A Secure Payment Scheme with Low Communication and Processing Overhead for Multihop Wireless Networks

Mohamed M. E. A. Mahmoud and Xuemin (Sherman) Shen, *Fellow, IEEE*
Department of Electrical and Computer Engineering
University of Waterloo, Waterloo, Ontario, Canada, N2L 3G1

***Abstract*—We propose RACE, a report-based payment scheme for multihop wireless networks to stimulate node cooperation, regulate packet transmission, and enforce fairness. The nodes submit lightweight payment reports (instead of receipts) to the accounting center (AC) and temporarily store undeniable security tokens called *Evidence*s. The reports contain the alleged charges and rewards without security proofs, e.g., signatures. The AC can verify the payment by investigating the consistency of the reports, and clear the fair reports with almost no processing overhead or cryptographic operations. For cheating reports, the *Evidences* are requested to identify and evict the cheating nodes that submit incorrect reports. Instead of requesting the *Evidence*s from all the nodes in the cheating reports, RACE can identify the cheating nodes with requesting few *Evidence*s. Moreover, *Evidence* aggregation technique is used to reduce the *Evidences*' storage area. Our analytical and simulation results demonstrate that RACE requires much less communication and processing overhead than the existing receipt-based schemes with acceptable payment clearance delay and storage area. This is essential for the effective implementation of a payment scheme because it uses micropayment and the overhead cost should be much less than the payment value. Moreover, RACE can secure the payment and precisely identify the cheating nodes without false accusations.**

***Index Terms*—**Cooperation incentive schemes; network-level security and protection; payment schemes; and selfishness attacks.

1. INTRODUCTION

In multihop wireless networks (MWNs), the traffic originated from a node is usually relayed through the other nodes to the destination for enabling new applications and enhancing the network performance and deployment [1]. MWNs can be deployed readily at low cost in developing and rural areas. Multihop packet relay can extend the network coverage using limited transmit power, improve area spectral efficiency, and enhance the network throughput and capacity. MWNs can also implement many useful applications such as data sharing [2] and multimedia data transmission [3]. For example, users in one area (residential neighborhood, university campus, etc) having different wireless-enabled devices, e.g., PDAs, laptops, tablets, cell phones, etc, can establish a network to communicate, distribute files, and share information. However, the assumption that the nodes are willing to spend their scarce resources, such as battery energy, CPU cycles, and available network bandwidth, to relay others' packets without compensation cannot be held for civilian applications where the nodes are autonomous and aim to maximize their welfare.

Selfish nodes will not relay others' packets and make use of the cooperative nodes to relay their packets, which degrades the network connectivity and fairness. The fairness issue arises when the selfish nodes make use of the cooperative nodes to relay their packets without any contribution to them, and thus the cooperative nodes are unfairly overloaded because the network traffic is concentrated through them. The selfish behavior also degrades the network connectivity significantly, which may cause the multihop communication to fail [4].

Payment (or incentive) schemes [5] use credits (or micropayment) to motivate the nodes to cooperate in relaying others' packets by making cooperation more beneficial than selfishness. The nodes earn credits for relaying others' packets and spend these credits to get their packets relayed by others. In addition to cooperation stimulation, these schemes can enforce fairness, discourage *Message-Flooding* attacks, regulate packet transmission, and efficiently charge for the network services. Fairness can be enforced by rewarding the nodes that relay more packets and charging the nodes that send more packets. For example, the nodes situated at the network center relay more packets than the other nodes because they are more frequently selected by the routing protocol. Since the source nodes pay for relaying their packets, the payment schemes can also regulate packet transmission and discourage *Message-Flooding* attacks where the attackers send bogus messages to deplete the intermediate nodes' resources. Moreover, since the communication sessions may be held without involving a trusted party and the nodes may roam among different foreign networks, the payment schemes can charge the nodes efficiently without contacting distant home location registers [6].

The existing credit card payment systems are designed for different system and threat models, which makes using them in MWNs infeasible. For example, in credit card payment systems, each transaction usually has one customer and one merchant, and the merchants' number is low and their identities are known before the transaction is held. For the payment schemes in MWNs, there is usually one customer (the source node) and multiple merchants (the intermediate nodes), the merchants' number is large because any network node can work as a merchant (or packet relay), a transaction's value is much less than those in the credit card payment systems, the relation between a customer and a merchant is usually short due to the network dynamic topology, and the nodes are involved in low-value transactions very frequently because once a route is broken, a new transaction should be held to re-establish the route. Due to these unique characteristics, MWNs need a specially designed payment scheme which is different from the existing credit card payment systems.

A good payment scheme should be secure due to using payment, and require low overhead. However, the existing receipt-based payment schemes impose significant processing and communication overhead and implementation complexity. Since a trusted party may not be involved in communication

sessions, the nodes compose proofs of relaying others' packets, called receipts, and submit them to an offline accounting center (AC) to clear the payment. The receipts' size is large because they carry security proofs, e.g., signatures, to secure the payment, which significantly consumes the nodes' resources and the available bandwidth in submitting them. The AC has to apply a large number of cryptographic operations to clear the receipts, which may require impractical computational power and make the practical implementation of these schemes complex or inefficient. Moreover, since a transaction (relaying packets) value may be very low, the scheme uses micropayment, and thus a transaction's overhead in terms of submitting and clearing the receipts should be much less than its value. Therefore, reducing the communication and the payment processing overhead is essential for the effective implementation of the payment scheme and to avoid creating a bottleneck at the AC and exhausting the nodes' resources.

In this paper, we propose **RACE,** a **R**eport-based p**A**yment s**C**hem**E** for MWNs. The nodes submit lightweight payment reports (instead of receipts) to the AC to update their credit accounts and temporarily store undeniable security tokens called *Evidence*s. The reports contain the alleged charges and rewards of different sessions without security proofs, e.g., signatures. The AC verifies the payment by investigating the consistency of the reports and clears the payment of the fair reports with almost no cryptographic operations or computational overhead. For cheating reports, the *Evidences* are requested to identify and evict the cheating nodes that submit incorrect reports, e.g., to steal credits or pay less. In other words, the *Evidence*s are used to resolve disputes when the nodes disagree about the payment. Instead of requesting the *Evidence*s from all the nodes in the cheating reports, RACE can identify the cheating nodes with submitting and processing few *Evidence*s. Moreover, *Evidence* aggregation technique is used to reduce the storage area of the *Evidences.*

In RACE, *Evidence*s are submitted and the AC applies cryptographic operations to verify them only in case of cheating, but the nodes always submit security tokens, e.g., signatures, and the AC always applies cryptographic operations to verify the payment in the existing receipt-based schemes. RACE can clear the payment nearly without cryptographic operations and with submitting lightweight reports when *Evidences* are not frequently requested. Wide-spread cheating actions are not expected in civilian applications because the common users do not have the technical knowledge to tamper with their devices. Moreover, cheating nodes are evicted once they commit one cheating action and it is neither easy nor cheap to change identities. Our analytical and simulation results demonstrate that the communication and processing overhead of RACE is much less than the existing receipt-based schemes with acceptable payment clearance delay and *Evidence*s' storage area, which is necessary to make the practical implementation of the payment scheme effective. Moreover, RACE can secure the payment and precisely identify the cheating nodes without false accusations or stealing credits.

To the best of our knowledge, RACE is the first payment scheme that can verify the payment by investigating the consistency of the nodes' reports without submitting and processing security tokens and without false accusations. RACE is also the first scheme that uses the concept of *Evidence* to secure the payment and requires cryptographic operations in clearing the payment only in case of cheating.

The remainder of this paper is organized as follows. Section 2 reviews the related works. Section 3 gives the network and adversary models. Section 4 presents RACE. Security analysis and performance evaluation are given in Sections 5 and 6, respectively, followed by conclusion and future work in Section 7.

2. Related Works

The existing payment schemes can be classified into tamper-proof-device (TPD) based and receipt-based schemes. In TPD-based payment schemes [7-10], a TPD is installed in each node to store and manage its credit account and secure its operation. For receipt-based payment schemes [11-20], an offline central unit called the accounting center stores and manages the nodes' credit accounts. The nodes usually submit undeniable proofs for relaying packets, called receipts, to the AC to update their credit accounts.

In Nuglets [7], the self-generated and forwarded packets by a node are passed to the TPD to decrease and increase the node's credit account, respectively. A packet purse and packet trade models have been proposed. For the packet purse model, the source node's credit account is charged the full payment before sending a packet and each intermediate node acquires the payment for relaying the packet. For the packet trade model, each intermediate node runs an auction to sell the packets to the next node in the route, and the destination node pays the total cost of relaying the packets. In SIP [8], after receiving a data packet, the destination node sends a RECEIPT packet to the source node to issue a REWARD packet to increment the credit accounts of the intermediate nodes. In CASHnet [9], the credit account of the source node is charged and a signature is attached to each data packet. Upon receiving the packet, the credit account of the destination node is also charged and a digitally signed ACK packet is sent back to the source node to increase the credit accounts of the intermediate nodes.

The receipt-based payment schemes cause more overhead than the TPD-based schemes because they require submitting receipts to the AC and processing them. However, the TPD-based payment schemes suffer from the following serious issues. First, the assumption that the TPD cannot be tampered with, cannot be guaranteed because the nodes are autonomous and self-interested, and the attackers can communicate freely in an undetectable way if they could compromise the TPDs. Second, the nodes cannot communicate if they do not have sufficient credits during the communication time. Unfortunately, the nodes at the network border cannot earn as many credits as the other nodes because they are less frequently selected by the routing protocol. Finally, since credits are cleared in real time, the multihop communications fail if the network does not have enough credits circulating around because the nodes do not have sufficient credits to communicate. In [10], it is shown that the overall credits in the network decline gradually with using TPD-based schemes because the total charges may be more than the total rewards. This 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.

In order to eliminate the need for TPDs, an offline central bank called the AC is used to store and manage the nodes' credit accounts. In Sprite [11], for each message, the source node

signs the identities of the nodes in the route and the message and sends the signature as a proof for sending a message. The intermediate nodes verify the signature, compose receipts containing the identities of the nodes in the route and the source node's signature, and submit the receipts to the AC to claim the payment. The AC verifies the source node's signature to make sure that the payment is correct. However, the receipts overwhelm the network because the scheme generates a receipt per message.

