FESCIM: Fair, Efficient, and Secure Cooperation Incentive Mechanism for Multihop Cellular Networks

Mohamed M.E.A. Mahmoud, *Member*, *IEEE*, and Xuemin (Sherman) Shen, *Fellow*, *IEEE*

Abstract—In multihop cellular networks, the mobile nodes usually relay others' packets for enhancing the network performance and deployment. However, selfish nodes usually do not cooperate but make use of the cooperative nodes to relay their packets, which has a negative effect on the network fairness and performance. In this paper, we propose a fair and efficient incentive mechanism to stimulate the node cooperation. Our mechanism applies a fair charging policy by charging the source and destination nodes when both of them benefit from the communication. To implement this charging policy efficiently, hashing operations are used in the ACK packets to reduce the number of public-key-cryptography operations. Moreover, reducing the overhead of the payment checks is essential for the efficient implementation of the incentive mechanism due to the large number of payment transactions. Instead of generating a check per message, a small-size check can be generated per route, and a check submission scheme is proposed to reduce the number of submitted checks and protect against collusion attacks. Extensive analysis and simulations demonstrate that our mechanism can secure the payment and significantly reduce the checks' overhead, and the fair charging policy can be implemented almost computationally free by using hashing operations.

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

1 Introduction

Multihop cellular network (MCN) [1], [2], [3], [4] is a network architecture that incorporates the ad hoc characteristics into the cellular system. A node's traffic is usually relayed through other nodes to the destination. The network nodes commit bandwidth, data storage, CPU cycles, battery power, etc., forming a pool of resources that can be shared by all of them. The utility that the nodes can obtain from the pooled resources is much higher than that they can obtain on their own. The considered MCN is used for civilian applications where the network has long life and the mobile nodes are supposed to have long-term relations with the network. Multihop packet relay can reduce the dead areas by extending the communication range of the base stations without additional costs. It can also reduce the energy consumption because packets are transmitted over shorter distances, and improve the area spectral efficiency and the network throughput and capacity [5], [6], [7]. However, due to involving autonomous devices in packet relay, the packet routing process suffers from new security challenges that endanger the practical implementation of MCN.

The assumption that the network nodes are willing to relay other nodes' packets may not hold for civilian applications where the nodes are autonomous and self-interested in the sense that they aim to maximize their welfare and minimize their contributions. Although the proper network operation requires the nodes to collaborate, collaboration consumes the nodes' resources such as energy and computing power, and does not provide direct benefits. Selfish nodes are not interested in cooperation without incentive and make use of the cooperative nodes to relay their packets, which has a negative effect on the network fairness and performance. A fairness issue arises when selfish nodes take advantage of the cooperative nodes without any contribution to them. The selfish behavior also significantly degrades the network performance, which may result in failure of the multihop communications [8], [9].

Reputation-based and incentive mechanisms [10], [11] have been proposed to thwart selfishness attacks. For reputation-based mechanisms [12], [13], [14], the nodes usually monitor the transmissions of their neighbors to make sure that the neighbors relay other nodes' traffic, and thus, selfish nodes can be identified and punished. For incentive mechanisms, packet relay is a service not an obligation. The source and destination nodes pay credits (or virtual currency) to the intermediate nodes for relaying their packets. Credits can stimulate the nodes' cooperation by proving that it is more beneficial for the nodes to cooperate than behaving selfishly.

Reputation-based mechanisms suffer from essential problems that discourage implementing them in MCN. First, monitoring the nodes' transmissions by overhearing the channel is not energy efficient for transmitters. The full-power transmission is used instead of adapting the transmission power according to the distance separating the transmitter and the receiver [15] to enable more neighboring nodes to hear the packet transmission. Second, reputation-based mechanisms do not achieve fairness because the highly contributing nodes are not compensated. For example, although the nodes at the network center relay

• *The authors are with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada. E-mail: {mmabdels, xshen}@bbcr.uwaterloo.ca.*

Manuscript received 23 May 2010; revised 20 Feb. 2011; accepted 12 Mar. 2011; published online 28 Apr. 2011.
For information on obtaining reprints of this article, please send e-mail to: tmc@computer.org, and reference IEEECS Log Number TMC-2010-05-0247. Digital Object Identifier no. 10.1109/TMC.2011.92.

more packets than those at the periphery, they are not compensated. Third, the mechanisms suffer from unreliable detection of the selfish nodes and false accusation of the honest nodes. That is because it is difficult to differentiate between a node's unwillingness and incapability to cooperate, e.g., due to low resources, packet collision, and network congestion.

In addition to cooperation stimulation, the incentive mechanisms can achieve fairness by charging or rewarding credits to balance a node's contribution and its benefit. The node's contribution can be relaying other nodes' packets or paying credits, whereas the node's benefit can be relaying its packets or earning credits. In other words, credits are used to compensate the nodes for relaying other nodes' packets to enforce fairness. Moreover, incentive mechanisms can discourage launching *Resource-Exhaustion* attack by sending bogus messages to exhaust the intermediate nodes' resources because the nodes pay for relaying their packets. Incentive mechanisms can also efficiently bill the network users when they roam among different foreign networks by paying all the parties involved in the communication without contacting distant home location registers [16].

However, the efficient implementation of the existing incentive mechanisms is questionable because they impose significant overhead. First, the fair charging policy is to charge both the source and destination nodes when both of them benefit from the communication. To securely implement this charging policy, two signatures are usually required per message (one from the source node and the other from the destination node) to prevent payment repudiation and manipulation. Nevertheless, the extensive use of the public key cryptography is very costly, which degrades the network performance and stimulates the nodes to behave selfishly. Second, since a trusted party may not be involved in the communication sessions, the nodes usually compose undeniable proof of packet relaying called check, and submit the checks to a central unit called the accounting center (AC) for clearance. However, submitting and processing a large number of checks implies significant communication and computation overhead and implementation complexity.

In this paper, we propose **FESCIM**, a **F**air, **E**fficient, and **S**ecure **C**ooperation **I**ncentive **M**echanism, to stimulate the node cooperation in MCN. In order to efficiently and securely charge the source and destination nodes, the lightweight hashing operations are used in the *ACK* packets to reduce the number of public-key-cryptography operations. The destination node generates a hash chain and signs its root, and acknowledges message reception by releasing a hash value from the hash chain. In this way, the destination node generates a signature per group of messages instead of generating a signature per message.

Furthermore, instead of generating a check per message or generating a nodal check for each intermediate node, a small-size check containing the payment data for all the intermediate nodes is generated per route. In addition, trusting one node to submit the check is not secure because this node may collude with the source and destination nodes to not submit the check, as we will discuss in Section 4.2. Instead of submitting the checks by all the intermediate nodes to thwart collusion attack, a *Probabilistic-Check-Submission* scheme is proposed to reduce the number of submitted checks and protect against collusion attack. In Section 5, we will show that if each intermediate node submits a low ratio of randomly chosen checks, most of the checks can be probabilistically submitted under collusion attack. Extensive analysis and simulations demonstrate that FESCIM can secure the payment, charge the source and destination nodes almost computationally free, and significantly reduce the number of generated and submitted checks.

The remainder of this paper is organized as follows: Section 2 reviews the related work and Section 3 presents the system models. The proposed cooperation incentive mechanism is presented in Section 4. Security analysis and performance evaluation are provided in Sections 5 and 6, respectively, followed by conclusion and future work in Section 7.

2 Related Work

In tamper-proof device (TPD)-based incentive mechanisms [17], [18], [19], [20], [21], a TPD is installed in each node to manage its credit account and secure its operation. In Nuglets [17], [18], the self-generated and forwarding packets are passed to the TPD to decrease and increase the node's credit account, respectively. The packet purse and the packet trade models have been proposed. In the packet purse model, only the source node pays by loading some credits in each packet before sending it. Each intermediate node acquires the amount of credits that cover the packet's relaying cost. In the packet trade model, each intermediate node runs an auction to sell the packets to the following node in the route. In this way, each intermediate node earns some credits and the destination node pays the total packet relaying cost. In SIP [19], after receiving a data packet, the destination node sends a payment RECEIPT packet to the source node to issue a REWARD packet to increment the credit accounts of the intermediate nodes. In CASHnet [20], [21], the source node's traffic credit account is charged and a signature is attached for each data packet. Upon receiving the packet, the destination node's traffic credit account is also charged and a digitally signed *ACK* packet is sent back to increase the helper credit accounts of the intermediate nodes. Users regularly visit service points to buy traffic credits with real money and/or transfer helper credits to traffic credits.

The TPD-based incentive mechanisms suffer from the following problems. First, the assumption that the TPD cannot be tampered is neither secure nor practical for MCNs. That is because the nodes are autonomous and self-interested and the attackers can communicate freely in an undetectable way if they could compromise the TPDs [22]. Moreover, since the security protection of these mechanisms completely fails if the TPDs are tampered, only a small number of manufacturers can be trusted to make the network nodes, which is too restrictive for civilian networks. Second, a node cannot communicate if it does not have sufficient credits at the communication time. The nodes at the network edge cannot earn as many credits as the nodes at other locations because they are less frequently selected by the routing protocol. Furthermore, the credit distribution has direct impact on the network performance, e.g., if a small number of nodes have a large ratio of the network credits, the network performance significantly

degrades because the rich nodes are not motivated to cooperate and the poor nodes cannot initiate communications. Finally, since credits are cleared in real time, the network performance degrades if the network does not have enough credits circulating around. In [23], it is shown that the overall credits in the network decline gradually because the total charges are not necessarily equal to the total rewards. That is because the source node is fully charged after sending a packet but some intermediate nodes may not be rewarded when the route is broken. Although CASHnet can alleviate this problem by buying credits with real money, it is shown in [23] that in spite of having helper credits, some nodes starve because they cannot find a service point to convert them to traffic credits.

In order to eliminate the need for TPDs, a central bank called the AC can be used to store and manage the nodes' accounts. In [24], the source node appends a payment token to each transmitted packet, and each intermediate node uses its secret key to check whether the token corresponds to a winning ticket. Winning tickets are submitted to the AC to reward the winning nodes. The source and destination nodes are charged per packet but the intermediate nodes are rewarded per winning ticket. In a security flaw, the colluding nodes can exchange tokens to be checked in each node to steal credits. In our earlier work [25], instead of submitting payment checks to the AC, each node submits an activity report containing its alleged charges and rewards of different sessions. The AC uses a reputation system to identify the cheating nodes that report false charges and/or rewards to steal credits. However, due to the nature of the reputation systems, some honest nodes may be falsely identified as cheaters and the colluding nodes may manage to steal credits.

