Int. J. Communications, Network and System Sciences, 2017, 10, 299-326
http://www.scirp.org/journal/ijcns
ISSN Online: 1913-3723
ISSN Print: 1913-3715

HiMAC: Hierarchical Message Authentication Code for Secure Data Dissemination in Mobile Ad Hoc Networks

Khaleel Mershad, Ali Hamie, Mohamad Hamze

Department of Computer and Communications Engineering, Faculty of Engineering, Arts, Sciences, and Technology University in Lebanon (AUL), Beirut, Lebanon

Email: khaleel.mershad@aul.edu.lb, ali.hamie@aul.edu.lb, mohamad.hamze@aul.edu.lb

How to cite this paper: Mershad, K., Hamie, A. and Hamze, M. (2017) HiMAC: Hierarchical Message Authentication Code for Secure Data Dissemination in Mobile Ad Hoc Networks. *Int. J. Communications, Network and System Sciences*, **10**, 299-326.
https://doi.org/10.4236/ijcns.2017.1012018

Received: October 28, 2017
Accepted: December 12, 2017
Published: December 15, 2017

Abstract

Mobile ad-hoc networks are wireless self-organized networks in which mobile nodes can connect directly to each other. This fact makes such networks highly susceptible to security risks and threats, as malicious nodes can easily disguise as new trusted nodes and start attacking the network after a certain period of time. Hence, the security of data transmission in MANET has been a hot topic in the past years. Several research works attempted to detect and stop various attacks on MANET nodes and packets. This paper presents an efficient mechanism for secure data dissemination in MANETs. Our approach is based on the identity based cryptography and Message Authentication Code (MAC). The proposed security mechanism prevents malicious nodes from tampering or replaying intermediate packets by means of signing and encrypting the packet at each intermediate trusted node. We tested the efficiency of our system using the ns2 simulator by comparing it to a similar security mechanism. The simulations illustrate that our approach obtains many advantages over other existing approaches for secure data dissemination in MANETs.

Keywords

MANET, Security, ID Based Cryptography, MAC Protocol, Hierarchical Security, Trust Mechanism, Cryptography Analysis, NS2

1. Introduction

Over the last years mobile computing and mobile ad hoc networks have rapidly developed and expanded. Mobile computing is referred to any system that uses dynamic wireless communications and does not depend on preinstalled infrastructure. Mobile ad hoc networks (MANETs) are a decentralized type of wireless networks which are basically deployed for the purpose of temporary communication in both normal and adverse situations. It is called an ad hoc network because it does not rely on a preexisting infrastructure, such as routers in wired networks or Access Points (APs) in managed (infrastructure) wireless networks. Instead, each node participates in routing by forwarding data to other nodes, and the process of determining which nodes should forward data is dynamically calculated based on the network connectivity. MANETs can use various routing schemes like hop-by-hop communication or other classical and modern approaches. In addition to classic routing, MANETs can use flooding for forwarding data. Flooding itself becomes an important issue when we study network congestion.

As MANET applications that include information sharing and data dissemination are increasing in a rapid pace, the demand for optimizing network resources and attaining data security has become a paramount concern for the research community. Achieving strong security in MANETs is a very challenging task due to the presence of two main constraints: 1) constantly changing topology and 2) high mobility of nodes. As nodes join and leave the network, the network topology constantly changes. Under such circumstances, it is difficult to maintain a constant routing table at each node which becomes a very expansive process that consumes the node limited resources. Moreover, MANET nodes might have high mobility in certain cases, such as natural disasters and evacuation scenarios. Taking into regards the above challenges, a unified security approach that maintains fast and successful communication efficiency is required for data dissemination in MANETs.