Unlike Sprite that charges only the source node, FESCIM [12] adopts fair charging policy by charging both the source and destination nodes when both of them are interested in the communication. In PIS [13], the source node attaches a signature to each message and the destination node replies with a signed acknowledgement packet. PIS can reduce the receipts' number by generating a fixed-size receipt per session regardless of the number of messages instead of generating a receipt per message in Sprite. In order to reduce the communication and processing overhead, CDS [14] uses statistical methods to identify the cheating nodes that submit incorrect payment. However, due to the nature of the statistical methods, the colluding nodes may manage to steal credits, some honest nodes may be falsely accused of cheating which is called false accusations, some cheating nodes may not be identified which is called missed detections, and it may take long time to identify the cheating nodes.

In [15], a payment scheme has been proposed for hybrid ad-hoc networks, but involving the base stations in every communication session 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 stations before they are dropped because the intermediate nodes cannot verify the authenticity and the integrity of the messages. In [16], each node has to contact the AC in each communication session to get coins to buy packets from the previous node in the session. However, the interactive involvement of the AC in each session is not efficient, causes long delay, and creates a bottleneck at the AC.

ESIP [17] proposes a communication protocol that can be used for a payment scheme. ESIP transfers messages from the source to the destination nodes with limited number of public key cryptography operations by integrating public key cryptography, identity based cryptography, and hash function. Public key cryptography and hash function are used to ensure message integrity and payment non-repudiation to secure the payment. Identity based cryptography is used to efficiently compute a shared symmetric key between the source node and each node in the session. Using these keys, the source node attaches a keyed hash value for each intermediate node to verify the message integrity. Comparing to PIS, ESIP requires fewer public key cryptography operations but with larger receipts' size. Unlike ESIP that aims to transfer messages efficiently from the source to the destination nodes, RACE aims to reduce the overhead of submitting the payment data to the AC and processing them. Although ESIP can be used with RACE, we use a simple communication protocol due to space limitation and to focus on our contributions.

A mechanism is proposed in [18] to thwart packet dropping attacks. Payment is used to thwart the rational packet-dropping attacks, and reputation system is used to identify and evict the irrational packet dropping attackers once their packet-dropping rates exceed a threshold. In [19], Zhu et al. propose a payment scheme, called SMART, for delay tolerant wireless networks (DTNs). SMART uses layered coins and can secure the payment against new attacks such as *Credit-Forgery*, *Nodular-Tontine*, and *Submission-Refusal*. Lu et al. [20] propose a payment scheme for DTNs that focuses on the fairness issue. The intermediate nodes earn credits for forwarding the delivered messages and gain reputation for forwarding the undelivered messages which gives them preference in forwarding future messages. However, the proposed payment schemes for DTNs may not be efficiently applicable to MWNs because DTNs lack fully connected end-to-end routes and tolerate long packet delivery delay.

Table 1 summarizes the main features of RACE and the existing payment schemes. RACE is more secure than CDS because it does not suffer from false accusations, missed detections, collusion attacks, and delay in identifying attackers. Moreover, RACE requires much less communication and processing overhead comparing to receipt-based schemes [11-13, 17], but with more and acceptable storage area and payment clearance delay.

Table 1: Comparison between RACE and the existing payment schemes.

	RACE	**Receipt-based schemes [11-13, 17]**	**CDS**
Communication overhead	Low	Large	Low
Payment processing overhead	Fair reports: light overhead Cheating reports: Cryptographic operations	Cryptographic operations are always used	Lightweight statistical operations
Payment clearance delay	Much shorter than CDS in case of cheating	The shortest delay	Very long delay in case of cheating
Storage area	More than receipt-based schemes	More than CDS and less than RACE	Smallest storage area
Security	- No false accusations and missed detections - Strong protection against colluders - Cheaters are identified after the first cheating action	- No false accusations and missed detections - Strong protection against colluders - Cheaters are identified after the first cheating action	- False accusations and missed detections. - Vulnerable to collusion attacks - Long time to identify cheaters

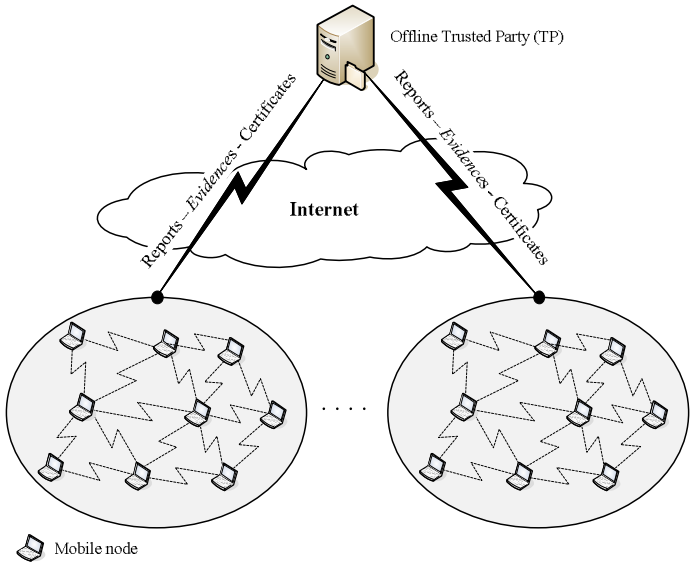

Fig. 1: The architecture of the considered network.

3. System Models

3.1 Network Model

For military and disaster recovery applications of MWNs, the network can be considered ephemeral because it is used for a specific purpose and short duration. In this paper, we adopt the network model used in [7-17] that targets the civilian applications of MWNs where the network has long life and the nodes have long-term relations with the network. As illustrated in Fig. 1, the considered MWN has an offline trusted party (TP) and mobile nodes. The TP contains the accounting center (AC) and the certificate authority (CA). The AC maintains the nodes' credit accounts and the CA renews and revokes the nodes' certificates. Each node A has to register with the trusted party to receive a symmetric key K_A, private/public key pair, and certificate. The symmetric key is used to submit the payment reports and the private/public keys are required to act as source or destination node. We assume that the clocks of the nodes are synchronized. The details of this synchronization process are out of the scope of the paper, but several mechanisms have been proposed to synchronize the nodes [21]. Once the AC receives the payment reports of a session and verifies them, it clears the payment if the reports are fair; else, it requests the *Evidence*s to identify the cheating nodes. The CA evicts the cheating nodes by denying renewing their certificates.

RACE can be implemented with any source routing protocol, such as DSR [22], which establishes end-to-end routes before transmitting data. Source nodes' packets may be relayed several hops by intermediate nodes to their destinations. The nodes can contact the TP at least once during a period of few days. In this connection, the nodes submit the payment reports and the *Evidence*s (if requested), receive renewed certificates to be able to continue using the network, and purchase credits with real money to enable the nodes that cannot earn sufficient credits, such as those at the network border, to communicate, and also to avoid credit decline because the total charges may be more than the rewards when routes are broken. This connection can occur via the base stations of cellular networks, Wi-Fi hotspots, or wired networks such as Internet.

For the payment model, source nodes are charged for every transmitted message even if it does not reach the destination nodes, but the intermediate nodes are rewarded only for the delivered messages. Some schemes, such as [23], consider that the reward of relaying a packet is proportional to the incurred energy in relaying the packet. This rewarding policy can be integrated with RACE, but similar to [11-20], we use fixed rewarding rate, e.g., λ credits per unit-sized packet to simplify our description and focus on our contributions.

3.2 Adversary Model

The mobile nodes are probable attackers but the TP is fully secure. The mobile nodes are autonomous and self-interested and thus motivated to misbehave. The TP is run by an operator that is motivated to ensure the network proper operation. As discussed in [24], it is impossible to realize secure payment between two entities without a trusted third party. The attackers have full control on their nodes and can change their operation and infer the cryptographic data. The attackers can work individually or collude with each other under the control of one attacker to launch sophisticated attacks. These strong assumptions are necessary due to implementing payment in the network. Similar to [11-20], the attackers are rational in the sense that they misbehave only when they can achieve more benefits than behaving honestly. Particularly, the attackers aim to steal credits, pay less, and communicate for free. Table 2 gives the description of the used symbols in this paper.

Table 2: Description of the used symbols.

Symbol	Description
X, Y	X is concatenated to Y.
F	A flag bit indicating whether the last received packet by a node is for acknowledgement (ACK) or data.
$h^{(i)}$	The hash value number i in a hash chain generated by the destination node.
H(P)	The hash value resulted from hashing P.
$H_K(P)$	The keyed hash value resulted from hashing P using the key K.
ID_A	The identity of an intermediate node A.
ID_S and ID_D	The identities of the source node (S) and the destination node (D), respectively.
K_A	The shared key between node A and the TP.
M_X	The message sent in the *X*th data packet.
n	The number of nodes in a route.
$P_C(n)$	The average payment clearance delay for a route with n nodes.
R	The concatenation of the identities of the nodes in a route, e.g., $R = ID_S, ID_A, \ldots, ID_D$.
$Sig_S(P)$ and $Sig_D(P)$	The signatures of the source and the destination nodes on P, respectively.
T_{Cert}	The lifetime of a certificate.
Ts	The time stamp of a route establishment.

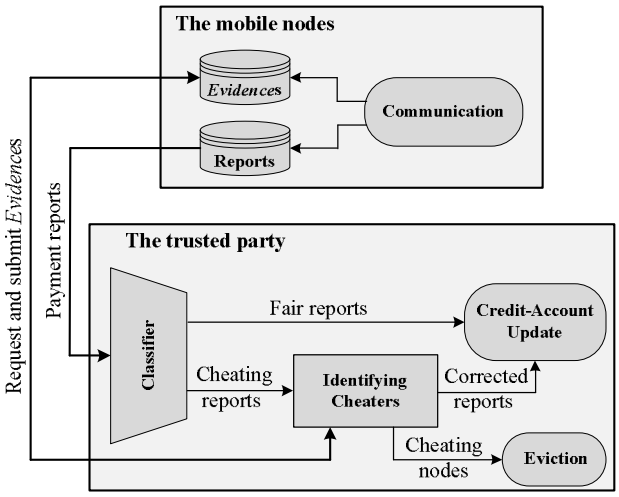

Fig. 2: The architecture of RACE.