In [26], each node in a route buys packets from the previous node and sells them to the next node. The packets' buyers contact the AC to get deposited coins and the packets' sellers submit the coins to the AC to claim their payment. However, the interactive involvement of the AC in each communication session is not efficient and creates a bottleneck at the AC. In [27], an incentive mechanism has been proposed for MCN. Unlike the original MCN architecture proposed in [1], the base stations are involved in every communication session, which may lead to suboptimal routes when the source and destination nodes reside in the same cell. In addition, the corrupted messages are relayed to the base station before they are dropped because the intermediate nodes cannot verify the authenticity and the integrity of the messages.

In Sprite [28], the source node signs the identities of the nodes in the route and appends its signature to each transmitted message. The intermediate and the destination nodes compose checks and submit them to the AC to claim the payment. In Express [29], the source node generates a hash chain for each intermediate node $\mathbb{ID}_K$ and commits to the hash chain by digitally signing the root of the hash chain and sending the signature to $\mathbb{ID}_K$. Each time the node $\mathbb{ID}_K$ relays a message, the source node releases the preimage of the last sent hash value. The source, intermediate, and destination nodes compose checks and submit them to the AC. However, the nodes have to generate and store a large number of hash chains because any node in the network may act as an intermediate node due to the node mobility. The packet overhead is large especially if the number of intermediate nodes is large because the source node attaches one hash value for each intermediate node. In Sprite and Express, only the source node pays no matter how the destination node benefits from the communication. Moreover, since the intermediate nodes are rewarded for the relayed messages that do not reach the destination, all the nodes in a route submit the checks because packet relay is considered successful by a node if a next node in the route submits a valid check. We call this check submission scheme *All-Submitters* because all the intermediate nodes submit all the checks. In Sprite and Express, significant communication and computation overhead is implied due to generating and submitting a large number of checks because a check is generated per message and all the nodes in a route submit all the checks.

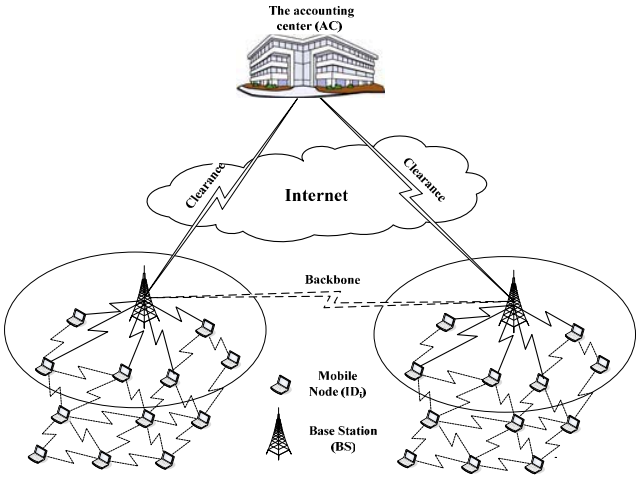

Fig. 1. The architecture of the multihop cellular network.

In [30], an incentive mechanism has been proposed for ad hoc networks that are used to connect the nodes to the Internet. The source node signs the identities of the nodes in the route and appends this signature to the message, and the destination node signs a check and sends it to the last intermediate node to submit to the AC. Since only the last intermediate node submits the check, we call this check submission scheme *Fixed-Submitter*. However, the source and destination nodes can communicate freely and the intermediate nodes are not rewarded if the last intermediate node colludes with the source and destination nodes to not submit the checks. Moreover, generating two signatures per message is too costly for mobile nodes. In [31], a hash chain is used to efficiently authenticate digital streams but FESCIM uses a hash chain to achieve payment nonrepudiation to secure the payment.

3 System Models

3.1 Network and Communication Models

As shown in Fig. 1, the considered MCN includes an accounting center, a set of base stations, and mobile nodes. The AC stores and manages the credit accounts of the nodes, and generates private/public key pair and certificate with unique identity for each node. Once the AC receives a check, it updates the accounts of the participating nodes. The base stations are connected with each other and with the AC by a backbone network that may be wired or

wireless. FESCIM can be implemented on the top of any routing protocol, such as DSR [32] and AODV [33], to establish an end-to-end communication session provided that the full identities of the nodes in the route are known to the source and destination nodes. It is important to include these identities in the source and the destination node's signatures to compose valid checks.

All communications are unicast and the nodes can communicate in one of two modes: pure ad hoc or hybrid. For pure ad hoc mode, the source and destination nodes communicate without involving base stations. The source node's messages may be relayed in several hops by the intermediate nodes to the destination node. For hybrid mode, at least one base station is involved in the communication. The source node transmits its messages to the source base station (BS_S), if necessary in multiple hops. If the destination node resides in a different cell, the messages are forwarded to the destination base station (BS_D) that transmits the messages to the destination node possibly in multiple hops. The nodes can contact the AC at least once every few days. This connection can occur via the base stations or the wired networks such as the Internet. During this connection, the nodes submit checks, renew their certificates, and convert credits to real money and/or purchase credits with real money.

3.2 Threat and Trust Models

Since the mobile nodes are autonomous and self-interested, the attacker has full control on his node, and thus he can change its operation. The attackers work individually or collude with each other under the control of one authority to launch sophisticated attacks. The attackers are rational in the sense that they misbehave when they can achieve more benefits than behaving honestly. Specifically, the attackers attempt to steal credits, pay less, and communicate freely. The mobile nodes are probable attackers because they are motivated to misbehave to increase their welfare, but the AC is fully secure. It is impossible to realize secure payment between two entities without trusted third party [34]. For the trust models, the network nodes fully trust the AC to correctly perform billing and auditing, but the AC does not trust any node or base station in the network. The network nodes also trust their operators' base stations but do not trust those of other operators.

3.3 Payment Model

A fair charging policy is to support cost sharing between the source and destination nodes when both of them benefit from the communication. In order to make FESCIM flexible, the payment-splitting ratio is adjustable and service dependent, e.g., a DNS server should not pay for name resolution. For rewarding policy, some incentive mechanisms, such as [35], consider that a packet relaying reward is proportional to the incurred energy in relaying the packet. It is difficult to implement this rewarding policy in practice without involving complicated route discovery process and calculation of enroute individual payments. Therefore, similar to [19], [25], [28], [29], we use fixed rewarding rate, e.g., λ credits per unit-sized packet.

In MCNs, packet loss may occur normally due to node mobility, channel impairment, etc. Ideally, any node that has ever tried to relay a packet should be rewarded no matter whether the packet eventually reaches its destination or not because relaying a packet consumes the node's resources. However, it is difficult to corroborate an intermediate forwarding action without involving too much overhead, e.g., all the intermediate nodes have to submit all the checks in [28], [29]. Moreover, rewarding the nodes for relaying route establishment packets or packet retransmissions significantly increases the number of checks because a large number of nodes may relay route establishment packets and packet retransmission frequently happens in wireless networks. Therefore, the AC charges the source and destination nodes for every transmitted message even if the message does not reach the destination, but the AC rewards the intermediate nodes only for the delivered messages. For fair rewarding policy, the value of λ is determined to compensate the nodes for relaying route-establishment packets, packet retransmission, and undelivered packets. In Section 5, we will argue that our charging and rewarding policies can thwart rational attacks and encourage the nodes' cooperation. Similar to the VISA system and the incentive mechanisms in [25], [28], [29], [30], the nodes communicate first and pay later. The AC issues certificates to enable the nodes to transact by issuing digital checks without the need for direct verification from the AC to avoid frequently contacting the AC and thus creating a bottleneck at the AC.

The nodes at the network border cannot earn as many credits as those at other locations because they are less frequently selected by the routing protocol. In order to communicate, they can purchase credits with real money. However, we do not consider this as a fairness problem because the philosophy behind incentive mechanisms is that packet relay is a service not an obligation. This service may not be requested from some nodes, i.e., the customers (source and destination nodes) request the packet-relay service from the best service providers (shortest route nodes). If the traffic is directed through the border nodes, obviously, we sacrifice the network performance because the routes may be long. Due to the node mobility, the border nodes can change their location and earn more credits as shown in [19]. Moreover, the border nodes do not relay as many packets as others, and thus, it is fair to charge the border nodes real money to compensate the other nodes that relayed more packets. Table 1 gives the used notations in this paper.

4 The Proposed FESCIM

In this section, we first present FESCIM for hybrid mode and then for pure ad hoc mode.

4.1 Hybrid Mode

4.1.1 Route Discovery Phase

In order to establish an end-to-end route, the source node broadcasts the *Route Request Packet* (*RREQ*) containing the identities of the source (ID_S) and the destination (ID_D) nodes, the route establishment time stamp (T_S), and the payment-splitting ratio (P_r). The source node is charged the ratio of P_r of the total payment and the destination node is charged the ratio of $1-P_r$. A network node appends its identity and broadcasts the packet if the time stamp is within a proper range. The *RREQ* packet is relayed by BS_S to BS_D (if the destination node resides in a different base

TABLE 1
The Useful Notations

Symbol	Description
A, B	A is concatenated to B.
C	The number of colluding nodes in a route.
ID_k	The identity of the intermediate node k, or node with identity ID_k.
ID_S and ID_D	The identities of the source and destination nodes, respectively.
H(M)	The hash value resulted from hashing M.
$H_D^X(i)$	The hash value number X in the *i*th hash chain used in a route.
M_X	The message sent in the *X*th data packet in a session.
n	The number of intermediate nodes in a session.
P_r	The payment-splitting ratio paid by the source node and the ratio (1-P_r) of the payment is paid by the destination node.
R_C	The ratio of cheques submitted by an intermediate node.
S_i	The session unique identifier.
$Sig_S(M)$ and $Sig_D(M)$	The signatures of the source and destination nodes on M, respectively.
T_S	A session establishment time stamp.
Z+1	Hash chain size.

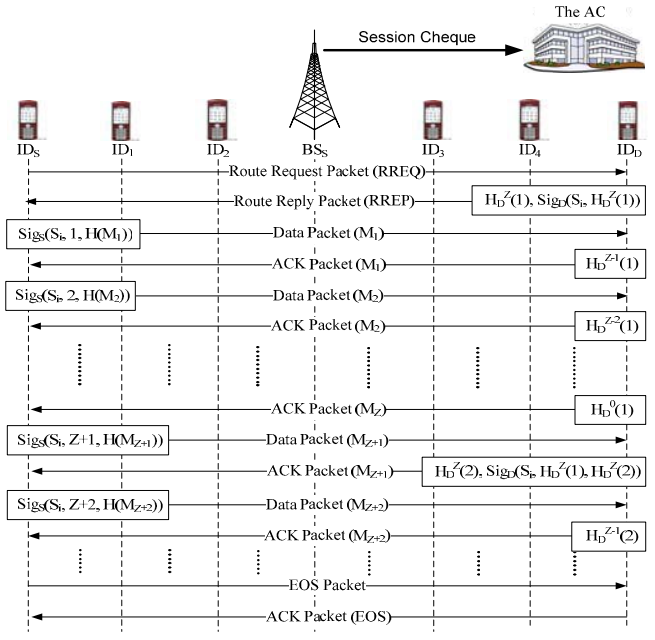

Fig. 3. The exchanged security tags in a session.

station) that broadcasts it. Finally, the destination node sends back the *Route Reply Packet* (*RREP*) to establish the route. The source node initiates a new route discovery phase if the route is broken.