In this paper, we propose an approach that allows a source mobile node to send a data packet securely to a destination mobile node, while ensuring that the packet was forwarded through trusted intermediate mobile nodes. In other words, our approach makes sure that a malicious node will be detected if it performs an attack on a packet while it is being sent from a source to a destination. Hence, our approach creates what we can call a trusted route between a source and a destination. Contrary to other systems, our approach does not maintain the trusted route at each node. Rather, the route is calculated dynamically on the fly while the packet is being forwarded by an intermediate node. The trusted route ensures that the packet is forwarded by trusted nodes only. In order to do that, each intermediate node adds its own signature and a timestamp to the packet before encrypting it with the destination's public key. When the destination receives the packet, it hierarchically decrypts it and authenticates each intermediate node by checking its signature. Hence, the destination can reversely track the route of the packet and make sure that all nodes that forwarded the packet were trusted ones. We call our approach Hierarchical Message Authentication Code or HiMAC.

In addition to authenticating intermediate nodes, HiMAC strives to provide trustworthiness between MANET nodes. In previous works, maintaining the trust between nodes depended on the existence of a centralized authority to distribute certificates to the nodes. However, this approach may not work well in MANETs since the assigned trusted third party may leave the network at any time. Moreover, it might take a long time to establish the trust for a newly arrived node. For these reasons, HiMAC maintains a trust mechanism via distributed cooperation between all nodes. Although there exist many proposals for secure communication in MANETs [1]-[8], they still face the problem of how to maintain efficiency, trust, and security in a MANET framework.

The rest of the paper is organized as follows: A brief overview of securing data dissemination in MANETS is presented in Section 2. The detailed description of our proposed system is in Section 3, while Section 4 analyzes the encryption/ decryption delays via mathematical analysis. In Section 5 we evaluate our approach via software simulations. Finally, concluding remarks are presented in Section 6. Before proceeding to Section 2, we present a list of the acronyms used in this paper in **Table 1**.

2. Literature Review

A vast number of security mechanisms and protocols have been proposed for

Table 1. List of acronyms.

Acronym	Definition
HiMAC	Hierarchical Message Authentication Code
MANET	Mobile Ad hoc Network
AP	Access Point
QoS	Quality of Service
DoS	Denial of Service
SMT	Secure Message Transmission
SEAD	Secure Efficient Ad hoc Distance vector
PLRSA	Promiscuous Listening Routing Security Algorithm
SecMR	Secure Multipath Routing
SAODV	Secure Ad hoc On-Demand Distance Vector
TAODV	Trusted Ad hoc On-Demand Distance Vector
DSR	Dynamic Source Routing
GKA	Group Key Agreement
TTL	Time-to-Live
RSA	Rivest-Shamir-Adleman public-key cryptosystem
AES	Advanced Encryption Standard
3DES	Triple Data Encryption Standard
MIPS	Million instructions per second
RREQ	Route Request
RREP	Route Reply
RWP	Random Waypoint movement model

MANETs by the research community over the years. These protocols are studied form different perspectives keeping in mind the properties and constraints of MANETs. In general, the most studied security attacks in MANETs are the Gray Hole attacks and the Black Hole attacks [9] [10] [11].

Hu *et al.* [1] presented the different attacks against routing in ad hoc networks, and the design and performance evaluation of a new secure on-demand ad hoc network routing protocol which they gave the name Ariadne. They also discussed the different types of Denial-of-Service attacks on routing paths and the various preventive approaches. Papadimitratos *et al.* [3] [4] proposed the Secure Message Transmission (SMT) protocol for mobile ad hoc networks. They described that the SMT protocol is better matched to support QoS for real-time communications in the ad hoc networking environment more than other environments. Hu *et al.* [5] proposed a Secure Efficient Distance Vector Routing for Mobile Wireless Ad Hoc Networks which they called the Secure Efficient Ad hoc Distance vector routing protocol (SEAD). The proposed protocol uses one-way hash functions instead of cryptographic operations in securing the routed messages.

Wu *et al.* [6] presented a survey on various possible attacks and countermeasures in Mobile Ad Hoc networks. Li *et al.* [7] proposed a routing security algorithm for mobile hosts to detect malicious attacks in the middle; and they named it promiscuous listening routing security algorithm (PLRSA). Their proposed algorithm is distributed in nature without any need of communication between hosts. Each node in PLRSA, can switch into the promiscuous listening mode to intercept all packets passing through the mobile host in order to monitor the other nearby nodes. When a node performs a malicious behavior, such as dropping or tampering data packets, the other nearby nodes will detect the spiteful behavior.