4. The Proposed RACE

As shown in Fig. 2, RACE has four main phases. In *Communication* phase, the nodes are involved in communication sessions and *Evidence*s and payment reports are composed and temporarily stored. The nodes accumulate the payment reports and submit them in batch to the TP. For the *Classifier* phase, the TP classifies the reports into fair and cheating. For the *Identifying Cheaters* phase, the TP requests the *Evidence*s from the nodes that are involved in cheating reports to identify the cheating nodes. The cheating nodes are evicted and the payment reports are corrected. Finally, in *Credit-Account Update* phase, the AC clears the payment reports.

Algorithm 1: Data transmission/composition of *Evidence* and report

```
1: // n_i is the node running the algorithm.
2: if (n_i is source node) then
3:       P_X ← R, X, Ts, M_X, Sig_S(R, X, Ts, H(M_X));
4:       Send(P_X) to the first node in the route;
5: else
6:       if ((R, X, Ts are correct) or Verify(Sig_S(R, X, Ts, H(M_X)) ==
                                                    TRUE )) then
7:             if (n_i is intermediate node) then
8:                Relay the packet;
9:                Store Sig_S(R, X, Ts, H(M_X));
10:            end if
11:            if (n_i is destination node) then
12:               Send(h^(X));
13:            end if
14:      else
15:            Drop the packet;
16:            Send error packet to the source node;
17:      end if
18: end if
19: if (P_X is last packet) then
20:      Evidence = {R, X, Ts, H(M_X), h^(0), h^(X),H(Sig_S(R, X, Ts,
                                          H(M_X)), Sig_D(R, Ts, h^(0))};
21:      Report = {R, Ts, F, X};
22:      Store Report and Evidence;
23:  end if
```

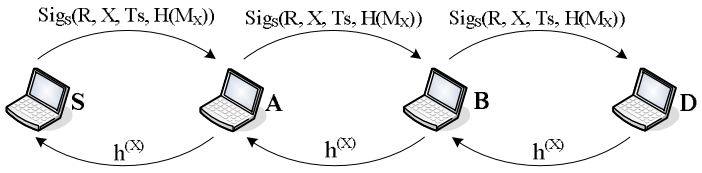

Fig. 3: The security tokens of the *X*th data and ACK packets.

4.1 Communication

The *Communication* phase consists of route establishment, data transmission, *Evidence* composition, and payment-report composition/submission processes.

Route establishment: In order to establish an end-to-end route, the source node broadcasts the *Route Request* (*RREQ*) packet containing the identities of the source (ID_S) and the destination (ID_D) nodes, time stamp (Ts), and Time-To-Live (TTL) or the maximum number of intermediate nodes. After a node receives the *RREQ* packet, it appends its identity and broadcasts the packet if the number of intermediate nodes is fewer than TTL. The destination node composes the *Route Reply* (*RREP*) packet for the nodes relayed the first received *RREQ* packet and sends the packet back to the source node. The destination node generates a hash chain by iteratively hashing a random value ($h^{(K)}$) K times to produce the hash chain root ($h^{(0)}$), where $h^{(i-1)} = H(h^{(i)})$ and $1 \le i \le K$. The optimal value of K depends on many factors such as the number of messages the source node needs to send, and the average number of messages sent through a route before it is broken. Estimating a good value for K can save the destination node's resources because once a route is broken, the unused hash values in the hash chain should not be used for another route to secure the payment. The nodes can estimate the value of K and periodically tune it.

The *RREP* packet contains the identities of the nodes in the route (e.g., R = ID_S, ID_A, ID_B, ID_D in the route shown in Fig. 3), $h^{(0)}$, and the destination node's certificate and signature ($Sig_D(R, Ts, h^{(0)})$). This signature authenticates the hash chain and links it to the route. The intermediate nodes verify the destination node's signature, relay the *RREP* packet, and store the signature and $h^{(0)}$ for composing the *Evidence*.

Data transmission: The source node sends data packets to the destination node through the established route and the destination node replies with ACK packets. For the *X*th data packet, the source node appends the message M_X and its signature to R, *X*, Ts, and the hash value of the message ($H(M_X)$) and sends the packet to the first node in the route. The security tokens of the *X*th data and ACK packets are illustrated in Fig. 3. The source node's signature is an undeniable proof for transmitting *X* messages and ensures the message's authenticity and integrity. Signing the hash of the message instead of the message can reduce the *Evidence* size because the smaller-size $H(M_X)$ is attached to the *Evidence* instead of M_X. Before relaying the packet, each intermediate node verifies the signature to ensure the message's authenticity and integrity and verifies R and *X* to secure the payment. Each node stores only the last signature for composing the *Evidence*, which is enough to prove transmitting *X* messages, e.g., after receiving the *X*th data packet, the nodes should store $Sig_S(R, X, Ts, H(M_X))$ and remove $Sig_S(R, X-1, Ts, H(M_{X-1}))$, and so on. The data transmission ends when the source node transmits its last message or if the route is broken, e.g., due to node mobility or channel impairment. Algorithm 1 gives the pseudo code of the phases of data transmission and composition of *Evidence* and report.

After receiving the *X*th data packet, Fig. 3 shows that the destination node sends back an ACK packet containing the pre-image of the last sent hash value (or $h^{(X)}$) to acknowledge receiving the message M_X, where $h^{(1)}$ is released in the first ACK and $h^{(2)}$ in the second and so on. Each intermediate node verifies the hash value by making sure that $h^{(X-1)}$ is obtained from hashing $h^{(X)}$. The nodes store only the last released hash value for composing the *Evidence*. The possession of $h^{(X)}$ by a node is a proof of delivering *X* messages but the possession of $Sig_S(R, X, Ts, H(M_X))$ is a proof of delivering *X*-1 messages and receiving one. The number of delivered messages can be computed from the number of hashing operations to map $h^{(X)}$ to $h^{(0)}$, and the number of transmitted messages (*X*) is signed by the source node. An intermediate node cannot drop the *X*th data packet and claim delivering it because the hash function is one way, i.e., it is computationally infeasible to compute $h^{(X)}$ from $h^{(X-1)}$. Hash chains have been used for many purposes due to their low energy and computation overhead and non-repudiation and one-way properties. In RACE, hash chains are used to reduce the number of public key cryptography operations, i.e., instead of generating a signature per ACK packet to secure the payment, one signature is generated by the destination node per K ACK packets.

If a node in the route does not receive a data or ACK packet within a time interval, the session is considered stale. The node A can estimate this interval as $n_A \times$ (cryptographic-delay + transmission-delay), where n_A is the number of nodes between A and the source node for data packets and the destination node for ACK packets. The cryptographic-delay is the maximum computation time required by a node to perform the cryptographic operations, and the transmission-delay includes any other delays, such as the propagation, queuing, and channel contention delays.

***Evidence* composition:** *Evidence* is defined as information that is used to establish proof about the occurrence of an event or action, the time of occurrence, the parties involved in the event, and the amount of payment. The purpose of an *Evidence* is to resolve a dispute about the amount of the payment resulted from data transmission. Fig. 4 gives the general format of an *Evidence*. The figure shows that an *Evidence* contains two main parts called DATA and PROOF. The DATA contains the necessary data to regenerate the nodes' signatures and describes the payment, i.e., who pays whom and how much. From Fig. 4, the DATA contains the identities of the nodes in the route (R), the number of received messages (*X*), the session establishment time stamp (Ts), the root of the destination node's hash chain $h^{(0)}$, the hash value of the last message ($H(M_X)$), and the last released hash value ($h^{(V)}$). V = *X*-1 when the last received packet is the *X*th data packet because the session is broken before receiving the *X*th ACK packet that contains $h^{(X)}$, but V = *X* when the last received packet is the *X*th ACK packet. The DATA does not have $h^{(1)}$ when the route is broken after receiving the first data packet because the ACK that has $h^{(1)}$ is not received. The PROOF is an undeniable security token that can prove the correctness of the DATA and protect against payment manipulation, forgery, and repudiation. The PROOF is composed by hashing the destination node's signature and the last signature received from the source node, instead of attaching the signatures to reduce the *Evidence* size.

*Evidence*s have the following main features:

1. *Evidence*s are unmodifiable: If *X* messages are delivered, the intermediate nodes can compose *Evidence*s for fewer than *X* messages, but not for more. This is because the intermediate nodes have $Sig_S(R, i, Ts, H(M_i))$ and $h^{(i)}$ for i = {1, 2, .. , *X*}, which are sufficient for composing *Evidence*s for fewer than *X* messages. However, the intermediate nodes cannot compose *Evidence*s for more than *X* because it is computationally infeasible to compute $Sig_S(R, i, Ts, H(M_i))$ or $h^{(i)}$ for i > *X*.
2. If the source and destination nodes collude, they can create *Evidences* for any number of messages because they can compute the necessary security tokens.
3. *Evidence*s are unforgeable: If the source and destination nodes collude, they can create *Evidence* for sessions that did not happen, but the intermediate nodes cannot, because forging the source and destination nodes' signatures is impossible.
4. *Evidence*s are undeniable: This is necessary to enable the TP to verify them to secure the payment. A source node cannot deny initiating a session or the amount of payment because it signs the number of transmitted messages and the signature is included in the *Evidence*.
5. An honest intermediate node can always compose valid *Evidence* even if the route is broken or the other nodes in the route collude to manipulate the payment of the node. This is because it can verify the *Evidence*s to avoid being fooled by the attackers.

Reducing the storage area of the *Evidence*s is important because they should be stored until the AC clears the payment. Onion hashing technique can be used to aggregate *Evidence*s. The underlying idea is that instead of storing one PROOF per session, one compact PROOF can be computed to prove the credibility of the payment of a group of sessions. The compact *Evidence* contains the concatenation of the DATAs of the individual *Evidence*s and one compact PROOF that is computed by onion hashing the PROOFs of the individual *Evidence*s. Let PROOF(i) refer to the PROOF of the *Evidence* number i, the compact PROOF is computed as follows:

H(…., H(H(PROOF(1), PROOF(2)), PROOF(3)), … , PROOF(n))

PROOF(1) and PROOF(2) are concatenated and hashed together, and then PROOF(3) is added to the compact PROOF by adding one hashing layer and so on. The compact PROOF has the same size of the PROOF of individual *Evidence*, but it can prove the credibility of the payment of multiple sessions. The onion hashing technique enables the nodes to aggregate a recent *Evidence* with the old compact *Evidence*, i.e., *Evidence*s are always stored in aggregated format to reduce their storage area. The technique is called onion hashing because each aggregation operation requires adding one hashing layer.

