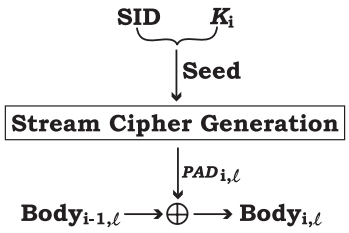

Figure 2: Encryption of the packets

When the base station BS_S receives the packet, it looks up its internal table using *SSID* in the packet as index, retrieves the secret keys of the forwarding nodes on the up-stream route and of the source, recomputes the pads $PAD_{i,\ell}$, and removes all encryptions from the packet. Then, it verifies the MAC of the source in the packet and checks that the sequence number has not already been used (i.e., the packet is not a duplicate). If one of these verifications is not successful, then it drops the packet. Otherwise, it forwards it 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*, computes a new MAC using the secret key of the destination D, computes the pad $PAD_{j,\ell}$ for each forwarding node j ($1 \leq j \leq d$) on the down-stream route, and encrypts the packet (including the MAC) by iteratively XORing it with every pad $PAD_{j,\ell}$. The result is the following:

$$\begin{aligned} DPkt_{0,\ell} &= [\ DSID \mid Body_{0,\ell}\ ] \\ Body_{0,\ell} &= PAD_{1,\ell} \oplus PAD_{2,\ell} \oplus \ldots \oplus PAD_{d,\ell} \oplus Body'_{0,\ell} \\ Body'_{0,\ell} &= \ell \mid Payload_\ell \mid MAC_{K_D}(DSID \mid \ell \mid Payload_\ell) \end{aligned}$$

BS_D stores $MAC_{K_D}(DSID \mid \ell \mid Payload_\ell)$ of every packet it sends together with the corresponding sequence number ℓ in order to be able to verify future destination acknowledgements (see Subsection 5.6).

Upon reception of $DPkt_{j-1,\ell}$, each down-stream forwarding node j 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:

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

Thus, when the packet arrives to D, all encryption pads have been removed, and D can verify the MAC generated by BS_D. D stores the MAC together with the sequence number ℓ for generating an acknowledgement to the base station (see Subsection 5.6).

5.4 Session states

The base stations and every node involved in a session (i.e., the source, the destination, and the up-stream and down-stream forwarding nodes) consider the session to be in one of the following two possible states: active or closed. A session becomes active for the base stations when they send the session setup confirmation messages *AConf* and *BConf*. For every node involved, the session becomes active when it receives a valid *AConf* or *BConf*.

Each entity i involved in a session starts a timer t_i when the session becomes active for it. t_i is restarted each time i receives a valid packet that belongs to the session. The node i closes the session if the timer t_i expires or if i receives one of the error messages related to the session and described in 5.5. For the nodes (except the destination), closing a session means that they can delete the related state information from their memories. The destination and the base station BS_D will keep state information until the destination sends and the base station receives the destination acknowledgement message.

5.5 Error handling

It is possible for a forwarding node i (both in the up-stream and the down-stream route) to explicitly provoke closing the session. This can happen in any of the following situations:

- Sending packets to the next hop on the route fails (i.e., i does not receive any acknowledgment from the next hop at the data link layer). In this case, i generates a *Broken Session Error* and sends it towards the sender (i.e., the source S if i is an up-stream forwarding node, or the base station BS_D if i is a down-stream forwarding node) to inform the previous forwarding nodes and the sender that the route is broken.

- i receives a packet to forward with an unknown session ID. It then generates an *Unknown Session ID Error* and sends it back to the sender.

- For some reason i does not want to (or cannot) participate in the packet forwarding anymore. It then generates a *Cancellation Error* and sends it back to the sender.

When the sender receives any of these error messages, it initiates a session re-establishment using an alternative route.

The forwarding nodes are motivated to relay these error messages; Indeed, as long as the "problematic session" is active, they keep receiving useless packets (these packets will never reach the destination via this specific route, meaning that the forwarding nodes will not get rewarded). Considering that the reception of all these useless packets is energy consuming (probably more than sending a short error message), it is not beneficial for the nodes to discard these error messages.

As these messages are not secured, one possible attack would be to forge a fake error message. But even if we secured these messages, it would always be possible for an attacker to provoke a session closing using other mechanisms (e.g., jamming attacks or denial of service attacks [11]).