As shown in Fig. 2, the destination node generates a hash chain with size of $Z+1$ by iteratively hashing a random value called seed ($H_D^0(i)$) Z times to obtain a final hash value called root ($H_D^Z(i)$), where $H_D^X(i) = H(H_D^{X-1}(i))$ and i is the hash-chain number. Note that multiple hash chains may be used in one route. From Fig. 3, the *RREP* packet contains the session identifier (S_i), the destination node's certificate, the root of the first hash chain ($H_D^Z(1)$), and the destination node's signature ($Sig_D(S_i, H_D^Z(1))$). S_i contains the identities of the nodes in the route, T_S, and P_r, e.g., $S_i = ID_S, ID_1, ID_2, BS_S, ID_3, ID_4, ID_D, T_S, P_r$ for the route shown in Fig. 3. The destination node's signature authenticates the node and proves its approval to pay for the session. The signature also proves that the hash chain has indeed been created by the destination node and links it to the route. Upon receiving the *RREP* packet, each intermediate node relays the packet if the signature is correctly verified, and the source node starts data transmission.

4.1.2 Data Generation and Relay Phase

For the *X*th data packet, Fig. 3 shows that the source node appends the message M_X and its signature ($Sig_S(S_i, X, H(M_X))$). Signing the hash of the message instead of the message can significantly reduce the check size because the smaller size $H(M_X)$ and not M_X is attached to the check. The source node attaches its certificate to the first data packet to enable the intermediate and destination nodes to verify its signatures. Before relaying a data packet, each intermediate node verifies the attached signature to ensure the message's integrity and authenticity and to verify the payment data that include S_i and X. The intermediate nodes store only the last source node's signature to be used in the check composition, i.e., after receiving the *X*th data packet, the intermediate nodes delete $Sig_S(S_i, X-1, H(M_{X-1}))$ and store $Sig_S(S_i, X, H(M_X))$ that is enough to prove that *X* messages have been transmitted and $X-1$ messages have been delivered. Each node in the route restarts a timer each time the node transmits or relays a packet. The route is considered broken when the timer expires. After receiving the *ACK* of the last message, the source node sends *End of Session* (*EoS*) packet to close the session.

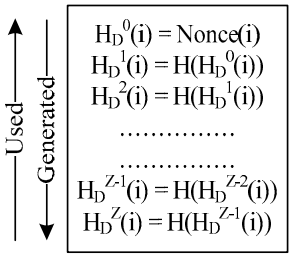

Fig. 2. Hash chain generated by the destination node.

4.1.3 ACK Generation and Relay Phase

Upon receiving the *X*th data packet and $X \leq Z$, the destination node sends back *ACK* packet containing the preimage of the last released hash value, or $H_D^{Z-X}(1)$, to acknowledge receiving the message in an undeniable way. Therefore, instead of generating a signature per *ACK* packet, one signature is generated per Z *ACK*s. Payment nonrepudiation and nonmanipulation are achievable because the hash function is one-way, i.e., only the destination node could have generated the hash chain because it is not possible to compute $H_D^{Z-X-1}(1)$ from $H_D^{Z-X}(1)$. Each intermediate node verifies the hash value by making sure that $H_D^{Z-X}(1)$ is generated from hashing $H_D^{Z-X-1}(1)$. As illustrated in Fig. 3, after releasing all the hash values of the first hash chain, the destination node creates a new hash chain and authenticates it by signing all the used hash chains' roots, or $Sig_D(S_i, H_D^Z(1), H_D^Z(2))$, instead of signing only the last hash chain's root. In this way, the intermediate nodes store only

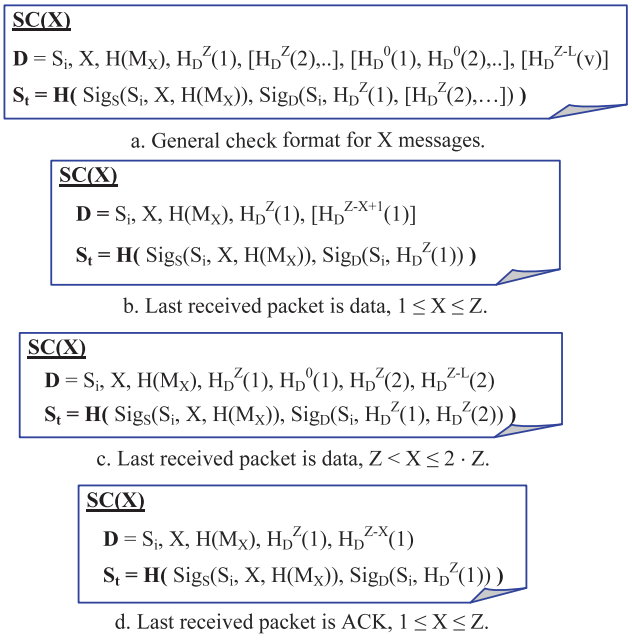

Fig. 4. The formats of the payment checks.

the last destination node's signature for the check composition because it can authenticate all the used hash chains in the route. The intermediate nodes also store the hash chains' roots and seeds and the last released hash value for the check composition.

4.1.4 Check Composition Phase

For each route, one check containing the payment data for all the intermediate nodes can be composed. The general format of the route check is shown in Fig. 4a, where "[Y]" means that Y may not exist in some cases. A check contains two main parts: *Descriptor (D)* and *Security Token* (S_t). The *Descriptor* contains S_i that has the identities of the payers and the payees, T_S, and P_r. The *Descriptor* also contains the messages' number (X), the hash value of the last received message, the hash chains' roots and seeds, and the last released hash value ($[H_D^{Z-L}(v)]$). The *Security Token* is an undeniable proof that prevents payment repudiation and manipulation, and thus ensures that the check is undeniable, unmodifiable, and unforgeable. In order to significantly reduce the check size, the *Security Token* is composed by hashing the source and destination nodes' signatures instead of attaching the large-size signatures. The check size depends on the number of used hash chains because two hash values should be attached for each hash chain, and thus properly choosing the hash chain size can minimize the check size.

Fig. 4b shows the composed check when the route is broken during relaying the Xth data packet and $1 \leq X \leq Z$, i.e., only one hash chain is used. If the route is broken after receiving the first data packet ($X = 1$), the check does not have $H_D^{Z-X+1}(1)$ because the *ACK* packet is not received. However, for $1 < X \leq Z$, the last hash value received from the destination node ($H_D^{Z-X+1}(1)$) is attached to the check. Fig. 4c shows the composed check when the route is broken after receiving the Xth data packet and $Z < X \leq 2 \cdot Z$, i.e., two hash chains are used. It can be seen that the check contains the seed and the root of the first hash chain, the root of the second hash chain, and the last released hash value ($H_D^{Z-L-1}(2)$), where $X = Z + L + 1$. Moreover, the two hash chains' roots are included in the destination node's signature. Fig. 4d shows the composed check when the last received packet is the *ACK* of the Xth message.

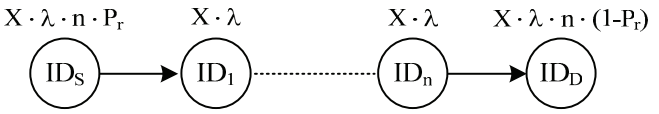

Fig. 5. The charges and rewards for X messages.

4.1.5 Check Clearance Phase

The base station submits the check to the AC for redemption, but the nodes submit the check if the base station belongs to a different operator. The nodes also submit the check if the route is not complete, i.e., the *EOS* packet is not received, and the base station does not have correct payment information. For example, if the route is broken during relaying the *ACK* of M_X from ID_D to BS_D, the BS_D's check does not prove that the Xth message is delivered, and thus, the nodes are not rewarded for the last message if they do not submit the check. Once the AC receives a check, it checks that the check has not been deposited before using its unique identifier (S_i). Then, the AC generates the source and destination nodes' signatures and hashes them to verify the check's credibility. The check is valid if the resultant hash value is identical to the check's *Security Token*. For a check (SC(X)), the number of transmitted messages (X) is signed by the source node, and the number of delivered messages can be computed from the number of hashing operations to map $H_D^{Z-X}(1)$ to $H_D^Z(1)$. Finally, the AC clears the check according to the charging and rewarding policy discussed in Section 3.3.

4.2 Pure Ad Hoc Mode

The proposed mechanism for hybrid mode can be used for pure ad hoc mode, but the intermediate nodes have to submit the checks to the AC because the base stations are not involved in the communication. A check contains payment data for all the nodes in the route, but it is not secure to trust one node to submit the check because it may collude with the source and destination nodes so as not to submit the check to increase their welfare. Fig. 5 shows the charges and rewards for sending X messages in a route with n intermediate nodes. If the source and destination nodes collude with K intermediate nodes and the check is not submitted, the colluders can save $X \cdot \lambda \cdot (n - K)$ credits. Obviously, the colluders can achieve gains when $K < n$, and thus, the source and destination nodes can compensate the colluding intermediate nodes. On the other hand, it is not efficient to submit a check by each intermediate node [28], [29] due to significantly increasing the number of redundant checks. In this section, we propose two schemes for efficiently thwarting the collusion attacks against check submission.

4.2.1 Changeable-Submitter Check Submission Scheme

Similar to [30], one node submits the check, but the position of the check submitter is randomly changed instead of fixed. All the intermediate nodes execute a public function

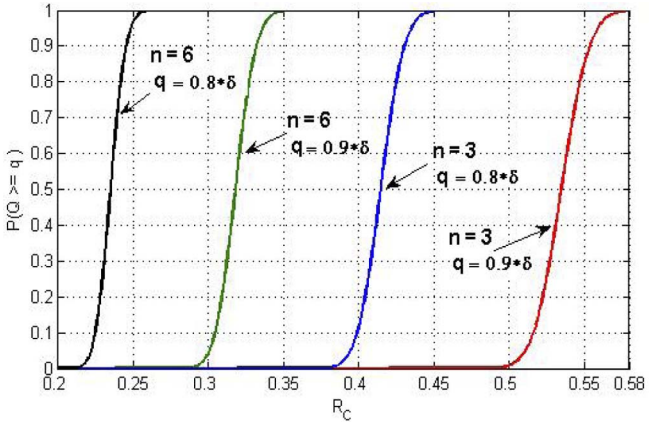

Fig. 6. The effect of $\mathrm{R_C}$ on the number of submitted checks.

called *Selector* to decide the check submitter. The *Selector*'s input is the session identifier and the return value is the position of the check submitter, e.g., if the return value is 1, the first intermediate node after the source node is the check submitter and so on. Obviously, changing the function's input can change the position of the check submitter, which increases the difficulty of the collusion attack. The *Selector* can be implemented using a hash function with deriving the return value from the output hash value such that all possible cases are equally likely, i.e., all the intermediate nodes have the same probability to be the check submitter.