However, the *Evidence* aggregation process is irreversible because the hash function is unidirectional, i.e., the compact *Evidence* cannot be decomposed to individual *Evidence*s. Thus, if the TP requests an *Evidence* that is aggregated in the compact *Evidence*, the node has to submit the compact *Evidence* and the TP has to verify all the PROOFs of the sessions of the compact *Evidence*, instead of verifying only the PROOF of the requested *Evidence*. Therefore, aggregating more *Evidence*s can further reduce their storage area, but with more communication and processing overhead if an *Evidence* is requested. This is acceptable because *Evidence*s are requested only in case of cheating and RACE requests the *Evidence*s from few nodes instead of all the nodes in the cheating reports. The aggregation level can be flexible and dependent on the available memory space, e.g., a storage-constrained node can aggregate all *Evidence*s in only one compact *Evidence*.

$DATA = R, X, Ts, H(M_X), h^{(0)}, [h^{(V)}]$

$PROOF = H(Sig_S(R, X, Ts, H(M_X)), Sig_D(R, Ts, h^{(0)}))$

Fig. 4: The general format of an *Evidence*.

Table 3: Numerical examples for reports submitted by node A.

Session identifier	F	X
$\mathbf{ID_A}$, ID_W, ID_C, ID_B, Ts_1	0	12
ID_C, ID_W, ID_Y, ID_Z, $\mathbf{ID_A}$, Ts_2	1	17
ID_W, ID_Y, $\mathbf{ID_A}$, ID_Z, Ts_3	0	15
…….		

Payment report composition/submission: A payment report contains the session identifier, a flag bit (F), and the number of messages (X). The session identifier is the concatenation of the identities of the nodes in the session and the time stamp. The flag bit is zero if the last received packet is data and one if it is ACK. Table 3 gives numerical examples for the payment reports of node A. For the first report, A is the source node and claims sending 12 messages but it did not receive the ACK of

the last message because F is zero. For the second report, A is the destination node and claims receiving 17 messages. For the third report, A is an intermediate node and claims receiving 15 messages but it did not receive the ACK of the last message. Algorithm 2 gives the pseudo code of the payment clearance phase.

Algorithm 2: Submission/clearance of reports and *Evidences*

1: $n_i \rightarrow$ TP: **Submit**(Reports[t_{i-1}, t_i));
2: TP $\rightarrow n_i$: **Evidences_Request**(Ses_IDs[t_{i-2}, t_{i-1}));
3: $n_i \rightarrow$ TP: **Submit**(Req_Evs[t_{i-2}, t_{i-1}));
4: TP: **Identify_Cheaters**();
5: **if** (n_i is honest) **then**
6: TP $\rightarrow n_i$: A renewed certificate;
7: TP: Clear the payment reports;
8: **end if**

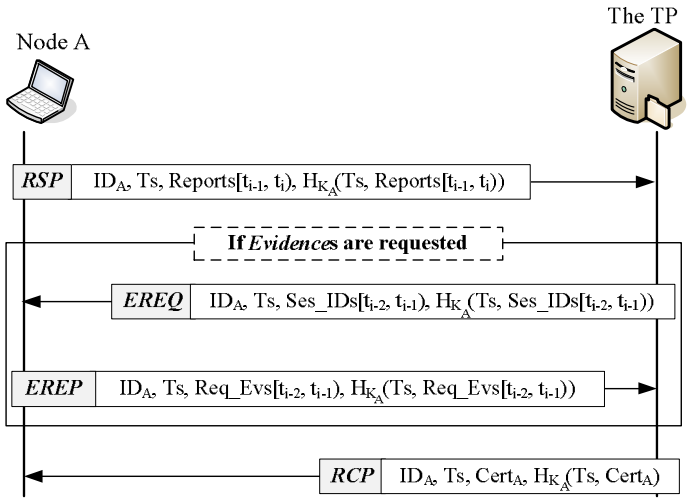

Fig. 5: The submission of reports and *Evidences*.

As shown in Fig. 5, node A sends a *Report Submission Packet* (*RSP*) to the TP at time t_i to submit the reports of the sessions held since the last contact at t_{i-1}. The packet contains the reports of the sessions held in $[t_{i-1}, t_i)$ (Reports$[t_{i-1}, t_i)$), timestamp (Ts), and a keyed hash value ($H_{KA}()$) to ensure the packet's integrity and authenticity, where K_A is the long-term symmetric key shared between node A and the TP. Thus, the TP can make sure that the packet has not been manipulated and the reports are indeed sent by the intended node, which is important to secure the payment and hold the nodes accountable for any misbehavior. If the TP requests *Evidences* from node A, it sends an *Evidences Request Packet* (*EREQ*) containing the session identifiers of the reports that their *Evidences* are requested (Ses_IDs$[t_{i-2}, t_{i-1})$). Node A replies with *Evidences Reply Packet* (*EREP*) containing the requested *Evidences* (Req_Evs$[t_{i-2}, t_{i-1})$). If node A is honest, the TP sends a *Renewed Certificate Packet* (*RCP*) containing a renewed certificate for node A with the same identity and public/private keys but with updated lifetime. Therefore, only the efficient hashing operations are used to submit the reports and *Evidences* securely to the TP. Note that *RSP* and *RCP* are also required in receipt-based payment schemes to submit receipts.

4.2 Classifier

After receiving a session's payment reports, the AC verifies them by investigating the consistency of the reports, and classifies them into fair or cheating. For fair reports, the nodes submit correct payment reports, but for cheating reports, at least one node does not submit the reports or submits incorrect reports, e.g., to steal credits or pay less. Fair reports can be for complete or broken sessions. For a complete session, all the nodes in the session report the same number of messages and F of one. If a session is broken during relaying the *X*th data packet, the reports of the nodes from S to the last node that received the packet report *X* and F of zero, but the other nodes report *X*-1 and F of one. If a session is broken during relaying the *X*th ACK packet, the nodes in the session report *X* messages, and the nodes from D to the last node that received the ACK report F of one, but the other nodes report F of zero. The reports are classified as cheating if they do not achieve one of the aforementioned rules.

Table 4 gives numerical examples for fair reports. Case 1 is reports for complete session and Cases 2 to 4 are reports for broken sessions. For Case 1, all the nodes report the same number of messages and F of one. For Case 2, the session was broken during relaying the ACK packet number 11 and B is the last node that received the packet. For Case 3, the session was broken during relaying the data packet number 8 and node A is the last node that received the packet. For Case 4, the session was broken during relaying the first data packet, and node B is the last node that received the packet and thus nodes C and D did not submit the payment report of the session.

Table 4: Numerical examples for fair reports.

Case №		S	A	B	C	D
1	X	11	11	11	11	11
	F	1	1	1	1	1
2	X	11	11	11	11	11
	F	0	0	1	1	1
3	X	8	8	7	7	7
	F	0	0	1	1	1
4	X	1	1	1	--	--
	F	0	0	0	--	--

4.3 Identifying Cheaters

As shown in Fig. 2, in the *Identifying Cheaters'* phase, the TP processes the cheating reports to identify the cheating nodes and correct the financial data. Our objective of securing the payment is to prevent the attackers (singular of collusive) from stealing credits or paying less, i.e., the attackers should not benefit from their misbehaviors. We should also guarantee that each node can earn the correct payment even if the other nodes in the session collude to steal credits. The AC requests the *Evidence* only from the node that submits report with more payment instead of all the nodes in the session because it should have the necessary and undeniable proofs (signatures and hash chain elements) for identifying the cheating nodes. In this way, the AC can precisely identify the cheating nodes with requesting few *Evidences*, i.e., one *Evidence* per session. Numerical examples will be given in Section 5 to clarify how cheating nodes can be identified without false accusations.

To verify an *Evidence*, the TP composes the PROOF by ge-

nerating the nodes' signatures and hashing them, and thus the *Evidence* is valid if the computed PROOF is similar to the *Evidence*'s PROOF.

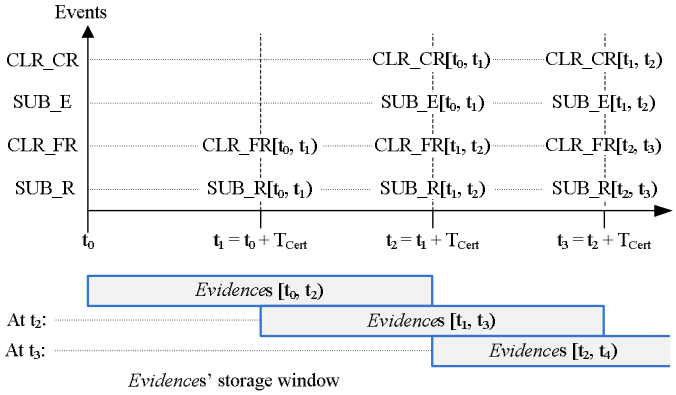

Fig. 6: The worst-case timing of the reports submission and clearance.

4.4 Credit-Account Update

As shown in Fig. 2, the *Credit-Account Update* phase receives fair and corrected payment reports to update the nodes' credit accounts. The payment reports are cleared according to the charging and rewarding policy discussed in Section 3.

In receipt-based payment schemes, a receipt can be cleared once it is submitted because it carries undeniable security proof, but the AC in RACE has to wait until receiving the reports of all nodes in a session to verify the payment. The maximum payment clearance delay (or the worst-case timing) occurs for the sessions that are held shortly after at least one node contacts the AC and the node submits the report after the certificate lifetime (T_{Cert}), i.e., at least one report is submitted after T_{Cert} of the session occurrence. It is worth to note that the maximum time duration for a node's two consecutive contacts with the TP is T_{Cert} to renew its certificate to be able to use the network.