5.6 Destination acknowledgement

The destination D must acknowledge the reception of every packet. However, in order to save resources, it does not send acknowledgements on a per packet basis. Instead, it acknowledges all received packets in a single batch when it considers the session to be closed (see Subsection 5.4). D sends the destination acknowledgement *DAck* when it has a good (preferably single-hop) connection to any base station. The format of the acknowledgment message is as follows:

$$DAck = [\ DSID \mid Batch \mid LastPkt \mid LostPkts \mid MAC_{K_D}(DSID \mid Batch \mid LastPkt \mid LostPkts)\ ]$$

where *LastPkt* is the sequence number of the last packet received in the session, *LostPkts* is a set of sequence numbers of missing packets preceding *LastPkt* and

$$Batch = \bigoplus_{\substack{\ell \leq LastPkt \\ \ell \notin LostPkts}} MAC_{K_D}(DSID \mid \ell \mid Payload_\ell)$$

where $MAC_{K_D}(DSID \mid \ell \mid Payload_\ell)$ is the MAC received in the packet with sequence number ℓ.

The rationale of this acknowledgement format is based on the assumption that, in most of the cases, there will only be a few packets missing and therefore *LostPkts* is small. In addition, the most likely reason for packet loss is failure of the route (e.g., because of node mobility), in which case, only packets following *LastPkt* are lost, and these are not included in *LostPkts*.

When a base station receives an acknowledgement, it first verifies the $MAC_{K_D}(DSID \mid Batch \mid LastPkt \mid LostPkts)$. It then checks *Batch* by XORing all the MACs of the packets up to *LastPkt* and excluding those in *LostPkts* and comparing the result to the received value. If the verification fails then the base station ignores the acknowledgement.

5.7 Charging

As we already mentioned in Section 4, charging and remuneration are implemented by the manipulation of the accounts of the nodes; this operation is performed by the network operator. When BS_S receives a packet P_ℓ of length L_ℓ that verifies correctly, then the initiator A is charged for a given amount $n(L_\ell)$ (depending on the packet size). If the correspondent B is the sender of the packet, it is also charged $n(L_\ell)$ to prevent a free-riding attack: as A pays for the communication, B can be tempted to use the session to communicate freely with a colluding node that is in the route to A. The colluder would receive the packets and drop them after extracting the useful information. If B is charged $n(L_\ell)$, cheating is not interesting anymore because it is equivalent to establishing a session with the colluding node and communicating regularly. B is refunded $n(L_\ell)$ if and only if the operator receives a valid acknowledgement for the packet[6] P_ℓ.

The up-stream forwarding nodes (the nodes in the path between S and BS_S) are remunerated when the packet reaches BS_S and the down-stream forwarding nodes (the nodes in the path between BS_D and D) are remunerated when the operator receives the destination acknowledgement message that acknowledges the reception of the packet P_ℓ. Each forwarding node (both in the up-stream and in the down-stream) receives a credit of $\alpha(L_\ell)$. In order to motivate the destination to send an acknowledgement, the destination is also charged a small amount ε (that is refunded only if the corresponding packet is acknowledged by the destination) when BS_D injects the packet into the down-stream route. If no acknowledgement arrives, then the operator keeps the charge[7] ε.

As both $n(L_\ell)$ and $\alpha(L_\ell)$ depend on the packet size and not on the number of forwarding nodes in the path, the operator will take a loss for long routes but will make a profit from short routes. The charge $n(L_\ell)$ and the rewards $\alpha(L_\ell)$ should thus be set so that – relative to the average path length – the operator makes the desired profit on average. More details about the averaging principle for micropayments are presented in [26].

6. SECURITY

As we will see in Subsection 6.1, we provide incentives to foster collaboration between rational participants. Furthermore, we design our protocol in a way that it also provides disincentives against cheating (see Subsection 6.2).