4.2.2 Probabilistic-Check-Submission Scheme

Instead of submitting all the checks by each intermediate node [28], [29], each node submits the ratio of $\mathrm{R_C}$ of randomly chosen checks such that a minimum ratio of the unrepeated checks will be submitted probabilistically. Equations (1) and (2) give the probability that at least q unrepeated checks out of δ are submitted for routes with n intermediate nodes and C colluding nodes, where Q denotes an integer random variable $(0 \leq \mathrm{Q} \leq \delta)$ that represents the number of unrepeated checks that are submitted:

$$\mathrm{P}(\mathrm{Q} \geq \mathrm{q}) = \sum_{\mathrm{j=q}}^{\delta} \mathrm{P}(\mathrm{Q} = \mathrm{j}) \tag{1}$$

$$\mathrm{P}(\mathrm{Q} = \mathrm{j}) = \binom{\delta}{\mathrm{j}} \cdot (1 - (1 - \mathrm{R_c})^{\mathrm{n-C}})^{\mathrm{j}} \cdot ((1 - \mathrm{R_c})^{\mathrm{n-C}})^{\delta - \mathrm{j}}. \tag{2}$$

Fig. 6 shows the relation between $\mathrm{P}(\mathrm{Q} \geq \mathrm{q})$ and $\mathrm{R_C}$ at different values of q and n assuming that δ is 1,000 checks and all the nodes do not collude $(\mathrm{C} = 0)$. *It can be seen that most of the checks can be submitted with a very high probability when R_C is much less than 1*. For example, when n equals to 6, the probabilities of submitting at least 80 percent and 90 percent of the checks are 1 if each node submits only 27 percent and 34 percent of the checks, respectively. However, when n equals to 3, $\mathrm{R_C}$ should be 0.44 and 0.57 to submit at least 80 percent and 90 percent of the checks with the probability of 1, respectively, i.e., $\mathrm{R_C}$ should be larger for shorter routes to guarantee submitting the same ratio of the checks. $\mathrm{R_C}$ should be determined to achieve a specific probability of submitting a minimum ratio of the checks. In this paper, we assume that $\mathrm{R_C}$ is chosen to make

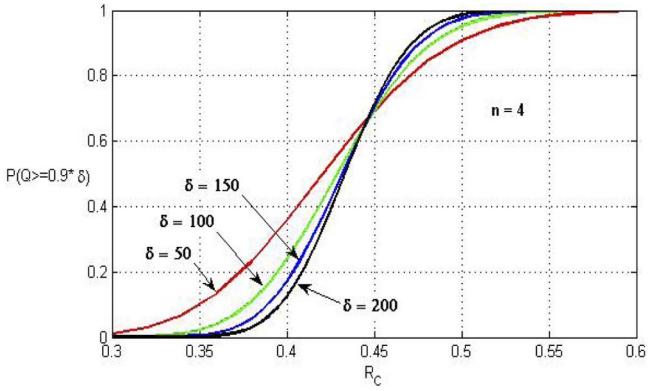

Fig. 7. The effect of δ on $\mathrm{P}(\mathrm{Q} \geq 0.9 \cdot \delta)$.

$\mathrm{P}(\mathrm{Q} \geq \mathrm{q}) = 0.9$. The minimum $\mathrm{R_C}$ that achieves this probability is denoted by $\mathrm{R_C}^*$, e.g., from Fig. 6, $\mathrm{R_C}^*$ is 0.56 and 0.33 when n equals 3 and 6, respectively. The rationale here is that a little increase in $\mathrm{R_C}$ significantly increases the number of redundant checks in the long run, but the increase of $\mathrm{R_C}$ above $\mathrm{R_C}^*$ has little effect on the check submission probability. From Fig. 6, when n equals 3, the increase of $\mathrm{R_C}$ from 0.52 to 0.55 and from 0.55 to 0.58 changes $\mathrm{P}(\mathrm{Q} \geq 0.9 \cdot \delta)$ with 0.75 and 0.15, respectively.

Fig. 7 shows the effect of δ on the probability of clearing at least 90 percent of the checks. The figure demonstrates that properly choosing the value of $\mathrm{R_C}$, e.g., between 0.5 and 0.55, can make δ have very little effect on $\mathrm{P}(\mathrm{Q} \geq 0.9 \cdot \delta)$. Fig. 8 shows the effect of the collusion attack on $\mathrm{P}(\mathrm{Q} \geq 0.9 \cdot \delta)$ at different values of $\mathrm{R_C}$ and n of 5. The figure demonstrates how the honest nodes can take protective measures to protect against the colluding nodes. To mitigate the effect of one and two colluding nodes, $\mathrm{R_C}^*$ should be 0.46 and 0.56, respectively, i.e., if each node submits only 56 percent of the checks, we can probabilistically guarantee submitting at least 90 percent of the checks even if two nodes collude. Therefore, *the increase of R_C strengthens the robustness against the collusion attack but with additional redundant checks, which shows an intuitive trade-off between security and overhead*.

4.3 Discussion

Although our focus was on the unidirectional data transmission in the previous sections, FESCIM can also be applied for bidirectional data transmission. In this case, FESCIM can bypass sending some *ACK* packets by using the data packets

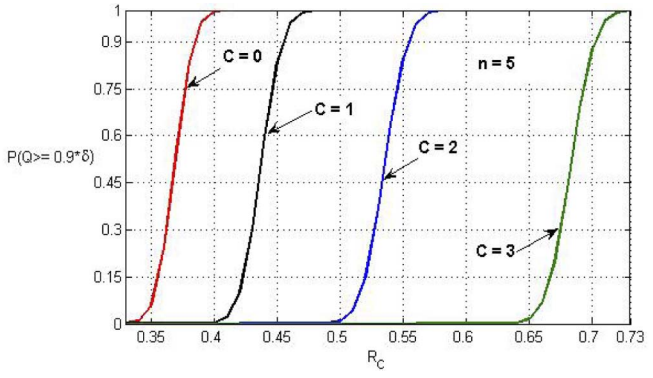

Fig. 8. The effect of collusion attack on the ratio of submitted checks.

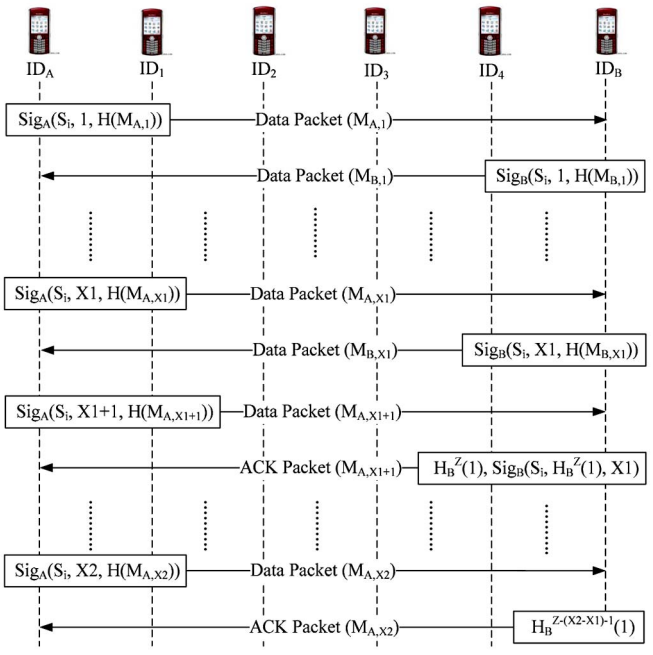

Fig. 9. The security tags of the bidirectional data transmission.

to serve as *ACK*s. Fig. 9 shows the exchanged security tags in FESCIM for bidirectional data transmission assuming that the nodes $\mathbb{ID}_A$ and $\mathbb{ID}_B$ send X2 and X1 messages, respectively, and X2 > X1. The Xth data packet sent from $\mathbb{ID}_A$ to $\mathbb{ID}_B$ contains $\mathrm{S_i}$, X, the message $\mathrm{M_{A,X}}$ and its hash value, and the signature $\mathrm{Sig_A(S_i, X, H(M_{A,X}))}$. The Xth data packet sent from $\mathbb{ID}_B$ to $\mathbb{ID}_A$ contains $\mathrm{S_i}$, X, the message $\mathrm{M_{B,X}}$ and its hash value, and the signature $\mathrm{Sig_B(S_i, X, H(M_{B,X}))}$. The security tag $\mathrm{Sig_B(S_i, X, H(M_{B,X}))}$ means that node $\mathbb{ID}_B$ has sent X messages to $\mathbb{ID}_A$ and has also received X messages from $\mathbb{ID}_A$. After $\mathbb{ID}_B$ sends all its messages, it switches to the hash-chain-based *ACK*s to enable $\mathbb{ID}_A$ to complete sending its messages. The first *ACK* packet contains $\mathbb{ID}_B$'s signature for a hash chain root $(\mathrm{H_B^Z(1)})$ and X1. Fig. 10a shows the check format of the bidirectional data transmission.

FESCIM can also be used for asymmetric paths. As shown in Fig. 11, the path P1 with the intermediate nodes $\mathbb{ID}_1$ and $\mathbb{ID}_2$ can only transport packets from $\mathbb{ID}_S$ to $\mathbb{ID}_D$, and the path P2 with the intermediate nodes $\mathbb{ID}_3$ and $\mathbb{ID}_4$ can only transport packets from $\mathbb{ID}_D$ to $\mathbb{ID}_S$. The Xth data packet

SC(X1, X2)

D = $\mathrm{S_i}$, X1, X2, $\mathrm{H(M_{A,X2})}$, $\mathrm{H_B^Z(1)}$, $\mathrm{H_B^{Z-(X2-X1)-1}(1)}$

$\mathbf{S_t}$ = **H(** $\mathrm{Sig_A(S_i, X2, H(M_{A,X2}))}$, $\mathrm{Sig_B(S_i, H_B^Z(1), X1)}$ **)**

a. Bidirectional data transmission.