Fig. 6 shows the worst-case timing of the submission and clearance of the reports with considering that the reports are submitted every T_{Cert}, where SUB_R, SUB_E, CLR_FR, and CLR_CR refer to the events of submitting reports, submitting *Evidence*s, clearing fair reports, and clearing cheating reports. At t_1, the nodes submit the payment reports of the sessions held in $[t_0, t_1)$ and the fair reports of these sessions are cleared. Thus, the maximum payment clearance delay of fair reports is T_{Cert} for the sessions held shortly after t_0, but the average payment clearance delay is $T_{Cert}/2$ for the sessions held in $[t_0, t_1)$ assuming that the sessions are held according to uniform random distribution. At t_2, the TP requests the *Evidence*s of the cheating reports of the sessions held in $[t_0, t_1)$. Thus, the maximum payment clearance delay for cheating reports is $2\times T_{Cert}$ for the sessions held shortly after t_0, but the average payment clearance delay is 1.5 T_{Cert} for the cheating reports of the sessions held in $[t_0, t_1)$. The figure also shows that the maximum time for storing an *Evidence* is $2\times T_{Cert}$, e.g., for the reports of sessions held shortly after t_0. At t_2, the nodes delete the *Evidence*s of the sessions held in $[t_0, t_1)$ because the AC must have cleared their reports.

However, the nodes submit the reports at different times because the connection to the TP may not be available on a regular basis, and thus the duration between each two submissions may not be the same and may be less than or equal to T_{Cert}. Hence, the maximum payment clearance delay may be less than T_{Cert}. T_i is a continuous random variable that denotes the time duration between two submissions for a node, where $T_i \in [0, T_{Cert}]$ and the submission durations of the nodes are independent and identically distributed (i.i.d.) random variables. In order to estimate the average and maximum payment clearance delay, we consider two models. For model I, each node contacts the TP when it accumulates a large number of reports or when the remaining time of its certificate's lifetime is short to reduce the communication overhead. We model this behavior with truncated exponential distribution given in Eq. 1, where the probability that a node contacts the TP is high as T_i approaches T_{Cert}. For model II, a node submits the reports once it has a connection to the TP and the connections are uniformly distributed over the time interval $[0, T_{Cert}]$.

Eqs. 2 and 3 give the probabilities of submitting the reports at most by time t for models I and II, respectively. The payment of a session is cleared when all the nodes in the session submit their reports. T(n) is a continuous random variable that denotes the time duration for n nodes to contact the AC, where $T(n) \in [0, T_{Cert}]$. Eq. 4 gives the probability that T(n) is less than or equal to t. The probability density functions of T(n) using models I and II are given in Eqs. 5 and 6, respectively. Eq. 7 gives the average time duration for n nodes to contact the AC, which is equivalent to the maximum payment clearance delay for a session. Eq. 8 gives the average payment clearance delay for a session with n nodes assuming that sessions are held according to uniform distribution.

$$f(T_i) = \frac{\lambda \cdot e^{-\lambda \cdot t}}{1 - e^{-\lambda \cdot T_{Cert}}} \quad (1)$$

$$P(T_i \le t) = \frac{1 - e^{-\lambda \cdot t}}{1 - e^{-\lambda \cdot T_{Cert}}} \quad (2)$$

$$P(T_i \le t) = \frac{t}{T_{Cert}} \quad (3)$$

$$P(T(n) \le t) = \prod_{i=1}^{i=n} P(T_i \le t) \quad (4)$$

$$f(T(n)) = e^{-\lambda \cdot t} \cdot \lambda \cdot n \cdot \frac{(1 - e^{-\lambda \cdot t})^{n-1}}{(1 - e^{-\lambda \cdot T_{Cert}})^{n}} \quad (5)$$

$$f(T(n)) = \frac{n}{T_{Cert}} \cdot \left(\frac{t}{T_{Cert}}\right)^{n-1} \quad (6)$$

$$E(T(n)) = \int_0^{T_{Cert}} t \cdot f(T(n))\, dt \quad (7)$$

$$P_C(n) = \frac{E(T(n))}{2} \quad (8)$$

$$E(T_i) = \frac{1}{\lambda} \cdot \left[\frac{1 - (\lambda \cdot T_{Cert} + 1) \cdot e^{-\lambda \cdot T_{Cert}}}{1 - e^{-\lambda \cdot T_{Cert}}}\right] \quad (9)$$

We have selected λ to be 1/7 and T_{Cert} to be 10, 15, and 20 days, to make the average time duration between two submissions by a node (given in Eq. 9) to be 3.9, 4.9, and 5.73 days, respectively. Figs. 7 and 8 show the probabilities that the payment of a session with n nodes is cleared at most by time t for models I and II, respectively, with T_{Cert} of 15 days and λ of 1/7. The figures show that the increase of the route length (n) de-

creases the probability of clearing the payment by time t because the AC has to wait more time to receive the reports from more nodes. Moreover, the maximum payment clearance delay is less than T_{Cert} with high probability. Figs. 9 and 10 show the average payment clearance delay at different values of n for models I and II, respectively. It can be seen that the average payment clearance delay can be much less than T_{Cert}. For example, at n = 5 nodes and T_{Cert} = 15 days, the average payment clearance delay is 5 and 6.2 for models I and II, respectively. It can also be seen that shortening T_{Cert} can decrease the payment clearance delay.

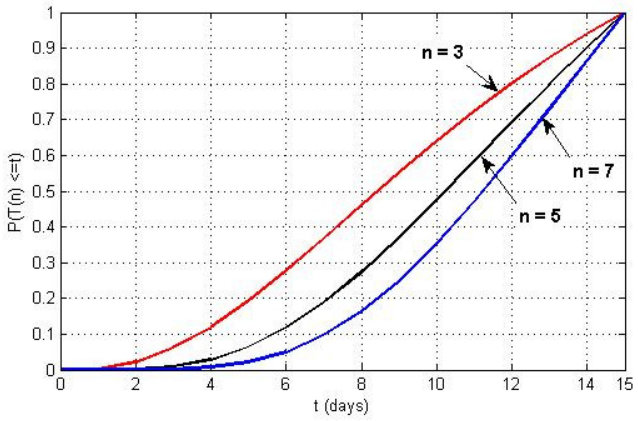

Fig. 7: $P(T(n) \leq t)$ Vs t at different values of n for model I.

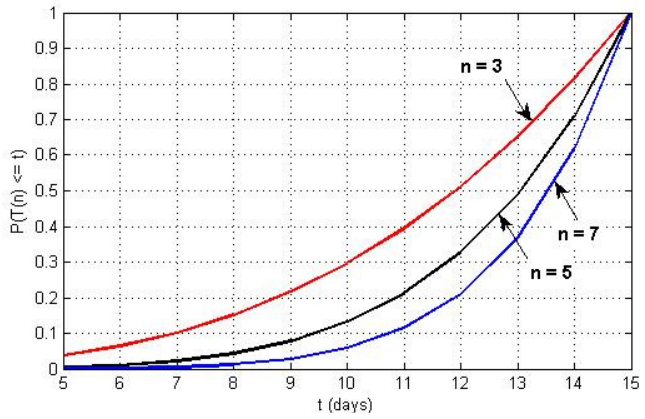

Fig. 8: $P(T(n) \leq t)$ Vs t at different values of n for model II.

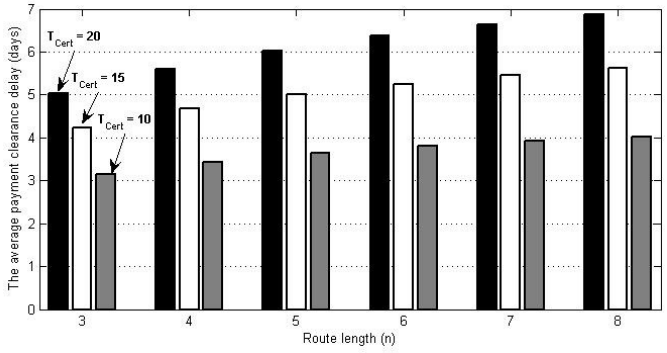

Fig. 9: The average payment clearance delay for model I.

Although our analysis demonstrates that the payment clearance delay can be less than T_{Cert} when the nodes submit their reports according to truncated exponential and uniform distributions, this delay is bounded by T_{Cert} and acceptable due to using post-paid payment model where the nodes communicate first and pay later, i.e., the nodes do not need to wait until clearing the payment to communicate. Moreover, the nodes can purchase credits with real money.

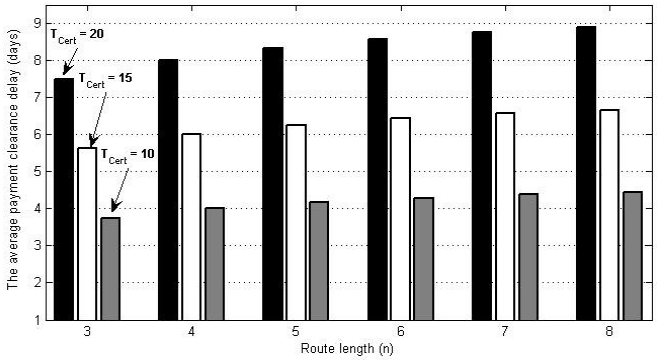

Fig. 10: The average payment clearance delay for model II.

5. Security Analysis

Our security objective is to prevent an attacker or even a group of colluding attackers from achieving gains such as stealing credits or paying less. The signatures of the source node can ensure the messages' integrity and authenticity and secure the payment. Signatures and hash chains have non-repudiation property because it is computationally infeasible to compute a node's signature without knowing the private key and to compute $h^{(i)}$ from $h^{(i-1)}$. This non-repudiation property is used to secure the payment by enabling the nodes to compose valid *Evidences* and enabling the TP to verify the *Evidences* to identify the cheating nodes.

In order to evaluate the identifying cheaters' process, numerical examples for cheating reports are given in Table 5. In Cases 1 and 2, the reports of the intermediate and destination nodes are consistent but the source node claims sending fewer number of messages. The source node can compose valid *Evidence* if it cheats because it has the tokens $Sig_S(R, 6, Ts, H(M_6))$ and $h^{(6)}$, and thus it is not effective to request the *Evidence* from the source node. The TP can request the *Evidence* from an intermediate node or the destination node. The source node is a cheater if the *Evidence* is correct because the *Evidence* cannot be composed without the source node's signature for ten messages and the intermediate and destination nodes cannot compute this signature. For Case 2, it is obvious that the route was broken at node B during relaying the data packet number ten. For Case 3, the source and destination nodes' reports are consistent but the intermediate nodes claim relaying more messages. If an intermediate node submits valid *Evidence*, the source and destination nodes are cheaters because the *Evidence* should contain the source node's signature for twelve messages and $h^{(11)}$ or $h^{(12)}$. It is not effective to request the *Evidence* from the source or the destination node because they can collude to generate valid *Evidence*.