6.1 Incentives for collaboration

The up-stream nodes benefit from forwarding a packet, since once this packet has been received by the base station (but only then), each of them will be credited for their collaboration. The down-stream nodes benefit from first forwarding a packet, and then (in the opposite direction) the acknowledgement, since each of them will be credited once the base station receives this acknowledgement. The destination benefits from sending the acknowledgement, since it will be reimbursed once this is received by the base station. While neither the down-stream nodes nor the destination will be credited if the acknowledgement does not make it to the base station, the source will still be debited the same amount (at the receipt of the packet by the base station). Therefore, it is not rational for the source to collude with the destination or any down-stream nodes to drop the acknowledgement. Down-stream nodes may collude with the destination to only send information sufficient to construct the acknowledgement (or explicitly construct this using secret information of the destination), yet this is not rational from the point of view of the destination, who supposedly derives some benefit from receiving the packet and who does not want to give out its secret key, since this allows others to impersonate it.

6.2 Disincentives against cheating

Instead of proving security against any existing combination of adversarial behaviors, we will argue that our protocol is resistant against any attack from a set of well-specified sorts. In the following, we describe primarily those attacks corresponding to a filtering adversary. This will be followed by a brief discussion on security corresponding to an invasive adversary.

[6]The packet can be lost in the way to the destination because of misbehavior (e.g., free-riding attack) or because of a broken link. As the operator cannot distinguish between the two cases, it keeps $n(L_\ell)$ if no acknowledgement is received from the destination of the packet.

[7]The packet can be, for some reason, lost in the down-stream route. However, as the operator cannot distinguish between this case and the one where D does not want to send the acknowledgment to save resources, it charges D in all cases.

Security against a refusal to pay: Each node acting as a source needs to attach a MAC of the packet sent. This MAC can only be computed with the knowledge of the secret key of the associated device. We have assumed that the base stations are trusted; therefore, they will not generate such a MAC on behalf of a node, and nobody else knows the secret key needed to generate the MAC. Therefore, the MAC uniquely identifies the initiator, who cannot claim to not have initiated the sending of the associated packet. Therefore, if a node refuses to pay, he can be disconnected from the network in the future (by having the base stations not recognize his MACs). In addition, some deposit may be rescinded. The combination of these measures makes it foolish to refuse to pay for packets that are received by the base station.

Security against incorrect reward claims: A node will be credited if it (a) appears to the base station to be part of a route during session setup, and (b) appears to be part of the route during the packet transmission. The former is determined by the base station from the MACs applied on a message associated with the session setup; each such MAC requires knowledge of the associated secret key of the node. The latter is determined by means of a correct packet (for the up-stream forwarding nodes) or acknowledgement (for the down-stream forwarding nodes). The base station will apply decryptions/encryptions corresponding to each node on the up-stream or the down-stream routes. If these are not matched by the application of the same encryption/decryption of the node in question, the packet or the acknowledgement will be incorrect. The encryptions/decryptions require knowledge of the secret keys of the associated nodes. Under the filtering assumption, these are never shared, and therefore each node in the route (as understood by the base stations) need to be involved in the routing in order (for any of them) to be credited.

Security against free-riding: Free-riding is when two parties who are not direct neighbors piggy-back information on packets they are routing, allowing the "latter" node to receive the string from the "former" one. If an adversary attaches some string to a packet in transmission, or replaces some portion of the packet with such a string, then the string will be encrypted or decrypted at least once. Since the two parties will not know the key of the party or parties performing this encryption/decryption, the string will be garbled. The only thing the attackers can do is to pass information by manipulation of the *length* of the packets; this is a channel of such low bandwidth that it is not a relevant practical threat to consider.

Security against an invasive adversary: Above, we have described how the simple cryptographic mechanisms of our scheme defend against various filtering attacks. The power of the adversary is increased in several ways by taking the step to an invasive adversary. First of all, given that the adversary may extract the secret keys of the devices he controls, each such device may then *emulate* all controlled devices. Practically speaking, this means that one device may "act" as several consecutive devices but only incurring the communication costs of a single device. If successfully performed, this would have a profound effect on how rewards are assigned. Although the invasive adversary is also assumed to be able to cause modifications of the routing tables of honest devices (which has an effect somewhat of a similar nature to that of the "emulation attack"), the underlying source routing we use in our protocol is expected to be secure and thus we do not consider "invasive routing misbehavior" in this paper.

Whereas these attacks cannot be defended against in a definitive manner, there are satisfactory statistical approaches that detect and provide evidence of such behavior and allow a better accuracy with an increased degree of cheating. We give an example of these techniques in the following.