SC(X)

D = $\mathrm{S_i}$, $\mathrm{H_D^{Z-X}(1)}$, $\mathrm{H_D^Z(1)}$, $\mathrm{H_S^{Z-X}(1)}$, $\mathrm{H_S^Z(1)}$

$\mathbf{S_t}$ = **H(** $\mathrm{Sig_S(S_{i,P2}, H_S^Z(1))}$, $\mathrm{Sig_D(S_{i,P2}, H_D^Z(1))}$ **)**

b. Asymmetric paths.

Fig. 10. The check format of the bidirectional data transmission and asymmetric paths.

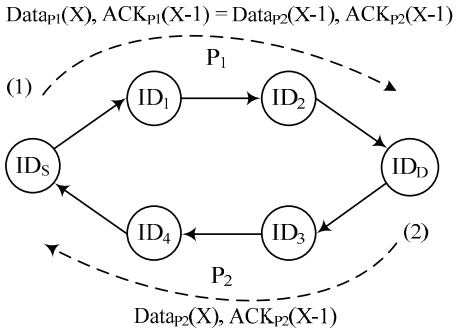

Fig. 11. FESCIM for asymmetric paths.

in P1 contains $\mathrm{Data_{P1}(X)}$ that is similar to the data packet in Fig. 3, and $\mathrm{ACK_{P1}(X-1)}$ that contains the destination node's *ACK* for the message $\mathrm{M_{X-1}}$, where $\mathrm{ACK_{P1}(X-1)}$ is sent from $\mathbb{ID}_D$ to $\mathbb{ID}_S$ through P2. The Xth data packet in P1 also contains the source node's *ACK* for the $X-1$th data packet sent through P2 $(\mathrm{ACK_{p2}}(X-1))$ to enable $\mathbb{ID}_3$ and $\mathbb{ID}_4$ to compose checks. In other words, P1 transports the source node's messages and P2 transports the *ACK*s of the messages. The source and destination nodes compose two hash chains by iteratively hashing random values to obtain the hash chains' roots $(\mathrm{H_D^Z(1)}$ and $\mathrm{H_S^Z(1)})$. The source and destination nodes send the signatures $\mathrm{Sig_S(S_{i,P2}, H_S^Z(1))}$ and $\mathrm{Sig_D(S_{i,P2}, H_D^Z(1))}$ to authenticate the hash chains and link them to P2, where $\mathrm{S_{i,P2} = ID_S, ID_4, ID_3, ID_D, T_S, P_r}$. The source and destination nodes release one element from their hash chains, i.e., $\mathrm{Data_{P2}(X)}$ contains a hash value from the destination node's hash chain to acknowledge the data received from $\mathbb{ID}_S$ through P1, and $\mathrm{ACK_{P2}(X-1)}$ contains a hash value from the source node's hash chain to acknowledge receiving $\mathrm{Data_{P2}(X-1)}$. $\mathbb{ID}_1$ and $\mathbb{ID}_2$ can compose checks using the mechanism discussed in Section 4.1, and Fig. 10b shows the composed checks by $\mathbb{ID}_3$ and $\mathbb{ID}_4$.

5 Security Analysis

For *Double-Rewarding* attack, the attacker attempts to illegally increase its rewards by submitting a check multiple times. In order to thwart the attack and identify the attackers, the AC checks whether the check has been deposited using the check unique identifier $(\mathrm{S_i})$. For *Double-Spending* attack, the attacker attempts to generate identical checks for different sessions to pay once. Two checks cannot have the same identifier because it contains the identities of the session nodes and time stamp. For *Check-Forgery-and-Manipulation* attack, the attackers attempt to forge checks or manipulate valid checks to increase their rewards. This is not possible with using secure hash function and signature scheme because it is not possible to forge or modify the source and destination nodes' signatures and to compute the private keys from the public ones. It is also infeasible to compute the hash value of the source and destination nodes' signatures without computing the signatures, and to compute $\mathrm{H_D^{Z-X}(i)}$ from $\mathrm{H_D^{Z-X+1}(i)}$. The AC can identify the attackers when the verifications of their checks repeatedly fail.

For *Receiver-Robbery* attack, the source node colludes with some intermediate nodes to steal credits from the

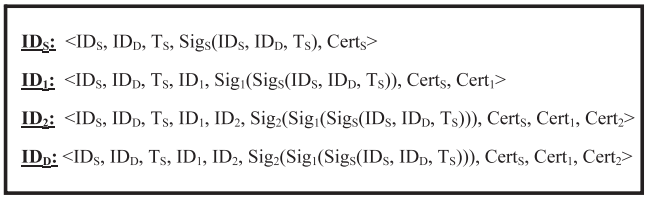

Fig. 12. Secured *RREQ* packets.

destination node by sending bogus messages paid by the source and destination nodes or composing faked or manipulated checks. For example, from Fig. 5, if the source node colludes with K intermediate nodes, the colluding intermediate nodes earn $(X \cdot \lambda \cdot K)$ credits but the source node pays $(X \cdot \lambda \cdot n \cdot P_r)$ for X messages. Obviously, the colluders can achieve gains when $(X \cdot \lambda \cdot K - X \cdot \lambda \cdot n \cdot P_r) > 0$ or $(K - n \cdot P_r) > 0$. In FESCIM, the colluding nodes cannot fake or manipulate checks to steal credits from the destination node because the destination node's signature and hash chain elements are required to compose a valid check. Moreover, a session cannot be established and a valid check cannot be composed if the destination node is not interested in the communication because its signature is required. The intermediate nodes are rewarded only when the destination node acknowledges receiving correct messages, and thus they do not earn from bogus messages. For *Packet-Replay* attacks, the internal or external attackers may record valid packets and replay them in different place and/or time claiming that they are fresh, to establish sessions under the name of other nodes to communicate freely. FESCIM uses time stamp to ensure that stale packets can be identified and dropped. For *Impersonation* attack, the attackers attempt to impersonate other nodes to communicate freely or steal credits. This attack is not possible because the nodes use their private keys to sign the packets, and the attackers cannot compute other nodes' private keys.

For *Free-Riding* attacks, two colluding intermediate nodes in a legitimate session manipulate the session packets to piggyback their data to communicate freely. To thwart this attack, the integrity of the packets should be checked at each node, and thus, the first node after the first colluder can detect the packet manipulation and drop the packet. For data and *RREP* packets, the source and destination nodes' signatures can ensure the packets' integrity. The integrity of the *ACK* packets can also be ensured because it contains one element of the hash chain, which is verified at each node. The *RREQ* packets are much shorter than the data packets, e.g., if a node's identity requires 4 bytes and the average number of nodes in a route is 5, the average size of the *RREQ* packet is nearly 20 bytes, which is incomparable with 512 bytes of the data packets. FESCIM can be implemented on the top of a secure routing protocol such as ARAN [36] and Ariadne [37]. In this protocol, the intermediate nodes authenticate themselves to thwart the external attacks, e.g., the external attackers that are not members in the network participate in route discovery phase with the intention of dropping the data packets to launch *Denial-of-Service* attacks. Fig. 12 shows secured *RREQ* packets that can authenticate the intermediate nodes and thwart packet manipulation. Each intermediate node appends its certificate and signs the packet's signature to authenticate itself. In this way, launching *Free-Riding* attacks against *RREQ* packets is not possible because the packet integrity can be checked at each node.

For *Denial-of-Check-Submission* attack, the colluding attackers residing close to the base station may attempt to prevent the nodes from submitting the checks to the AC. First, the nodes accumulate the checks and submit them to the AC in batch to reduce the communication overhead, e.g., checks may be submitted every few days, and thus, the nodes can repeatedly try to contact the AC during this time. Second, the nodes do not delete the checks before receiving the *Checks-Submission-Confirmation* packet from the AC. The nodes transmit the *Checks-Submission-Request* packet to the AC to submit the checks. This packet contains the identity of the submitter, time stamp, the checks, and the submitter's signature. The signature authenticates the submitter, thwarts *Packet-Replay* attack, and ensures the packet's integrity. The AC replies with the *Checks-Submission-Confirmation* packet containing the AC's signature for the checks to confirm the submission of the checks. Third, the nodes may change their cells due to the nodes' mobility, and thus, if they cannot submit the checks through one base station, they can submit them through other base stations. Finally, the nodes can contact the AC using wired networks such as the Internet and Wi-Fi, as explained in Section 3.1.

Due to using the postpaid payment policy, the nodes can communicate even if they do not have sufficient credits at the communication time. For *Payment-Denial* attacks, the attackers may join the network for a short time and leave without paying their debts. Different from the traditional ad hoc networks that can be temporarily established and similar to the current single-hop cellular networks, MCN is a long-life network where the nodes have long-term relations with the network. Moreover, the postpaid payment policy has been widely used successfully in many services such as credit cards and cellular networks. In FESCIM, each node needs a certificate to use the network. Issuing a certificate is not free to make changing an identity costly. Similar to the existing single-hop cellular networks, the AC stores the personal information of the users, and thus, it can take the legal procedures against the users who do not pay. To limit the debt, a certificate lifetime is short and depends on the node's credits at the certificate issuing time and its average credit consumption rate. We can also mix both the prepaid and the postpaid payment policies to reduce the debt amount, e.g., each node has to pay some money in advance at certificate renewal.

Our payment model can counteract the rational attacks and encourage node cooperation. Particularly, the rational attacker can exhibit one of the following actions:

1. If the intermediate nodes are rewarded for relaying the messages that do not reach the destination node [28], [29], the colluding nodes can drop a message and relay only the source node's signature that is much shorter than the message to claim the payment for relaying the message. In our payment model, the nodes are motivated to forward the messages because they are rewarded only when the destination node receives them.
2. If the source and destination nodes are charged only for the successfully delivered messages, the

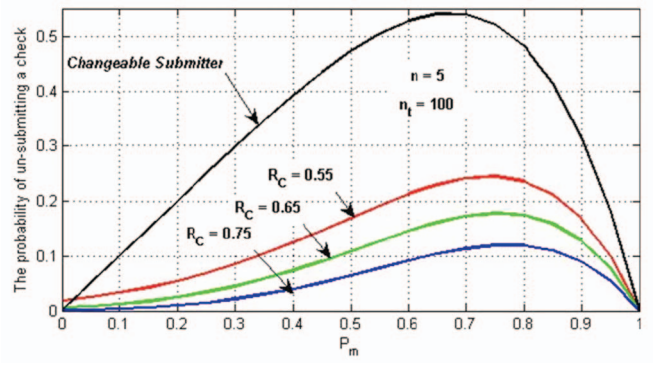

Fig. 13. The probability of unsubmitting a check.

destination node and/or colluding intermediate node may claim that the message does not reach the destination so as not to pay. To prevent this, both the source and destination nodes are charged for the undelivered messages.