In Case 4, the reports of the intermediate and destination nodes are consistent but the source node claims sending more messages. This case may be rare because the rational attackers will attempt to steal credits or pay less. The TP can clear the payment according to the nodes' reports without requesting *Evidences* to achieve our security strategy and discourage submitting incorrect reports because the source node pays more if it lies and the other nodes lose credits if they lie. How-

ever, the TP can identify the cheating nodes by requesting the *Evidence* from the source node because it should contain $h^{(12)}$ or $h^{(11)}$. If the *Evidence* is correct, the destination node is evicted but the intermediate nodes should not be evicted because eight messages may be indeed relayed in the session and the source and destination nodes collude to falsely accuse the intermediate nodes. Case 5 is similar to Case 4 but the source and destination nodes report the same number of messages. The payment is cleared according to the nodes' reports to punish the nodes that submit incorrect reports without stealing credits.

In Cases 6 and 7, node B can prove the credibility of its reports and earn the deserved payment even if the other nodes in the session collude. For Case 7, node B claims delivering seven messages but the other nodes claim receiving seven messages and delivering only six messages. If node B is honest, its *Evidence* should have $h^{(7)}$. If the *Evidence*s of node B are valid, the source and destination nodes are cheaters in Case 6 but only the destination node is a cheater in Case 7. For Case 8, as long as the destination node acknowledges receiving the message number seven, the intermediate nodes are rewarded for seven messages. In Cases 9 and 10, "--" means that the node did not submit the payment report of the session. For Case 9, if node A submits valid *Evidence*, the source and destination nodes are cheaters because they established a session but did not submit the payment reports. The nodes B and C are not rewarded to discourage un-submitting the payment reports. For Case 10, the source node is charged but the intermediate nodes are not rewarded without requesting *Evidence*s in order to punish the nodes that do not submit payment reports.

Table 5: Numerical examples for cheating reports.

Case №		S	A	B	C	D
1	X	6	10	10	10	10
	F	1/0	1	1	1	1
2	X	6	10	10	9	9
	F	1/0	0	0	1	1
3	X	5	12	12	12	5
	F	1	1/0	1/0	1/0	1
4	X	12	8	8	8	8
	F	1/0	1	1	1	1
5	X	9	4	4	4	9
	F	1/0	1	1	1	1/0
6	X	14	14	22	14	14
	F	1	1	1/0	1	1
7	X	7	7	7	7	6
	F	0	0	1	0	1
8	X	7	7	7	7	7
	F	0	0	1	0	1
9	X	--	4	--	--	--
	F	--	1/0	--	--	--
10	X	6	--	--	--	6
	F	1/0	--	--	--	1/0

The attackers may launch *Multiple-Redemptions and One-Redemption* attacks to deceive the AC. For *Multiple-Redemptions* attack, the attackers submit the same payment reports multiple times to be rewarded several times for the same sessions. The AC can thwart the attack and identify the attackers because it can ensure whether the payment of a session has been cleared before using the session unique identifier that includes the identities of the nodes in the session and time stamp. For *One-Redemption* attack, the attackers attempt to make the payment reports of different sessions have the same identifier to pay once because the AC clears the reports with the same identifier once. This attack is not possible in RACE because a session's identifier changes once the route or the time changes, i.e., the reports are different even if the same nodes participate in different sessions at different times.

Evidence Forgery and Manipulation attacks target the *Evidence* composition process. For *Evidence-Forgery* attack, the attackers attempt to forge *Evidence*s for sessions that did not happen to steal credits, and for *Evidence-Manipulation* attack, the attackers attempt to manipulate valid *Evidence*s to increase their rewards. In RACE, *Evidence*s are undeniable, unforgable, and unmodifiable. The source node cannot deny initiating a session and the amount of payment because its signature is included in the *Evidence*. The attackers cannot forge *Evidence*s or manipulate them with using secure hash function and public-key cryptosystem because it is impossible to compute $h^{(i)}$ from $h^{(i-1)}$ or compute the nodes' signatures without knowing the private keys. Moreover, it is also impossible to modify the source nodes' signatures, compute the private keys from the public ones, and compute the hash value of the signatures without computing the signatures. The TP can identify the attackers that forge *Evidence*s because the *Evidence*s' verifications fail.

The attackers may launch *Impersonation, Packet-Replay, and Free-Riding* attacks to target route establishment, report submission, and data transmission processes. For *Impersonation* attack, the attackers impersonate legitimate nodes to communicate freely or steal credits. This attack is impossible because the nodes use their private keys in signing the packets and their secret symmetric keys in submitting the payment reports. In *Packet-Replay* attack, the attackers record valid packets and replay them in different place and/or time to establish sessions under the name of others to communicate freely. In RACE, stale packets cannot be used to establish sessions because time stamps are used to verify the freshness of the packets. For *Free-Riding* attack, two colluding intermediate nodes in a legitimate session attempt to communicate freely by manipulating the packets to add their data. This is impossible in RACE because the integrity of the packets can be verified at each node, and thus the first intermediate node after the attacker can detect any addition or modification to the packets and thwart the attack by dropping them. The integrity of the data packets can be ensured by verifying the source nodes' signatures and the integrity of the ACK packets can be ensured by verifying the hash chain elements.

The attackers may attempt to manipulate their reports to launch *Reduced-Payment* and *False-Accusation* attacks. For *Reduced-Payment* attack, some intermediate nodes may collude with the source node to submit reports with less payment to charge the source node less. For example, if ξ intermediate nodes launch this attack successfully in a session with n nodes, the colluders can save $((n-2-\xi) \times (X-\omega) \times \lambda)$ credits, where X and ω are the correct and the submitted number of messages, respectively, and thus the source node can compensate the colluding intermediate nodes. In RACE, even if a group of nodes

colludes to reduce the rewards of an honest node, the honest node can compose valid *Evidence* and get its correct payment, such as Cases 6 and 7 in Table 5. For *False-Accusation* attack, the attackers manipulate their reports to insert non-existent nodes in a session to let the TP accuse them of not reporting the session. In RACE, if the victim nodes are intermediate, the payment is cleared without punishing or rewarding them as discussed in Cases 9 and 10 in Table 5, but if the victim nodes are source or destination, the attackers are evicted because they cannot submit correct *Evidence*s.

The charging and rewarding policy can counteract rational cheating actions. If the nodes are charged only for the successfully delivered messages, the destination nodes may collude with the source nodes to not send ACK packets so as not to pay. To prevent this, the source nodes are charged for undelivered messages. If the intermediate nodes are rewarded for the relayed messages that do not reach the destination, the colluding intermediate nodes can increase their rewards with consuming low resources by relaying only the smaller-size signatures but not the messages to compose valid *Evidence*s and claim relaying the messages. To prevent this, the intermediate nodes are rewarded only for delivered messages.

6. Performance Evaluation

Public-key cryptography is widely used to secure the wireless networks [25-27]. Using public-key cryptography in RACE is necessary to secure the payment because it enables the nodes to compose valid *Evidences* and enables the TP to identify the cheating nodes. Public-key cryptography technology and hardware implementation have been improved, and the signing and verifying operations can be performed by mobile nodes with acceptable overhead. In [28], digital signatures can be computed efficiently in two steps. The offline step is independent of the message and performed before the message to be signed is available; and a lightweight online step is performed once the message to be signed becomes available. In [29], FPGA implementation of the RSA cryptosystem can efficiently perform the signing and verifying operations in several milliseconds. Moreover, the proposed communication protocol in [17] that transfers messages from the source to the destination nodes with limited number of public-key-cryptography operations can be integrated with RACE, but the focus of this paper is on reducing the communication and the payment processing overhead.

6.1 Simulation Setup

We run a simulator to evaluate the overhead of RACE. 50 mobile nodes with 150 m transmission range are deployed in a square cell of 1200 m by 1200 m. Constant-bit-rate traffic source is implemented in each node as an application layer and the source and destination pairs are randomly chosen. We use the modified random waypoint model [30] to emulate the nodes' mobility. Specifically, a node travels towards a random destination that is uniformly selected within the network field; upon reaching the destination, it pauses for some time; and the process repeats itself afterwards. The nodes' speed is uniformly distributed in the range $[0, S_{max}]$ m/s, where S_{max} is 5 and 10 m/s, and the pause time is 20 s. The data packets are transmitted with the rate of 0.5 packet/s. We simulate the Dynamic Source Routing protocol (DSR) [22]. The time stamp (Ts), node's identity (ID_i), and message number (X) are five, four, and two bytes, respectively, and the hash chain size is 35. The simulation results are averaged over 200 runs and presented with 95 percent confidence interval. Table 6 summarizes the simulation parameters.

In the simulation, we consider 1024-bit RSA digital signature scheme [31] because the verifying operations performed by the intermediate and destination nodes require less time than the signing operations performed by the source node. According to NIST guidelines [32], the secure private keys should have at least 1024 bits. For the hash function, we use SHA-1 [33] with digest size of 20 bytes. We evaluate the expected processing delay due to performing the cryptographic operations by the mobile nodes using Crypto++5 library [34] and a laptop with an Intel processor at 1.2 GHZ and 1 GB RAM. The computation times of signing and verifying operations are 15.63 ms and 0.53 ms, respectively, and the computation time of hashing a 512-byte message is 29 μs. The resource of a real mobile node may be less than a laptop so the measured computation times are scaled by the factor of five and considered as delays to simulate performing the cryptographic operations in a limited-resource node. From [35], the consumed energy for signing and verifying operations are 546.5 mJ and 15.97 mJ, respectively, and the consumed energy for hashing a 512-byte message is 389.12 μJ.

With these network parameters, the simulated network is well connected and the route length is acceptable. The statistics of the simulated network demonstrate that the probability that a route has fewer than seven nodes is 0.89, the average route length is 4.21, and the network connectivity is 0.98. The network connectivity is the percentage of connected routes to the total number of possible routes, assuming any two nodes can be source and destination pair.

Table 6: Simulation parameters.