Detecting emulated nodes: The network operator can employ statistical methods to determine if the set of concurrent neighbors is inconsistent (e.g., whether some node is claimed to be in several physical locations simultaneously). 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 in question is dishonest, which increases the likelihood that neighbors (along the routes this device is on) are, too. It is also possible to detect such misbehavior if the operator captures a rogue device containing a set of keys belonging to other participants. This is an indication that all such participants are dishonest and thus, the larger the sets of collaborating cheaters, the greater the risk for a rogue device to be captured. Finally, the operator can suspect such an attack if two or more nodes seem to be always neighbors, despite mobility. More heuristics can be found in [19].

7. OVERHEAD

In this section, we will provide an estimate of the communication and computation overheads of the solution we have described. An estimate of the size of the different fields appearing in our protocol is provided by Table 1.

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

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

NbFwdrs is the number of forwarding nodes in 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.

The fields *ReqID*, *SID* and *oldSID* are encoded on 4 bytes each to avoid the risk of using the same identifier for two different requests or sessions. The field *Route* is the concatenation of the 16 bytes identifiers (assuming e.g. an IPv6 format) of the forwarding nodes. The *TrafficInfo* field is used to inform the forwarding nodes (both in the up-stream and in the down-stream routes) about the characteristics of the traffic the initiator and the correspondent intend to exchange; using 16 bytes to encode it seems to be reasonable. Finally, we encode the sequence number on 2 bytes to support long sessions (we can have up to 2^{16} packets in a single session).

7.1 Communication Overhead

7.1.1 Session Setup Phase

According to Table 1, establishing an initiator session with $NbFwdrs$ forwarding nodes represents an overhead of $64 + NbFwdrs * 32$ bytes while establishing a correspondent session with the same number of forwarding nodes represents an overhead of $84+NbFwdrs*32$ bytes ($40+NbFwdrs*16$ bytes overhead for the requests $AReq$ and $BReq$, 20 bytes overhead for the reply $DSSRep$ and $24+NbFwdrs*16$ bytes overhead for the confirmation messages $AConf$ and $BConf$). This means that the overhead represented by the session setup is directly related to the lifetime of the sessions, which, in turn, very much depends on the stability of the routes. Hence, we have studied the stability of the routes, by means of simulations.

Description of the simulations: We consider a network composed of 100 nodes and one base station. The nodes are randomly laid out on a 500x500 m^2 single cell[8] 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 [20]. The speed is uniformly chosen between 0 and 20 m/s (note that according to [32], the average speed is lower than 10 m/s) with different values of the pause time ($PauseT$): 0, 60, 120, 300 and 600 s [4]. We discard the first 1000 seconds of simulation time to remove the initial transient phase of the random waypoint mobility model [9] and we run 100 simulations for each value of the pause time.

Figures of interest: In our simulations we are interested in the three 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 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.
- The average number of forwarding nodes ($NbFwdrs$). This number 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 average percentage of nodes that have no route to the local base station ($Noroute$). It gives an idea about the connectivity of the network.

Results: The results, given in Table 2, show that the network is highly connected ($Noroute$ is low) and that the stability of the routes decreases for a higher mobility of forwarding nodes. We consider a 95% confidence interval (CI).

If we consider pause time = 0 s, the route remains stable for an average of 8.2 s. 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]. It is possible during 8.2 s to send 410 packets, representing 65.6 kbytes of data.

[8]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.

PauseT (s)	600	300	120	60	0
Noroute	0.22%	0.06%	0.25%	0.16%	0.22%
NbFwdrs	2	1.9	1.7	1.8	1.4
AvrLT (s)	325.2	73	40.5	21.6	8.2
95% CI	15.3	21.9	15.5	14.3	6.6

Table 2: Simulation results for the different values of the pause time (Random speed between 0 and 20 m/s)

The overhead of an end-to-end session setup is 237.6 bytes (the average number of forwarding nodes is 1.4), which represents only 0.3% of the amount of information (payload) it is possible to send during the session. Moreover, as explained in Subsection 5.2, it is possible to re-establish only the broken session (the initiator session or the correspondent session) which reduces this overhead.

7.1.2 Packet Sending Phase

Considering the field sizes of Table 1, we can see that the packet sending phase represents an overhead of 22 bytes per packet (4 bytes for the session identifier, 2 bytes for the sequence number and 16 bytes for the MAC).