3. Relaying the route establishment packets is beneficial for the nodes to participate in sessions and thus earn credits. If the destination node does not acknowledge a message, it pays for the message and the source node does not send the next message. Relaying the *ACK* packets is beneficial for the intermediate nodes to trigger the source node to generate more messages and thus earn more credits.

Although the charges are always more than or equal to the rewards, our payment model does not make credits disappear because purchasing credits for real money can compensate the credit loss. The rich nodes that have much more credits than their credit consumption may stop cooperation to save their resources. The AC converts credits to real money to motivate these nodes to cooperate because this conversion reduces the nodes' credits. Similar to the existing mechanisms, we believe that *ACK* packets are essential to secure the payment whether only the source node pays or both the source and destination nodes pay. That is because 1) if only the source node pays, some intermediate nodes may drop the messages and save only the source node's signature to be rewarded for the undelivered messages; and 2) if both the source and destination nodes pay, the source node may collude with some intermediate nodes to steal credits from the destination node by sending bogus messages.

In order to evaluate the security strength of the proposed check submission schemes, we introduce (3) and (4) which give the probabilities that a check is not submitted in the *Probabilistic* and *Changeable Submitter* schemes, respectively. They are equivalent to the probabilities that a session has at least one honest node and the honest nodes do not submit the check. $P_h(i,n)$ given in (5) is the probability that i honest nodes participate in a session with n intermediate nodes. $\mathrm{P_m}$ is the probability that an intermediate node is a colluder, which is equivalent to the ratio of colluding nodes in the network. $\mathrm{n_t}$ and n are the total number of nodes in the network and the number of intermediate nodes in a session, respectively. $\mathrm{n_h}$ and $\mathrm{n_c}$ are the number of honest and colluding nodes, where $\mathrm{n_h} = (1 - \mathrm{P_m}) \cdot \mathrm{n_t}$ and $\mathrm{n_c} = \mathrm{P_m} \cdot \mathrm{n_t}$. From Fig. 13, *in order to prevent submitting a low ratio of the checks in the Probabilistic-Check-Submission scheme, the source and destination nodes have to collude with a large number of nodes, which is not realistic for civilian applications and scalable network*. For example, in order to prevent submitting 10 percent of the checks, the ratios of the colluding nodes are 33 percent, 48 percent, and 64 percent when $\mathrm{R_C}$ is 0.55, 0.65, and 0.75, respectively. The figure also shows that the *Probabilistic* scheme can protect the check submission much better than the *Changeable-Submitter* scheme. The increase of $\mathrm{R_C}$ can reduce the probability of unsubmitting a check but with additional redundant checks. The probability of unsubmitting a check decreases at large value of $\mathrm{P_m}$ because the probability that an honest node participates in the session decreases:

$$\mathrm{P_u}(probabilistic\ \text{scheme}) = \sum_{i=1}^{i=n} p_h(i,n) \cdot (1 - \mathrm{R_c})^{\mathrm{i}}, \quad (3)$$

$$\mathrm{P_u}(Changeable\ Submitter\ \text{scheme}) = \sum_{i=1}^{i=n} p_h(i,n) \cdot \frac{\mathrm{n} - \mathrm{i}}{\mathrm{n}}, \quad (4)$$

$$P_h(i,n) = \binom{n}{i} \cdot \frac{\prod_{\mathrm{f}=0}^{\mathrm{i}-1}(\mathrm{n_h} - \mathrm{f}) \cdot \prod_{b=0}^{\mathrm{n}-\mathrm{i}-1}(\mathrm{n_c} - \mathrm{b})}{\prod_{\mathrm{k}=0}^{\mathrm{n}-1}(\mathrm{n_t} - \mathrm{k})}. \quad (5)$$

In Table 2, T is the total number of submitted checks in a noncollusion case, and Q is the number of unrepeated checks submitted under the collusion attack for a route with five intermediate nodes and 1,000 checks. It can be seen that there is no redundant checks ($\mathrm{cost} = 0$) in *Fixed-Submitter* and *Changeable-Submitter* schemes because each check is

TABLE 2
The Number of Submitted Checks, $\mathrm{n} = 5$

	Fixed-Submitter [30]	*All-Submitters*		*Changeable-Submitter*	*Probabilistic-Check-Submission*		
		Sprite	Express		$\mathrm{R_C} = 0.46$	$\mathrm{R_C} = 0.56$	$\mathrm{R_C} = 0.71$
C = 0	Q = 1000	Q = 1000		Q = 1000	$P(Q \geq 946) = 0.9$	$P(Q \geq 978) = 0.93$	$P(Q \geq 996) = 0.94$
C = 1	Q = 0	Q = 1000		Q = 800	$P(Q \geq 904) = 0.9$	$P(Q \geq 955) = 0.91$	$P(Q \geq 989) = 0.94$
C = 2	Q = 0	Q = 1000		Q = 600	$P(Q \geq 828) = 0.9$	$P(Q \geq 903) = 0.92$	$P(Q \geq 969) = 0.92$
C = 3	Q = 0	Q = 1000		Q = 400	$P(Q \geq 690) = 0.91$	$P(Q \geq 790) = 0.91$	$P(Q \geq 905) = 0.9$
C = 4	Q = 0	Q = 1000		Q = 200	$P(Q \geq 440) = 0.9$	$P(Q \geq 540) = 0.9$	$P(Q \geq 692) = 0.9$
T	1000	6000	7000	1000	2300	2800	3550
Cost	0	5000	6000	0	1300	1800	2550

submitted once. However, for the *Fixed-Submitter* scheme, if the source and destination nodes collude with the last intermediate node in the route ($C = 1$), all the checks are not submitted. The *Changeable-Submitter* scheme is more secure than the *Fixed-Submitter* scheme because the source and destination nodes have to collude with a larger number of nodes to achieve gains. The *all-Submitters* scheme is not vulnerable to the collusion attacks because each node submits all the checks, but the number of redundant checks is large because six and seven copies of each check are submitted in Sprite and Express, respectively. For the *Probabilistic-Check-Submission* scheme, the increase of R_C enhances the robustness against the collusion attack but with additional redundant checks. For example, the increase of R_C from 0.56 to 0.71 increases the number of submitted checks probabilistically from 903 to 969 when C equals 2 but with increasing the cost from 1,800 to 2,550. Moreover, the *Probabilistic-Check-Submission* scheme is less sensitive to the number of colluders comparing with the *Changeable-Submitter* scheme, e.g., at $R_C = 0.56$, when C increases from 1 to 3, Q drops from 800 to 400 in the *Changeable-Submitter* scheme and from 955 to 790 in the *Probabilistic-Check-Submission* scheme.

Choosing a proper value for R_C is important for reducing the number of redundant checks and mitigating the collusion attacks. The value of R_C may depend on the likelihood or the ease of attacking the incentive mechanism, e.g., the ease of obtaining multiple identities and compromising a device. The AC can dynamically estimate the value of R_C and broadcast it to the nodes. The AC can increase R_C to improve the protection level when more nodes complain that they receive less rewards than their estimations. Moreover, storage area should not be the main concern and the more important factors are bandwidth and energy because the capacities of the flash memories continue to rise as per Moore's Law and their costs continue to plummet [38]. Each node can store all the checks but submit only the ratio of R_C. The AC informs the nodes about the identifiers of the cleared checks, and thus, the nodes delete them and submit the uncleared checks to the AC.

6 Performance Evaluation

In this section, we evaluate the checks' overhead in terms of the check size and the number of generated checks. We also evaluate the overhead of the signed and hash-chain-based *ACK*s in terms of energy consumption and end-to-end packet delay.

6.1 Simulation Setup

In our simulation, we consider the RSA signature scheme and SHA-1 hash function with digest length of 20 bytes [39], [40]. Although the signature tags of the DSA and ECDSA signature schemes are shorter than that of the RSA, these schemes increase the end-to-end delay significantly because the verifying operations performed by the intermediate and destination nodes are computationally more demanding than the signing operations performed by the source node. In [41], it is shown that the verification time of the 1,024-bit RSA is more than 31 and 45 times faster than those of the 168-bit ECDSA and 1,024-bit DSA, respectively, and the signature generation is measured to be around 8 and 6 times slower. According to NIST guidelines [42], the secure private keys should have at least 1,024 bits.

TABLE 3
The Processing Times and Energy for RSA and SHA-1

	Processing time (ms)	Processing energy (mJ)
Signing operation	15.63	546.5
Verifying operation	0.53	15.97
SHA-1	16.79Megabytes/s (29μs/512 bytes)	0.76 μ J/B

In order to estimate the computational processing times for the signing, verifying, and hashing operations, we have implemented 1,024-bit RSA and SHA-1 using the Crypto++ library [43]. The mobile node is a laptop with an Intel processor at 1.6 GHZ and 1 GB Ram, and the operating system of the mobile node is Windows XP. The results given in Table 3 indicate that the RSA signature generation is computationally intensive but the signature verification is much faster. The energy consumption of the RSA and SHA-1 operations is measured in [44] and the results are given in Table 3. The resources of a real mobile node may be less than a laptop, so the results given in Table 3 are scaled by the factor of 5 in our simulations to estimate a limited-resource node.

We simulate an MCN in a square cell of $1{,}000\ \mathrm{m} \times 1{,}000\ \mathrm{m}$. Thirty-five mobile nodes are randomly deployed and a fixed base station is located at the center of the cell. The radio transmission range of the mobile nodes and the base station is 125 m. We use the modified random waypoint model [45] to emulate the nodes' mobility. Specifically, a node travels toward a random destination uniformly selected within the network field; upon reaching the destination, it pauses for some time; and the process repeats itself afterward. The node speed is uniformly distributed in the range $[0, S_{max}]$ m/s and the pause time is 20 s. The constant-bit-rate traffic source is implemented in each node as an application layer. The source and destination pairs are randomly selected. All data packets have 512 bytes and are sent at the rate of 0.5 packets/s. We simulate the dynamic source routing (DSR) protocol [32] over distributed coordination function of the IEEE 802.11 medium access control protocol. The time stamp, node's identity, and the number of messages are 5, 4, and 2 bytes, respectively. Network simulator NS2 (version 2.27) is used to evaluate the end-to-end delay and the network throughput using signed and hash-chain-based *ACK*s, and MATLAB is used to evaluate the check overhead. Our simulation is executed for 15 simulated min and each data point represents the average of 50 runs.