Komninos *et al.* [8] discussed the main security issues for protecting mobile ad hoc networks at the data link and network layers. They first identified the security requirements for these two layers and then the design criteria for creating secure ad hoc networks using multiple lines of defense against malicious attacks. Xiaopeng *et al.* [9] discussed the Gray Hole attack which leads to the Denial of Service (DoS) attack. In the Gray Hole Attack an adversary silently drops some or all of the data packets sent to it instead of forwarding them. Mavropodi [12] proposed an on-demand multipath routing protocol which they named secure multipath routing protocol (SecMR), and analyzed its security properties. Komninos *et al.* [13] described the layered security approach, design criteria and performance analysis of some MANET security protocols.

Zhao *et al.* [14] proposed a secure routing protocol with proactive security approach for MANETs. In this paper the authors allowed only legitimate nodes to participate in the bootstrapping process, rather than trying to detect adversary nodes while they are participating in the routing protocol. Marchang *et al.* [15] proposed two intrusion detection techniques for MANETs, which rely on collaborative efforts of nodes in a neighborhood to detect a malicious node in that neighborhood. Yeun *et al.* [16] proposed a group key agreement protocol for end-to-end security in MANET environments that do not have any fixed infrastructure. Cordasco *et al.* [17] presented a comparison between SAODV and TAODV which address routing security through cryptographic and trust-based means respectively. They also provided the performance comparison on actual resource-limited hardware. Kim *et al.* [18] discussed the security of the route discovery process in DSR.

Dutta *et al.* [19] proposed a generalized self-healing key distribution using a vector space access structure. They described three efficient constructions for scalable self-healing key distribution with t-revocation capability. Su [20] proposed a wormhole-avoidance routing protocol on the basis of anomaly detection. Makri *et al.* [21] reviewed and evaluated a number of constant round Group Key Agreement (GKA) protocols. Su [10] discussed the prevention of some selective black hole attacks on MANETs through intrusion detection systems. Khalil *et al.* [22] presented a scalable countermeasure for the control traffic tunneling attack. The proposed system uses trusted nodes called cluster heads (CH) for global tracking of node locations in order to detect and isolate malicious nodes. Bhalaji *et al.* [11] proposed a Dynamic Trust Based Method to alleviate Grey hole attack in MANETs. Singh *et al.* [23] presented a state of the art survey on the ant-based routing protocols and a taxonomy of various ant colony algorithms with their different advantages and disadvantages.

Omar *et al.* [24] propose a secure and reliable certificate chains recovery protocol for mobile ad hoc networks. In the proposed framework, the MANET users take the role of the certification service by issuing and managing the public-key certificates. The shortest and the safest certificate chains are selected in order to reduce the communication overhead and resist against compromised nodes which can generate false certificates. The authors in [25] propose a mesh-based multipath routing scheme to discover all possible secure paths using secure adjacent position trust verification protocol. Better link optimal path is determined by the Dolphin Echolocation Algorithm for efficient communication.

In addition to basic MANETs, the security of special types of Ad hoc Networks, such as Wireless Sensor Networks (WSNs) and Vehicular Ad hoc Networks (VANETs) have been extensively studied. For example, Dhyani *et al.* [26] present a mechanism for averting black hole attack in VANETs by unicasting data packet to vehicles and using the trust factor technique to detect routing misbehavior and ensure the relaying of data packets to the destination. Mershad *et al.* [27] introduce a system that takes advantage of the roadside units (RSUs) that are connected to the Internet and to secure the communications between VANET users. The system uses a hierarchal password-based key derivation function to provide data confidentiality at each intermediate node. Jan *et al.* [28] propose a lightweight payload-based mutual authentication scheme for a cluster-based hierarchical WSN. The proposed scheme operates in two steps. First, an optimal percentage of cluster heads is elected, authenticated, and allowed to communicate with neighboring nodes. Second, each cluster head, in a role of server, authenticates the nearby nodes for cluster formation.