If the packet size is 200 bytes (same example as in paragraph 7.1.1), the overhead represents 11% of the packet size. This overhead is reduced if we use larger packets.

7.1.3 Sending the Acknowledgment

Sending the acknowledgment is an offline operation that the destination does once per session. It represents an overhead of (38+2*$NbLostPkts$) bytes per session (4 bytes for the session identifier, 16 bytes for $Batch$, 2 bytes for $LastPkt$, 16 bytes for the MAC and 2*$NbLostPkts$ for $LostPkts$). Assuming the pessimistic value of $NbLostPkts = 100$, sending the acknowledgment represents an overhead of 238 bytes per session.

7.2 Computation Overhead

In this subsection, we consider the computation overhead for the mobile nodes. The overhead is expressed in terms of number of computations and battery consumption. However, as shown in [30], 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 the initiator and the intermediate nodes to perform 2 MAC computations 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]. It is possible for us to use an even faster construction with weaker cryptographic guarantees: We can use output segments of a strong stream cipher to key a weak but extremely fast stream cipher, allowing for frequent rekeying; Thus, if an adversary

breaks the weak cipher, he can only benefit from this until the next (automatic) rekeying. Note that the use of a weak cipher only allows temporary free-riding, but not stronger attacks (e.g., impersonation). Moreover, for each packet, the source has to perform one MAC computation and the destination one MAC verification.

Acknowledgment computation: During this operation, the destination needs to perform one MAC computation and to compute the *Batch* field, which requires a XOR operation for each received packet.

Numerical example: As an example, a processor Celeron 850 MHz 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 of course would scale correspondingly.

8. CONCLUSION

In this paper, we have addressed the problem of cooperation for packet forwarding in multi-hop cellular networks. We have proposed a set of protocols that rely exclusively on symmetric cryptography techniques, and are therefore compliant with the limited resources of most mobile stations. We have studied several kinds of attacks and shown that our system is able to resist to them. A fundamental design decision consists in relying on the concept of session, which tests and exploits the relative stability of routes. We have quantified the life time of these sessions and we have shown that they help keeping the overhead moderate. We have also proven that the usage of our charging and rewarding scheme indeed stimulates cooperation in multi-hop cellular networks.

In terms of future work, we intend to explore further the invasive attacks and to consider routing misbehavior. We will explore techniques aiming at the calibration of the relevant parameters, and extend our protocols to include the case in which the correspondent is charged. We will also consider the use of session keys in order to reduce the communication overhead between the base stations and the key repository. Finally, we will extend the charging and rewarding principle and make it possible to remunerate the intermediate nodes even for the packets that did not reach their target.

9. ACKNOWLEDGMENTS

The authors would like to thank Dirk Westhoff for his useful feedback, Catherine Boutremans for a helpful discussion on Voice over IP, and Mario Čagalj, Márk Félegyházi and Jun Luo for helpful discussions and comments. Thanks also to the anonymous reviewers for their useful feedback and for bringing reference [32] to our attention.

10. 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] 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.

[4] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva. A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols. In *Proceedings of MobiCom*, pages 85–97, 1998.

[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*, June 2002.

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

[7] L. Buttyán and J.-P. Hubaux. Report on a Working Session on Security in Wireless Ad Hoc Networks. *ACM Mobile Computing and Communications Review (MC2R)*, October 2002.

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

[11] V. Gupta, S. Krishnamurthy, and M. Faloutsos. Denial of Service Attacks at the MAC Layer in Wireless Ad Hoc Networks. In *Proceedings of Milcom*, 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, 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.

[17] 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.

[18] 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.

[19] 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.

[20] D. B. Johnson and D. A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In *Mobile Computing* edited by Tomasz Imielinski and Hank Korth, chapter 5, pages 153181. Kluwer Academic Publishers, 1996.

[21] 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.

[22] 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.

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

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

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

[26] S. Micali and R. L. Rivest. Micropayments Revisited. In *Proceedings of the Cryptographer's Track at the RSA Conference 2002*, pages 149–163, 2002.

[27] 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.

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

[29] 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.

[30] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. D. Tygar. SPINS: Security Protocols for Sensor Networks. In *Proceedings of MobiCom*, 2001.

[31] 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.

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

[33] 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.

[34] 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.