Parameter	Value
Data transmission rate	0.5 packet/s
Mobility model	Random waypoint model
Network dimension	1200 m by 1200 m
Node speed	$[0, S_{max}]$ m/s and S_{max} is 5 and 10 m/s
Number of nodes	50
Pause time	20 s
Radio transmission range	150 m
Routing protocol	DSR
Traffic source	Constant bit rate

6.2 Storage Overhead

The sizes of receipts, payment reports, and *Evidences* depend on the number of intermediate nodes because the nodes' identities are attached to them. Thus, changing the network parameters such as the network size, the nodes' radio transmission range and density, etc. will change the route length and have the same effect on RACE and receipt-based schemes. Table 7 gives the average size of receipt, report, and *Evidence* for RACE and receipt-based payment schemes. The receipt size of ESIP is larger than that of PIS due to attaching two hash values from the source node's hash chain and another two hash values from the destination node's hash chain. For Sprite, ESIP, PIS, and RACE, 1MB storage area can store up to 3531,

7282, 16425, and 10082 receipts and *Evidence*s, respectively. Although PIS requires low storage area, it needs two signatures per message, i.e., one from the source node and another from the destination node.

Table 7: Average sizes of receipts, report, and *Evidence* (bytes).

	Sprite	ESIP	PIS	RACE	
				Report	*Evidence*
Upper limit	307.692	150.048	66.432	24.627	107.432
Mean	297	144	64	23.84	104
Lower limit	286.308	137.952	61.568	23.053	100.568

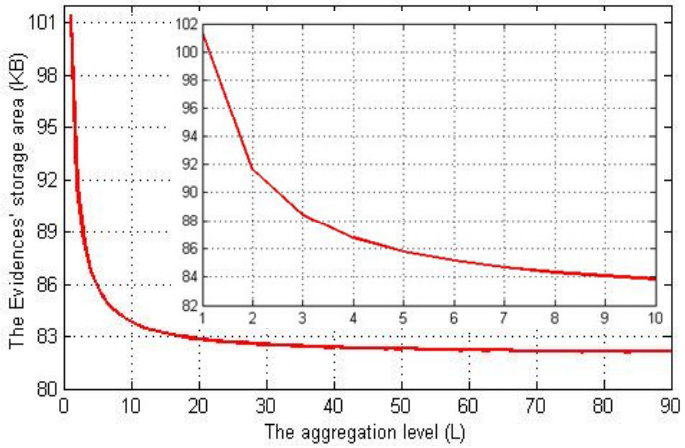

Fig. 11: The average storage area at different aggregation levels.

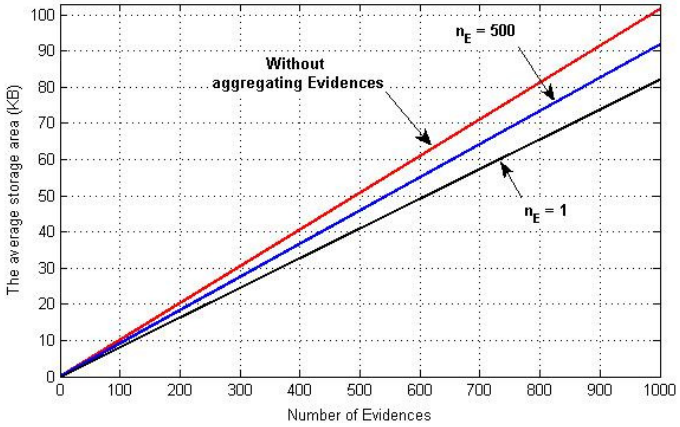

Fig. 12: The average storage area VS the number of *Evidence*s.

Fig. 11 shows the average *Evidence*s' storage area VS the aggregation level (L) for 1000 *Evidences*. The internal plot shows the average storage area at small aggregation level ($L \leq 10$), but the external plot shows the average storage area at large aggregation levels up to 90. The aggregation level i means that the 1000 *Evidence*s are stored in 1000/i compact *Evidence*s. The figure demonstrates that the increase of L over 10 has little effect on the storage area but increases the number of redundant *Evidence*s that are submitted if an *Evidence* is requested. For example, if L is two, 500 aggregated *Evidence*s are stored and each one can prove the payment for two reports, so if an *Evidence* is requested, the node submits a compact *Evidence* for two reports. Similarly, if L is 1000, all the *Evidence*s are aggregated in only one compact *Evidence*, and thus if an *Evidence* is requested, the node submits a compact *Evidence* for 1000 reports.

Fig. 12 gives the required storage area to store *Evidence*s VS the number of *Evidence*s without aggregation and with aggregating the *Evidence*s in n_E compact *Evidence*s. Without aggregation, the *Evidence*s occupy larger storage area, and the storage area can be reduced when all the *Evidence*s are aggregated in one compact *Evidence*.

Several measures have been taken to reduce the *Evidences'* size in RACE. One *Evidence* is composed per session regardless of the number of messages instead of generating an *Evidence* per message. The nodes' signatures are hashed to reduce the PROOF size and different *Evidence*s can be aggregated to a smaller-size compact *Evidence*. Moreover, the nodes of MWN are typically equipped with limited energy supplies and the network is characterized with limited bandwidth, and it is feasible to build cost-effective nodes with more than a gigabyte of flash memory [36], [37]. Therefore, storage area may not be the main concern, but bandwidth and energy are more scarce, i.e., reducing the amount of submitted data is more important than the size of stored data. As we will discuss in Subsection 6.3, the amount of submitted reports in RACE is much less than that of receipt-based payment schemes.

Table 8: The average amount of data to submit receipts and reports (KB).

Node speed \ Scheme		Sprite	ESIP	PIS	RACE
[0, 5] m/s	**Upper limit**	90.511	1.4239	0.549	0.2058
	Mean	87.79	1.36	0.53	0.2
	Lower limit	85.069	1.2961	0.511	0.1942
[0, 10] m/s	**Upper limit**	91.354	1.8443	0.818	0.3027
	Mean	88.18	1.77	0.788	0.293
	Lower limit	85.006	1.6957	0.758	0.2833

6.3 Communication Overhead

The communication overhead depends on the number and the size of receipts and reports. The number of receipts and reports generated in a session depends on the frequency of breaking the route between the source and destination nodes because a new receipt/report is generated when the route is broken, and therefore, the MAC layer and the simulation parameters will have the same effect on RACE and the receipt-based payment schemes. From Table 7, RACE requires submitting only 23.84 bytes for each payment report. This amount of data is much less than those of the existing receipt-based payment schemes because security tokens, e.g., signatures, are always submitted in receipt-based schemes but they are submitted only in case of cheating in RACE. Even if there are many cheaters in the network, RACE requires less communication overhead than the existing receipt-based schemes because the cheaters are excluded once they commit a cheating action. *A 512 KB data transmission is sufficient for submitting 1765, 3641, and 8192 receipts in Sprite, ESIP, and PIS, respectively, and submitting 21,992 reports in RACE.*

Table 8 gives the average amount of data to submit receipts and reports for ten-minute data transmission. The source and destination nodes are randomly selected and a new route is established each time the route between the source and destination nodes is broken. It can be seen that a large amount of data is submitted in Sprite because a receipt is generated per message and the receipt size is large. PIS requires submitting less

amount of data than ESIP because its receipts' size is less as indicated in Table 7, but PIS requires two signatures for transmitting a message. The amount of data to submit reports in RACE is much less than those of the receipt-based schemes. Table 8 indicates that more reports and receipts are submitted at high node mobility because the routes are more frequently broken, i.e., the source node's messages are transmitted over a larger number of routes.

Table 9: The required cryptographic operations for clearing the payment of ten-minute data transmission with S_{max} of 5 m/s.

Scheme		Number of signing‖ hashing‖ verifying operations	Energy Consumption (mJ)	Processing Time (ms)
Sprite	Upper limit	0‖0‖628.41	10035.77	333.06
	Mean	0‖0‖605.7	9673.03	321.02
	Lower limit	0‖0‖582.99	9310.29	308.98
ESIP	Upper limit	17.58‖315.39‖0	9732.53	283.99
	Mean	17.1‖306.7‖0	9464.49	276.17
	Lower limit	16.62‖298.01‖0	9196.46	268.35
PIS	Upper limit	17.84‖0‖0	9748.86	278.82
	Mean	17.1‖0‖0	9345.15	267.27
	Lower limit	16.36‖0‖0	8941.44	255.73
RACE (fair reports)		0‖0‖0	0	0
RACE (a cheating report)		2‖X+1‖0	1093 + 0.39 (X+1)	31.26 + 0.029 (X+1)

Table 10: The required cryptographic operations for clearing the payment of ten-minute data transmission with S_{max} of 10 m/s.

Scheme		Number of signing‖ hashing‖ verifying operations	Energy Consumption (mJ)	Processing Time (ms)
Sprite	Upper limit	0‖0‖628.66	10039.7	333.19
	Mean	0‖0‖608.4	9716.15	322.45
	Lower limit	0‖0‖588.14	9392.6	311.71
ESIP	Upper limit	26.13‖320.88‖0	14407.59	417.79
	Mean	25.2‖309.4‖0	13892.19	402.85
	Lower limit	24.27‖297.92‖0	13376.79	387.90
PIS	Upper limit	26.24‖0‖0	14337.82	410.06
	Mean	25.2‖0‖0	13771.8	393.88
	Lower limit	24.16‖0‖0	13205.78	377.69
RACE (fair reports)		0‖0‖0	0	0
RACE (a cheating report)		2‖X+1‖0	1093 + 0.39 (X+1)	1093 + 0.39 (X+1)

6.4 Payment Processing Overhead

Tables 9 and 10 give the processing overhead for clearing the payment of ten-minute data transmission at different node speed in terms of the number of cryptographic operations, the total energy cost, and the processing time, assuming that the TP is a laptop with an Intel processor at 1.2 GHZ and 1 GB RAM. The tables indicate that RACE does not need any cryptographic operations for clearing the payment in case of fair reports. The tables also give the overhead of verifying an *Evidence* with *X* messages. The simulation results indicate that the payment clearance overhead of RACE is much less than the existing receipt-based payment schemes. It can also be seen that more overhead is required at high node mobility because more receipts are generated due to breaking the routes more frequently, which shows that receipt-based payment schemes may not be efficiently applicable in case of high node mobility, but *the nodes' speed has no effect on the payment clearance overhead in RACE if the reports are fair.*

The low payment processing overhead can reduce the complexity and provide flexibility to the practical implementation of the TP. Moreover, since the payment schemes use micropayment, the overhead cost should be much less than the payment for the effective implementation of these schemes. The communication and processing overhead of the receipts will be very large with taking into account the following facts: (1) the simulation results given in Tables 8, 9, and 10 are only for ten-minute data transmission; (2) the nodes contact the TP every few days because this connection may not be available on a regular basis and to reduce the communication overhead; and (3) once a route is broken, a new route is established with a new receipt, and thus multiple receipts may be generated per session.