6.2 Simulation Results

6.2.1 Check Overhead

The simulation results given in Table 4 demonstrate that although FESCIM adopts more fair charging policy than Sprite and Express, the check size can be less. This is because of hashing the source and destination nodes' signatures and signing the hash of the message instead of the message, i.e., hashing the signatures can alleviate the effect of the long RSA signature tag. For FESCIM, a check

TABLE 4
Average Check Size (Bytes)

	Sprite	Express	FESCIM
Ad Hoc Mode	202.6	196	86.6 + 40 · i
Hybrid Mode	214.53	196	98.53 + 40 · i

size depends on the number of used hash chains in the session (i) because two hash values are attached to the check per hash chain. If the hash-chain size is long enough, FESCIM can generate one fixed-size check per route. A storage area of 1 MB can store up to 8,283; 5,176; and 5,350 checks in FESCIM, Sprite, and Express, respectively, when one hash chain is used in FESCIM.

Table 5 gives the statistical distribution of the number of hash chains used for a route. The simulation results demonstrate that more hash chains are used in low node mobility because more packets are transmitted before the route is broken. It can also be seen that the probability of using only one hash chain increases with the increase of Z. Properly choosing Z can reduce the number of used hash chains, which reduces the check size and saves the destination node's resources because the unused hash values in a chain should not be used for other routes to secure the payment. A good Z depends on the average number of transmitted packets before the route is broken, which is related to the packet transmission rate, the node speed, and the expected number of packets transmitted in the session.

Table 6 gives the expected number of checks and the amount of paid and earned credits in Sprite, Express, and FESCIM for a 300-second data transmission with different node speed. During the data transmission, a new route is established when the route is broken. For Sprite and Express, the simulation results demonstrate that the number of checks is large due to generating a check per message. The increase of the nodes' speed increases the number of checks because checks are generated for undelivered packets. FESCIM can significantly reduce the number of checks due to generating one check per route. More checks are generated at high node mobility because the routes are more frequently broken, i.e., the messages are transmitted over larger number of routes.

For Sprite and Express, the amount of paid credits is equal to the amount of earned credits because the payment is only for the nodes that relayed the undelivered messages. For FESCIM, the amount of paid credits is always more than or equal to the amount of earned credits because the source and destination nodes are charged full payment for the undelivered messages to reduce the number of submitted

TABLE 5
The Statistical Distribution of the Number of Used Hash Chains

S_{max}	Hash chain size (Z+1)	P(i = 1)	P(i = 2)	P(i = 3)	P(i > 3)
3 m/s	30	0.48	0.24	0.11	0.17
	50	0.6	0.28	0.12	0
10 m/s	30	0.89	0.11	0	0
	50	0.99	0.01	0	0

TABLE 6
The Number of Checks and the Amount of Paid and Earned Credits

Node speed		Number of cheques	Paid credits	Earned credits
[0, 3] m/s	Sprite and Express	153.32	523.23 · λ	523.23 · λ
	FESCIM	4.32	528.95 · λ	517.5 · λ
[0, 10]m/s	Sprite and Express	159.71	534.25 · λ	534.25 · λ
	FESCIM	10.71	550 · λ	517.5 · λ

checks. The increase of the nodes' speed increases the amount of paid credits because more credits are paid for the undelivered messages. In FESCIM, the amount of earned credits does not depend on the nodes' speed because the nodes earn credits only for the delivered messages. However, more credits are paid at high speed because the source and destination nodes pay for more undelivered messages.

6.2.2 ACK Overhead

For Z messages, the destination node generates Z signatures in signed *ACK*-based incentive mechanisms, but one signature and Z hashing operations are required in the *ACK* packets of FESCIM. From Table 3, we can see that the computational times of the signing and verifying operations are sufficient for 539 and 18 hashing operations for 512 bytes, respectively; and the consumed energy for the signing and verifying operations are sufficient for 1,404 and 41 hashing operations for 512 bytes, respectively, which demonstrates that FESCIM can adopt fair charging policy almost computationally free.

The average end-to-end delay is the average time interval between the data packet transmission and the reception of the destination node's acknowledgment. From Fig. 14, it can be seen that FESCIM can significantly reduce the end-to-end delay due to replacing the destination node's signature with the lightweight hashing operations. Above 20 connections, the delay increases significantly with and without implementing the cooperation incentive mechanism due to the significant increase in the channel contention and queuing delays.

The average network throughput is computed by dividing the size of the received data by all the nodes over

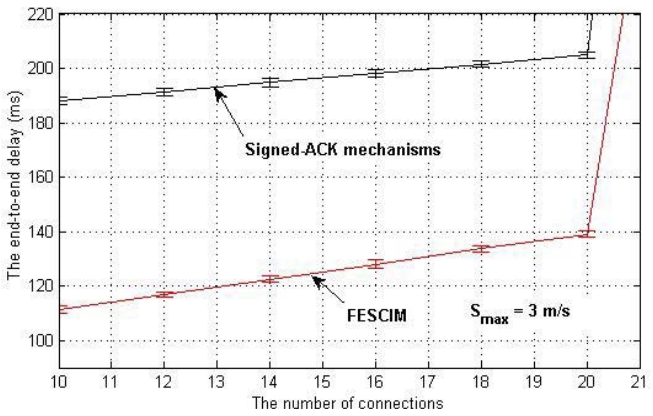

Fig. 14. The end-to-end delay for the signed ACK mechanisms and FESCIM.

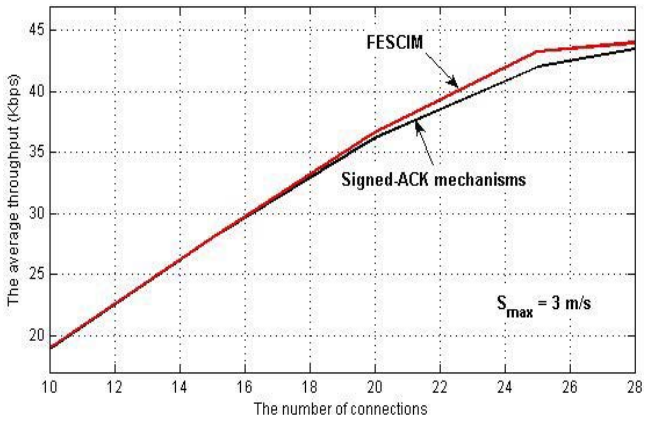

Fig. 15. The throughput for the signed ACK mechanisms and FESCIM.

the simulation time. As shown in Fig. 15, it can be seen that increasing the number of connections increases the network throughput until the network reaches its capacity. FESCIM is little better because more congestions occur in signed ACK mechanisms due to the longer packet processing delay in the intermediate nodes.

7 Conclusion and Future Work

In this paper, we have proposed a fair, efficient, and secure cooperation incentive mechanism for MCN. In order to fairly and efficiently charge the source and destination nodes, the lightweight hashing operations are used to reduce the number of public-key-cryptography operations. Moreover, to reduce the overhead of the payment checks, one small-size check is generated per session instead of generating a check per message, and the *Probabilistic-Check-Submission* scheme has been proposed to reduce the number of submitted checks and protect against the collusion attack. Extensive analysis and simulations have demonstrated that our incentive mechanism can secure the payment and significantly reduce the overhead of storing, submitting, and processing the checks. In addition, replacing the destination node's signatures with the hashing operations can charge the source and destination nodes almost computationally free.

In this paper, instead of generating two signatures per packet (one from the source and the other from the destination), we have replaced the destination node's signature with hashing operations to reduce the number of public-key-cryptography operations nearly by half. The source node attaches a signature in each data packet to ensure the payment nonrepudiation and to verify the message integrity at each intermediate node to thwart *Free-Riding* attacks. In our future work, we will focus on reducing the number of public-key-cryptography operations due to the source node's signatures. Although the payment nonrepudiation can be achieved using a hash chain at the source node side, we will study how to efficiently verify the message integrity at each intermediate node. In addition, similar to the existing incentive mechanisms, FESCIM can thwart selfishness attacks, but it cannot identify the irrational nodes that involve themselves in sessions with the intention of dropping the data packets to launch *Denial-of-Service* attacks. In our future work, we will study how the AC can process the checks to identify the irrational nodes. In FESCIM, if two nodes $\mathbb{ID}_A$ and $\mathbb{ID}_B$ submit checks with $\mathrm{Sig_S}(\mathrm{S_i}, \mathrm{X}, \mathrm{H}(\mathrm{M_X}))$ and $\mathrm{Sig_S}(\mathrm{S_i}, \mathrm{X}-1, \mathrm{H}(\mathrm{M_{X-1}}))$, the AC can learn that the data packet number X is dropped by an intermediate node A, B, or in-between node. However, packets may be dropped sometime, e.g., due to mobility or bad channel, or maliciously, but frequently dropping packets is an obvious malicious behavior. In our future work, we will study how the AC can precisely differentiate between the honest nodes and the irrational packet droppers in order to reduce the number of honest nodes that are falsely identified as irrational packet droppers.

Acknowledgments

This paper was presented in part at the 2009 IEEE International Conference on Communications.

References

[1] Y. Lin and Y. Hsu, "Multihop Cellular: A New Architecture for Wireless Communications," *Proc. IEEE INFOCOM,* vol. 3, pp. 1273-1282, Mar. 2000.

[2] X. Li, B. Seet, and P. Chong, "Multihop Cellular Networks: Technology and Economics," *Computer Networks,* vol. 52, no. 9, pp. 1825-1837, June 2008.

[3] C. Gomes and J. Galtier, "Optimal and Fair Transmission Rate Allocation Problem in Multi-Hop Cellular Networks," *Proc. Int'l Conf. Ad-Hoc, Mobile and Wireless Networks,* pp. 327-340, Aug. 2009.

[4] H. Wu, C. Qios, S. De, and O. Tonguz, "Integrated Cellular and Ad Hoc Relaying Systems: iCAR," *IEEE J. Selected Areas in Comm.,* vol. 19, no. 10, pp. 2105-2115, Oct. 2001.

[5] G. Shen, J. Liu, D. Wang, J. Wang, and S. Jin, "Multi-Hop Relay for Next-Generation Wireless Access Networks," *Bell Labs Technical J.,* vol. 13, no. 4, pp. 175-193, 2009.

[6] R. Schoenen, R. Halfmann, and B. Walke, "MAC Performance of a 3GPP-LTE Multihop Cellular Network," *Proc. IEEE Int'l Conf. Comm. (ICC),* pp. 4819-4824, May 2008.

[7] 3rd Generation Partnership Project, Technical Specification Group Radio Access Network, "Opportunity Driven Multiple Access," 3G Technical Report 25.924, Version 1.0.0, Dec. 1999.

[8] S. Marti, T. Giuli, K. Lai, and M. Baker, "Mitigating Routing Misbehavior in Mobile Ad Hoc Networks," *Proc. ACM MobiCom,* pp. 255-265, Aug. 2000.

[9] P. Michiardi and R. Molva, "Simulation-Based Analysis of Security Exposures in Mobile Ad Hoc Networks," *Proc. European Wireless Conf.,* Feb. 2002.