As MANETs can encounter fast topological changes, security becomes a paramount concern to detect and stop attackers before they succeed in performing their attacks. Security can be described in two folds: on route and data levels. Existing security solutions [1]-[8] are able to secure the network against a certain type of attacks, but remain vulnerable to other types of inside and outside attacks. Although there exist many solutions to mitigate these attacks but they are not sufficient enough with constant topological changes in the network. Hence a unified mechanism is required which prevents packet drop and tampering, packet replay, impersonation, and false data attacks. In this paper a trust mechanism between the different nodes is coupled with a strong authentication scheme in order to detect and avoid the described attacks.

3. Proposed Mechanism

A large number of research works attempted different solutions to security threats in various types of Ad hoc Networks, such as WSNs and VANETs. Nevertheless, Security remains a major concern in the future of such networks. Based on the literature review and the critical analysis of MANET security, we can see that no framework have established a strong proof of high security, efficiency, and trust classification at the same time. The main drawback in previous works is the difficulty of establishing the trustworthiness of newly arrived node. Many approaches in the literature establishes this trust by monitoring the behavior of a new node for a certain period of time, and then classifying this node as trustworthy if no malicious behavior was detected by it. However, an attacker might hide its malicious behavior for a long period of time, and then perform its attack after it is declared trustworthy by other nodes.

In our system, we argue that trust should be based on identity and not only on behavior. In real life, a person might be trusted by some people and at the same time not trusted by others. As mobile nodes are in reality the people operating them, the trust of a mobile node should be coupled with its user's identity. A newly arrived node will broadcast its identity to its neighbors. If one of the trusted neighbors acknowledges its trustworthiness, the node is saved as trusted by the other neighbors. If no neighbor knows the new node, it remains saved as suspicious until a certain trusted node in the network that knows the new node discovers its existence and declares its trustworthiness; or until a certain time has passed after which the behavior of the new node is classified as normal and its trustworthiness is established.

In order to maintain this trust mechanism, a single parameter is added to the neighbors list and to the routing table that are maintained by the routing protocol at each node. This parameter is called *trust*, and can have values equal to "malicious", "ambiguous", "potentially trusted", or "trusted". When a new node

N_n arrives, its neighbors classify it as "ambiguous". When a node N receives a neighbor-broadcast packet from a "trusted" node N_T that the new node N_n is trusted, it changes the *trust* of N_n to "potentially trusted". After that, N asks its user whether N_n should be classified as "trusted" or not, by displaying a list of the behavior of N_n so far, and the identity of the user of N_n as he announced himself. It's always up to the user of a node to declare that a certain node in the network can be classified as "trusted". As long as the user hasn't stated so, N_n remains "potentially trusted", since it was not fully "trusted" by the user of the node. At any time the user can change the *trust* of a node N_n from "potentially trusted" to "trusted", after he/she decides that he now trusts the user of N_n. Note that some MANET nodes (such as wireless printer) will not have a user. These nodes will not have the "potentially trusted" classification in their routing algorithm. Rather, such node will declare a new node as "trusted" whenever it receives a neighbor-broadcast packet from a trusted neighbor that this new node is "trusted". On the contrary, if any node receives a neighbor-broadcast from a "trusted" node that a certain node N_m is "malicious", it changes the *trust* of N_m to "malicious". Note that a neighbor-broadcast packet is a packet that is broadcasted only to neighbors, by setting the Time-to-Live (TTL) of the packet to 1. Also, note that neighbor-broadcast packets will be secured using the HiMAC security mechanism, as we will explain shortly.