RACE can significantly reduce the overhead of submitting the payment reports and clear the payment with almost no cryptographic operations or processing overhead when cheating actions are not frequent. Widespread cheating is not expected in civilian applications because the common users do not have the technical knowledge to tamper with their devices and the manufacturing companies (which are limited) cannot sacrifice their reputation and face liability for making tampered devices. Moreover, a cheating node is evicted once it commits one cheating action, and changing identity is not easy or cheap, e.g., the TP can impose fees for issuing new certificates.

7. Conclusion and Future Work

In this paper, we have proposed a report-based payment scheme for MWNs. The nodes submit lightweight payment reports containing the alleged charges and rewards (without proofs) and temporarily store undeniable security tokens called *Evidence*s. The fair reports can be cleared with almost no cryptographic operations or processing overhead, and *Evidence*s are submitted and processed only in case of cheating reports in order to identify the cheating nodes. Our analytical and simulation results demonstrate that our scheme can significantly reduce the communication and processing overhead comparing to the existing receipt-based payment schemes with acceptable payment clearance delay and *Evidence*s' storage area, which is necessary for the effective implementation of the scheme. Moreover, RACE can secure the payment and precisely identify the cheating nodes without false accusations.

In RACE, the AC can process the payment reports to know the number of relayed messages and the number of dropped messages by each node. In our future work, we will develop a trust system based on processing the payment reports to assign and maintain a trust value for each node in the network. The nodes that relay messages more successfully will have higher trust values, such as the low-mobility and the large-hardware-resources nodes. Based on these trust values, we will propose a trust-based routing protocol to route messages through the highly trusted nodes (which performed packet relay more successfully in the past) to minimize the probability of dropping the messages, and thus improve the network performance in terms of throughput and packet delivery ratio. However, the trust system should be secure against singular and collusive

attacks, and the routing protocol should make smart decisions regarding node selection with low overhead.

REFERENCES

[1] G. Shen, J. Liu, D. Wang, J. Wang, and S. Jin, "Multi-hop relay for next-generation wireless access networks", Bell Labs Technical Journal, vol. 13, no. 4, pp. 175-193, 2009.

[2] C. Chou, D. Wei, C. Kuo, and K. Naik, "An efficient anonymous communication protocol for peer-to-peer applications over mobile ad-hoc networks", IEEE Journal on selected areas in communications, vol. 25, no. 1, January 2007.

[3] H. Gharavi, "Multichannel mobile ad hoc links for multimedia communications", Proc. of the IEEE, vol. 96, no. 1, pp. 77-96, January 2008.

[4] S. Marti, T. Giuli, K. Lai, and M. Baker, "Mitigating routing misbehavior in mobile ad hoc networks", Proc. of ACM Mobile Computing and Networking (MobiCom'00), pp. 255–265, Boston, Massachusetts, USA, August 6-11, 2000.

[5] G. Marias, P. Georgiadis, D. Flitzanis, and K. Mandalas, "Cooperation enforcement schemes for MANETs: A survey", Wiley's Journal of Wireless Communications and Mobile Computing, vol. 6, issue 3, pp. 319–332, 2006.

[6] Y. Zhang and Y. Fang, "A secure authentication and billing architecture for wireless mesh networks", ACM Wireless Networks, vol. 13, no. 5, pp. 663–678, October 2007.

[7] 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, October 2004.

[8] 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, October, 2007.

[9] A. Weyland, "Cooperation and accounting in multi-hop cellular networks", Ph.D. thesis, University of Bern, November 2005.

[10] A. Weyland, T. Staub, and T. Braun, "Comparison of motivation-based cooperation mechanisms for hybrid wireless networks", Journal of Computer Communications, vol. 29, pp. 2661–2670, 2006.

[11] S. Zhong, J. Chen, and R. Yang, "Sprite: A simple, cheat-proof, credit based system for mobile ad-hoc networks", Proc. of IEEE INFOCOM'03, vol. 3, pp. 1987-1997, San Francisco, CA, USA, March 30-April 3, 2003.

[12] M. Mahmoud and X. Shen, "FESCIM: Fair, efficient, and secure cooperation incentive mechanism for hybrid ad hoc networks", IEEE Transactions on Mobile Computing (IEEE TMC), to appear.

[13] M. Mahmoud, and X. Shen, "PIS: A practical incentive system for multi-hop wireless networks", IEEE Transactions on Vehicular Technology (IEEE TVT), vol. 59, no. 8, pp. 4012-4025, 2010.

[14] M. Mahmoud and X. Shen, "Stimulating cooperation in multi-hop wireless networks using cheating detection system", Proc. IEEE INFOCOM'10, San Diego, California, USA, March 14-19, 2010.

[15] N. Salem, L. Buttyan, J. Hubaux, and M. Jakobsson, "Node cooperation in hybrid ad hoc networks", IEEE Transactions on Mobile Computing (IEEE TMC), vol. 5, no. 4, pp. 365-376, April 2006.

[16] J. Pan, L. Cai, X. Shen, and J. Mark, "Identity-based secure collaboration in wireless ad hoc networks", Computer Networks (Elsevier), vol. 51, no. 3, pp. 853-865, 2007.

[17] M. Mahmoud and X. Shen, "ESIP: Secure incentive protocol with limited use of public-key cryptography for multi-hop wireless networks", IEEE Transactions on Mobile Computing (IEEE TMC), vol. 10, no. 7, pp. 997-1010, July 2011.

[18] M. Mahmoud and X. Shen, "An integrated stimulation and punishment mechanism for thwarting packet drop in multihop wireless networks", IEEE Transactions on vehicular technology (IEEE TVT), vol. 60, no. 8, pp. 3947-3962, 2011.

[19] H. Zhu, X. Lin, R. Lu, Y. Fan, and X. Shen, "SMART: A secure multilayer credit based incentive scheme for delay-tolerant networks", IEEE Transactions on Vehicular Technology (IEEE TVT), vol. 58, no. 8, pp. 4628 - 4639, 2009.

[20] R. Lu, X. Lin, H. Zhu, X. Shen, and B. R. Preiss, "Pi: A practical incentive protocol for delay tolerant networks", IEEE Transactions on Wireless Communications (IEEE TWC), vol. 9, no. 4, pp. 1483-1493, 2010.

[21] B. Wehbi, A. Laouiti, and A. Cavalli, "Efficient time synchronization mechanism for wireless multi hop networks", Proc. of IEEE Personal, Indoor and Mobile Radio Communications (PIMRC), 2008.

[22] D. Johnson and D. Maltz, "Dynamic source routing in ad hoc wireless networks", Mobile Computing, Chapter 5, Kluwer Academic Publishers, pp. 153-181, 1996.

[23] L. Anderegg and S. Eidenbenz, "Ad Hoc-VCG: A trustful and cost-efficient routing protocol for mobile ad hoc networks with selfish agents", Proc. of ACM MobiCom, San Diego, CA, USA, September 2003.

[24] H. Pagnia and F. Gartner, "On the impossibility of fair exchange without a trusted third party", Technical Report TUD-BS-1999-02, Darmstadt University of Technology, March 1999.

[25] K. Sanzgiri, D. LaFlamme, B. Dahill, B. Levine, C. Shields, and E. Belding-Royer, "Authenticated routing for ad hoc networks", IEEE Selected Areas in Communications, vol. 23, no. 3, pp. 598–610, March 2005.

[26] Y. Hu, A. Perrig, and D. Johnson, "Ariadne: A secure on-demand routing protocol for ad hoc networks", Proc. of ACM MobiCom, Atlanta, GA, USA, September 2002.

[27] B. Wu, J. Chen, J. Wu, and M. Cardei, "A survey of attacks and countermeasures in mobile ad hoc networks", Wireless Network Security, Springer Network Theory and Applications, vol. 17, pp 103-135, 2007.

[28] S. Even, O. Goldreich, and S. Micali, "On-line/off-line digital signatures", Advances in Cryptology--Crypto '89, Lecture Notes in Computer Science, Springer-Verlag, Berlin, vol. 435, pp. 263-277, 1990.

[29] O. Nibouche, M. Nibouche, A. Bouridane, and A. Belatreche, "Fast architectures for FPGA-based implementation of RSA encryption algorithm", Proc. of IEEE Field-Programmable Technology conference, Brisbane, Australia, December 2004.

[30] J. Yoon, M. Liu, and B. Nobles, "Sound mobility models", Proc. of ACM MobiCom, San Diego, CA, USA, September 2003.

[31] A. Menzies, P. Oorschot, and S. Vanstone, "Handbook of applied cryptography", CRC Press, http://www.cacr.math.uwaterloo.ca/hac, Boca Raton, Fla., 1996.

[32] National Institute of Standards and Technology (NIST), "Recommendation for key management - Part 1: General (Revised)", Special Publication 800-57 200, 2007.

[33] NIST, "Digital hash standard", Federal information processing standards publication 180-1, April 1995.

[34] D. Wei, "Crypto++ Library 5.6.0", http://www.cryptopp.com, October, 2011.

[35] N. Potlapally, S. Ravi, A. Raghunathan, and N. Jha, "A study of the energy consumption characteristics of cryptographic algorithms and security protocols", IEEE Transactions on Mobile Computing (IEEE TMC), vol. 5, no. 2, pp. 128-143, March-April 2006.

[36] P. Desnoyers, D. Ganesan, H. Li, M. Li, and P. Shenoy, "PRESTO: A predictive storage architecture for sensor networks", Proc. of 10th Workshop on Hot Topics in Operating Systems (USENIX HotOS), Santa Fe, New Mexico, June 2005.

[37] A. Mitra, A. Banerjee, W. Najjar, D. Zeinalipour-Yazti, V. Kalogeraki, and D. Gunopulos, "High-performance low power sensor platforms featuring gigabyte scale storage", Proc. of IEEE/ACM 3rd International Workshop on Measurement, Modeling, and Performance Analysis of Wireless Sensor Networks (SenMetrics 2005), San Diego, CA, July 2005.