[10] J. Hu, "Cooperation in Mobile Ad Hoc Networks," Technical Report TR-050111, Computer Science Dept., Florida State Univ., Jan. 2005.

[11] G. Marias, P. Georgiadis, D. Flitzanis, and K. Mandalas, "Cooperation Enforcement Schemes for MANETs: A Survey," *J. Wireless Comm. and Mobile Computing,* vol. 6, no. 3, pp. 319-332, 2006.

[12] C. Song and Q. Zhang, "OMH-Suppressing Selfish Behavior in Ad Hoc Networks with One More Hop," *Mobile Networks and Applications,* vol. 14, no. 2, pp. 178-187, Feb. 2009.

[13] D. Djenouri and N. Badache, "On Eliminating Packet Droppers in MANET: A Modular Solution," *Ad Hoc Networks,* vol. 7, no. 6, pp. 1243-1258, Aug. 2009.

[14] G. Bella, G. Costantino, and S. Riccobene, "Evaluating the Device Reputation Through Full Observation in MANETs," *J. Information Assurance and Security,* vol. 4, no. 5, pp. 458-465, Mar. 2009.

[15] L. Feeney, "An Energy-Consumption Model for Performance Analysis of Routing Protocols for Mobile Ad Hoc Networks," *Mobile Networks and Applications,* vol. 3, no. 6, pp. 239-249, 2001.

[16] M. Peirce and D. O'Mahony, "Micropayments for Mobile Networks," technical report, Dept. of Computer Science, Trinity College, 1999.

[17] L. Buttyan and J. Hubaux, "Enforcing Service Availability in Mobile Ad-Hoc WANs," *Proc. ACM MobiHoc,* pp. 87-96, Aug. 2000.

[18] L. Buttyan and J. Hubaux, "Stimulating Cooperation in Self-Organizing Mobile Ad Hoc Networks," *Mobile Networks and Applications,* vol. 8, no. 5, pp. 579-592, Oct. 2004.

[19] Y. Zhang, W. Lou, and Y. Fang, "A Secure Incentive Protocol for Mobile Ad Hoc Networks," *ACM Wireless Networks*, vol. 13, no. 5, pp. 569-582, Oct. 2007.
[20] A. Weyland and T. Braun, "Cooperation and Accounting Strategy for Multi-Hop Cellular Networks," *Proc. IEEE Local and Metropolitan Area Networks (LANMAN '04)*, pp. 193-198, Apr. 2004.
[21] A. Weyland, "Cooperation and Accounting in Multi-Hop Cellular Networks," PhD thesis, Univ. of Bern, Nov. 2005.
[22] J. Hubaux, L. Buttyán, and S. Capkun, "The Quest for Security in Mobile Ad Hoc Networks," *Proc. ACM Symp. Mobile Ad Hoc Networking and Computing*, Oct. 2001.
[23] A. Weyland, T. Staub, and T. Braun, "Comparison of Motivation-Based Cooperation Mechanisms for Hybrid Wireless Networks," *J. Computer Comm.*, vol. 29, pp. 2661-2670, 2006.
[24] M. Jakobsson, J. Hubaux, and L. Buttyan, "A Micro-Payment Scheme Encouraging Collaboration in Multi-Hop Cellular Networks," *Proc. Seventh Financial Cryptography (FC '03)*, pp. 15-33, Jan. 2003.
[25] M. Mahmoud and X. Shen, "Stimulating Cooperation in Multi-Hop Wireless Networks Using Cheating Detection System," *Proc. IEEE INFOCOM*, pp. 776-784, Mar. 2010.
[26] J. Pan, L. Cai, X. Shen, and J. Mark, "Identity-Based Secure Collaboration in Wireless Ad Hoc Networks," *Computer Networks*, vol. 51, no. 3, pp. 853-865, 2007.
[27] N. Salem, L. Buttyan, J. Hubaux, and M. Jakobsson, "Node Cooperation in Hybrid Ad Hoc Networks," *IEEE Trans. Mobile Computing*, vol. 5, no. 4, pp. 365-376, Apr. 2006.
[28] S. Zhong, J. Chen, and R. Yang, "Sprite: A Simple, Cheat-Proof, Credit Based System for Mobile Ad-Hoc Networks," *Proc. IEEE INFOCOM*, vol. 3, pp. 1987-1997, Mar./Apr. 2003.
[29] H. Janzadeh, K. Fayazbakhsh, M. Dehghan, and M. Fallah, "A Secure Credit-Based Cooperation Stimulating Mechanism for MANETs Using Hash Chains," *Future Generation Computer Systems*, vol. 25, no. 8, pp. 926-934, Sept. 2009.
[30] B. Lamparter, K. Paul, and D. Westhoff, "Charging Support for Ad Hoc Stub Networks," *J. Computer Comm.*, vol. 26, no. 13, pp. 1504-1514, 2003.
[31] S. Miner and J. Staddon, "Graph-Based Authentication of Digital Streams," *Proc. IEEE Symp. Security and Privacy*, pp. 232-246, May 2001.
[32] D. Johnson and D. Maltz, "Dynamic Source Routing in Ad Hoc Wireless Networks," *Mobile Computing*, pp. 153-181, chapter 5, Kluwer Academic, 1996.
[33] C. Perkins and E. Royer, "Ad-Hoc On-Demand Distance Vector Routing," *Proc. IEEE Workshop Mobile Computing Systems and Applications*, pp. 90-100, Feb. 1999.
[34] H. Pagnia and F. Gartner, "On the Impossibility of Fair Exchange without a Trusted Third Party," Technical Report TUD-BS-1999-02, Darmstadt Univ. of Technology, Mar. 1999.
[35] L. Anderegg and S. Eidenbenz, "Ad Hoc-VCG: A Trustful and Cost-Efficient Routing Protocol for Mobile Ad Hoc Networks with Selfish Agents," *Proc. ACM MobiCom*, Sept. 2003.
[36] K. Sanzgiri, D. LaFlamme, B. Dahill, B. Levine, C. Shields, and E. Belding-Royer, "Authenticated Routing for Ad Hoc Networks," *IEEE Selected Areas in Comm.*, vol. 23, no. 3, pp. 598-610, Mar. 2005.
[37] Y. Hu, A. Perrig, and D. Johnson, "Ariadne: A Secure On-Demand Routing Protocol for Ad Hoc Networks," *Proc. ACM MobiCom*, Sept. 2002.
[38] A. Mitra et al., "High-Performance Low Power Sensor Platforms Featuring Gigabyte Scale Storage," *Proc. IEEE/ACM Measurement, Modeling, and Performance Analysis of Wireless Sensor Networks*, 2005.
[39] A. Menzies, P. Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*, http://www.cacr.math.uwaterloo.ca/hac, CRC Press, 1996.
[40] Nat'l Inst. Standards and Technology (NIST), "Digital Hash Standard," Fed. Information Processing Standards Publication 180-1, Apr. 1995.
[41] M. Wiener, "Performance Comparison of Public-Key Cryptosystems," *Technical Newsletter of RSA Laboratories' CryptoBytes*, vol. 4, no. 1, pp. 3-5, 1998.
[42] Nat'l Inst. of Standards and Technology (NIST), "Recommendation for Key Management - Part 1: General (Revised)," Special Publication 800-57 200, 2007.
[43] W. Dai, "Crypto++ Library 5.6.0," http://www.cryptopp.com, 2011.
[44] N. Potlapally, S. Ravi, A. Raghunathan, and N. Jha, "A Study of the Energy Consumption Characteristics of Cryptographic Algorithms and Security Protocols," *IEEE Trans. Mobile Computing*, vol. 5, no. 2, pp. 128-143, Mar./Apr. 2006.
[45] J. Yoon, M. Liu, and B. Nobles, "Sound Mobility Models," *Proc. ACM MobiCom*, Sept. 2003.

Mohamed M.E.A. Mahmoud received the PhD degree (April 2011) in electrical and computer engineering from the University of Waterloo, Ontario, Canada. He is currently a postdoctoral fellow with the Centre for Wireless Communications, Department of Electrical and Computer Engineering, University of Waterloo, under the supervision of Prof. Sherman (Xuemin) Shen. His research interests include wireless network security, privacy-preserving schemes, anonymous and secure routing protocols, trust and reputation systems, cooperation incentive mechanisms, and cryptography. Dr. Mahmoud is the first author for more than 20 papers published in major IEEE conferences and transactions. He won the prestigious Best Paper Award from the IEEE International Conference on Communications (ICC 2009), Dresden, Germany. This award was one of only 14 given, among 1,046 papers presented and more than 3,000 submitted, and was the sole award for the Communication and Information Systems Security Symposium. He has also served as a technical program committee member and reviewer for many conferences and journals. He is a member of the IEEE.

Xuemin (Sherman) Shen received the BSc degree from Dalian Maritime University, China, in 1982, and the MSc and PhD degrees from Rutgers University, Camden, New Jersey, in 1987 and 1990, respectively, all in electrical engineering. He is currently a professor and the university research chair with the Centre for Wireless Communications, Department of Electrical and Computer Engineering, University of Waterloo, Ontario, Canada. He is the author or coauthor of three books and more than 400 papers and book chapters on wireless communications and networks, control, and filtering. He serves as the editor-in-chief for peer-to-peer networking and applications and an associate editor for *Computer Networks*, *ACM/Wireless Networks*, and *Wireless Communications and Mobile Computing*. He has also served as a guest editor for *ACM Mobile Networks and Applications*. His research focuses on mobility and resource management in interconnected wireless/wired networks, ultra wideband wireless communications networks, wireless network security, wireless body area networks, and vehicular ad hoc and sensor networks. He is a registered professional engineer in the Province of Ontario and a distinguished lecturer of the IEEE Communications Society. He received the Excellent Graduate Supervision Award in 2006 and the Outstanding Performance Award in 2004 and 2008 from the University of Waterloo, the Premier's Research Excellence Award in 2003 from the Province of Ontario, and the Distinguished Performance Award in 2002 and 2007 from the Faculty of Engineering, University of Waterloo. He served as the tutorial chair for the 2008 IEEE International Conference on Communications, the technical program committee chair for the 2007 IEEE Global Telecommunications Conference, the general cochair for the 2007 International Conference in Communications and Networking in China and the 2006 International Conference on Quality of Service in Heterogeneous Wired/Wireless Networks, and the founding chair for the IEEE Communications Society Technical Committee on P2P Communications and Networking. He also serves as a founding area editor for the *IEEE Transactions on Wireless Communications* and an associate editor for the *IEEE Transactions on Vehicular Technology* and the *KICS/IEEE Journal of Communications and Networks*. He has also served as a guest editor for the *IEEE Journal on Selected Areas in Communications*, *IEEE Wireless Communications*, and the *IEEE Communications Magazine*. He is a fellow of the IEEE.