In order to secure data dissemination packets without affecting the route optimization, while at the same time maintaining trust between mobile nodes, we propose a Hierarchical security protocol for message authentication and encryption (HiMAC). Using our protocol, a node S who wants to send a packet to another node D, examines the set of possible nodes that can be selected as a "next hop" to D. For example, in Figure 1, the set of nodes that can act as "next hop" is $\{A_1, B_1, C_1\}$. At this stage, two factors are considered in selecting a certain node from this set: *trust* value and route efficiency. S first examines the *trust* of each node in the list. If only one node in the list is "trusted", then it is selected as the "next hop". If more than one node in the list is "trusted", S selects among these nodes the one that lies inside the most efficient route; *i.e.*, the node whose total path to the destination has the least cost. For example, in Figure 1, suppose

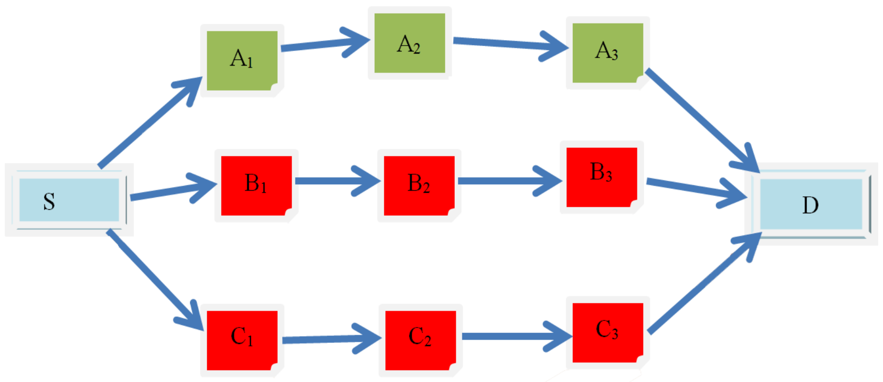


Figure 1. Example of multiple routes from source S to destination D.

that both B_1 and C_1 are "trusted", and that the cost of the route B_1-B_2-B_3 is equal to 15, while the cost of C_1-C_2-C_3 is equal to 20. Then S chooses B_1 as the "next hop".

Before sending the packet to B_1, S uses its private key to generate the message signature using MAC algorithm. After that, S combines (concatenates) the message, the generated signature, and a timestamp value into a single block, and encrypts this block using the receiver's (*i.e.*, D) public key. In HiMAC, each data packet will contain an additional element within its payload, which is the list of traversed nodes. This list is encrypted separately from the block that contains the {message, signature, and timestamp}. So, S adds its ID to the list of traversed nodes (S will be the first node in this list). Then S encrypts this list using D's public key. Next, S combines the encrypted block and the encrypted list of traversed nodes into a single packet, and sends the packet to the "next hop". Note that a special character separator will be used to separate the encrypted block from the encrypted list, and to separate the message, signature, and timestamp within the encrypted block. The reason for encrypting the list of traversed nodes is to prevent the attacker from maliciously changing the number of hops in the packet without the knowledge of the destination. In SAODV [29], the hop count is secured using hash chains. In HiMAC, we secure the hop count by integrating the list of traversed nodes and encrypting it hierarchly as we explain next. So, the final message that will be sent from S to "next hop" will be:

$$M_1 = E_{KPU_D}\left\{M_p \,||\, MAC_{KPR_S}\left(M_p\right) || \, T_s\right\} || \, E_{KPU_D}\left(ID_S\right) \tag{1}$$

where M_p is the original message, M_1 is the cypher message, ID_s is the ID of S (which will be the only element in the list of traversed nodes at this stage), T_s is the timestamp value, E_{KPU_D} denotes the encryption using D's public key, MAC_{KPR_S} denotes the MAC signature using S's private key, and || is a special character.

Each intermediate node I will perform the same steps that were done by S, with the exception that I will be using M_1 as the message to work on instead of M_p. First, I separates the encrypted block from the encrypted list of nodes, saves the encrypted block as M_{11} and the encrypted list as M_{12}. Then it calculates the next hop, generates signature of M_{11} using its own private key, generates block of {M_{11}||MAC(M_{11})|| timestamp} and encrypts it using destination's public key, and adds itself to M_{12} and encrypts the result with the destination's public key. Hence, the message that is forwarded by node I to the "next hop" will be:

$$M_2 = E_{KPU_D}\left\{M_{11} \,||\, MAC_{KPR_I}\left(M_{11}\right) || \, T_s\right\} || \, E_{KPU_D}\left(M_{12} \,||\, ID_I\right) \tag{2}$$

where $M_{11} = E_{KPU_D}\left\{M_p \,||\, MAC_{KPR_S}\left(M_p\right) || \, T_s\right\}$ and $M_{12} = E_{KPU_D}\left(ID_S\right)$.

This process will be repeated at each intermediate node, until the packet reaches D. Here, D will follow a reverse hierarchical decryption technique to retrieve the ID of each intermediate node who forwarded the packet, check the authenticity of each of these nodes using the latter's MAC signature, and repeat the process until it calculates the original message M_p. If all intermediate nodes who forwarded the packet are "trusted" by D, and the signatures of all these nodes were found correct, the packet will be considered fully secure. If one or more intermediate nodes who forwarded the packet were classified by D as "malicious" or "ambiguous", D might drop the packet, based on the application and the user's decision.

In details, when a destination node D receives a cypher message M_n, the first step is to separate the encrypted block (M_{n1}) from the encrypted list of nodes (M_{n2}). Then D decrypts both parts and retrieves from M_{n2} the ID of the last node that forwarded the packet (suppose this node is called I_n). Next, D separates the message, MAC signature, and timestamp from the decryption of M_{n1} (which we will call $M_{(n-1)}$, MAC_n, and Ts_n). After that, D uses the public key of I_n, $M_{(n-1)}$, and MAC_n as inputs to the MAC algorithm to check the correctness of the signature and ensure the authenticity of I_n. If authenticity is established and I_n is "trusted" by D, the latter moves to the next step, in which D decrypts $M_{(n-1)2}$ (which is obtained after removing I_n from M_{n2}) to obtain the ID of the intermediate node that forwarded the packet before I_n (which we will call $I_{(n-1)}$). Then D decrypts $M_{(n-1)}$ to obtain $\mathrm{M}_{(n-2)}$, $MAC_{(n-1)}$, and $Ts_{(n-1)}$. After that, D uses the public key of $I_{(n-1)}$, in addition to $M_{(n-2)}$ and $MAC_{(n-1)}$ as inputs to the MAC algorithm to check the authenticity of $I_{(n-1)}$, and so on … At the last step, the decrypted list of nodes will contain one element, which is the ID of S, and the decrypted block will contain the original message M_p, the MAC signature of S, and a timestamp. At this final stage D will authenticate S using its signature and will use the data from M_p in the application. Also, D will check the list of timestamps L{T_s} that were added by the intermediate nodes before encryption. If one or more timestamps are old, or if the timestamps are not in decreasing sequence, this shows a possibility of packet tampering or some kind of replay attack, in which case D will discard the packet.

Note that HiMAC depends on the existence of an efficient public key cryptography mechanism in the MANET. In this scheme, each mobile node should save the public key of each other "trusted" node in the network. In other words, a node shares its public key with other "trusted" nodes only. This can be achieved as follows: Suppose that a node N is currently classifying a new node N_n as "ambiguous". Now, N receives a neighbor-broadcast packet from a "trusted" node N_T that includes that N_n is "trusted". N changes N_n to "potentially trusted" and asks the user whether N_n should be trusted or not. Supposing that the user acknowledges that N_n is "trusted", then N changes the *trust* parameter of N_n to "trusted". After that, N checks if N_T is within range. If yes, N sends a "Public Key Request" packet to N_T, asking it for the public key of N_n. The latter encrypts the public key of N_n using the public key of N, and sends the result to N in a "Public Key Reply" packet. If N_T is not within range, N sends a neighbor-broadcast packet to its neighbors requesting the public key of N_n. When N receives a "Public Key Reply" packet that contains the public key of N_n from a neighbor N', it checks whether N' is "trusted" or not. If N' is trusted, N saves the received public key of N_n; else, N discards the packet of N'.

The main objective in HiMAC is to prevent an intruder from capturing a certain route through which data is transmitted and performing attacks on the data packets that are passing through the route. Mainly, we are concerned with data tempering attack, replay attack, flooding attack, and impersonation attack. The intruder will not be allowed to perform flooding attack since it will not be classified as "trusted" because it is not known by the users of the mobile nodes. Hence, its packets will not be forwarded by intermediate nodes. If an intruder performs a tampering attack on a packet, the destination will discover the attack since it will not be able to correctly decrypt the received message. Even if the intruder follows the rule of HiMAC and uses its signature to sign the packet, the destination will not be able to authenticate the intruder since it doesn't know the intruder's public key. Also, HiMAC can be used to detect replay attacks by attackers who capture a legitimate message and send it at a later time as a new message. This can be done at the destination by checking the list of timestamps that were added by the intermediate nodes as described before. Finally, HiMAC enables a destination node to detect an impersonation attack as follows: If a node N_a adds itself to the packet as node N_o, the destination node will detect the attack since it will not be able to verify the false signature of N_a since it will be different from the expected signature of N_o.

Figure 2 represents the Hierarchical Message Authentication Code (HiMAC) general functioning. Here a data message is to be sent form Sender (S) to Receiver (R). First we describe the case when S is a neighbor of R. In this case S incorporates the message with the MAC (M) which is generated with the help of

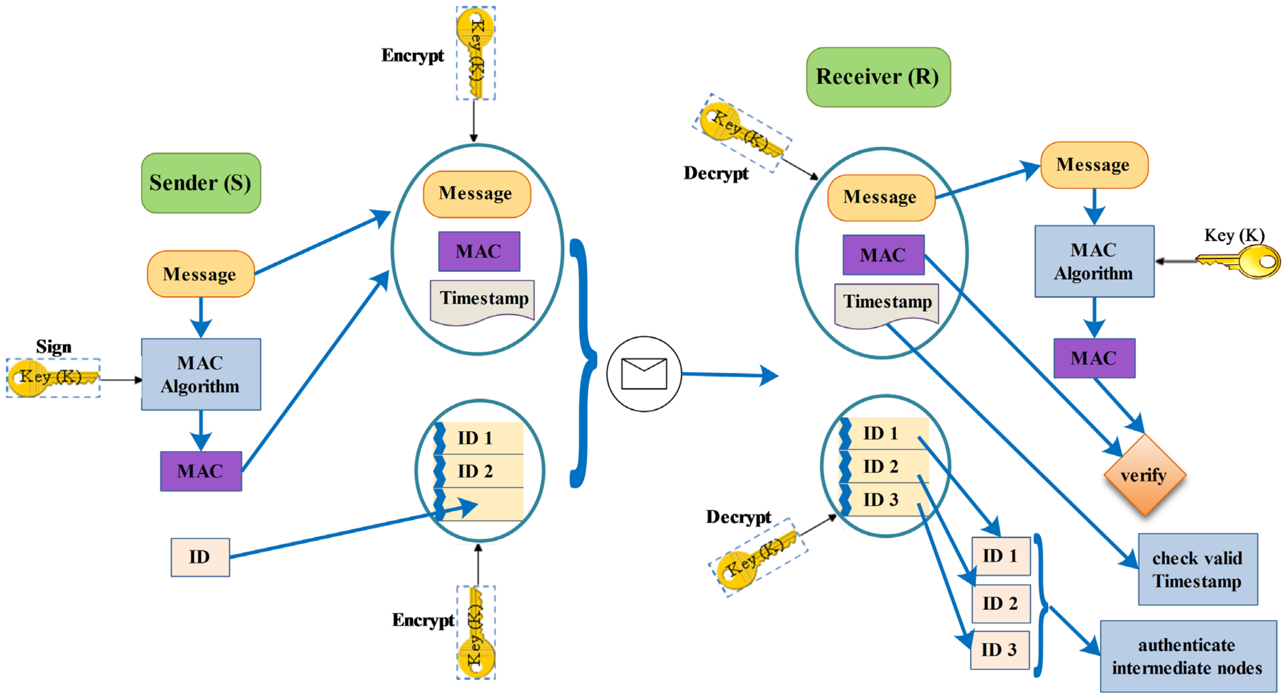


Figure 2. Hierarchical Message Authentication Code (HiMAC) operations from Sender (S) to Receiver(R).

MAC algorithm, and with a timestamp. At the receiver end the message is decrypted and the MAC is matched with the sender MAC and checked, if it is the same then R confirms the integrity of the sender. Otherwise the message integrity is not certified and the message should be dropped.

In the second scenario, we consider that there are some intermediary nodes I1, I2, I3 … Then the message should have to pass through these intermediary nodes. Each intermediate node will add its MAC and timestamp, in addition to its ID to the list of IDs in the message, as shown in **Figure 2**. The receiver node (R) will decrypt each layer of the message to generate the corresponding MAC and check the identity of the corresponding node after decrypting the list of IDs as shown in **Figure 2**. **Figure 3** and **Figure 4** illustrate the algorithms at the sender and receiver for the first and the second scenarios, while **Figures 5-7** describe the steps of the algorithms of scenario 2 using flowchart diagrams. We included only the flowcharts of scenario 2 as those of scenario 1 are simpler and can be deduced easily from scenario 2.

4. System Analysis

In the next section, we test the performance of the proposed security framework using network simulations. An important parameter that should be studied is the time required to encrypt/decrypt a message using HiMAC. In this section, we analyze the various factors that affect this parameter, and compare the results of our analysis with the simulation results in the next section. First, we note that we will be using the RSA algorithm from the Crypto++ library [30] in the simulations; hence, the benchmarks that will be presented in this section are based on Crypto++ RSA. Also, we note that when a message is encrypted by a source or an intermediate node and when the message is decrypted at the destination node, two main factors will affect the time needed for encryption/decryption: 1) the size of the RSA key, and 2) the size of the message. Several studies, such as [31] and [32] proved that the RSA encryption/decryption delay will increase as the key size increases. Also, it is well known that when we want to encrypt a large message using RSA, we first generate a symmetric key using a symmetric key cryptography algorithm (such as AES, 3DES, PRESENT, ...) and use it to encrypt the message, then we encrypt the symmetric key with the RSA public key. Hence, we use both symmetric and asymmetric encryption and decryption when using RSA to encrypt/decrypt large messages.

In this section, we will consider a plain-text message of size S_{d0} bytes at the sender. At the source and at each intermediate node, additional data will be added to the encrypted message as we explained in Section 3. Hence, the size of the message that should be encrypted will increase at each intermediate node. The mathematical analysis performed in [33] and [34] proved that the encryption/decryption delay of AES depends on the message size (S_d) and the number of million instructions per second (MIPS) of the processor (C_p). The encryption delay of AES was derived in [34] and found to be equal to:

Scenario 1:

Algorithm for sender S:

```
1. Input: Receiver R, Message $M_{S,}$  Private key $P_r$,
        MAC Algorithm $M_{algo}$(Message, Key), Timestamp $T_S$
2. Apply $M_{algo}(M_S, P_r)$ algorithm to generate signature MAC(M)
3. Add MAC (M) and $T_S$ to the input message $M_S$ to obtain ($M_S$, MAC(M), $T_S$)
4. Encrypt the result using public key of R
5. Output: Encrypted ($M_S$, MAC(M), $T_S$)
```

Algorithm for receiver R:

```
1. Input: Sender S, Message = Encrypted ($M_S$, MAC(M), $T_S$), Public key of S $P_u$
2. Decrypt the received message using private key
3. Apply $M_{algo}(M_S$ +MAC(M), $P_u)$ to check the validity of MAC(M)
4. If the MAC is valid then the message is authentic
```

Figure 3. HiMAC algorithms for sender S and receiver R for scenario 1.

Scenario 2:

Algorithm for Sender S:

```
1. Input: Receiver R, Message $M_{S,}$  Private key $P_r$, MAC Algorithm $M_{algo}$(Message, Key),
      Timestamp $T_S$
2. Apply $M_{algo}(M_S, P_r)$ algorithm to generate signature MAC(M)
3. Add MAC (M) and $T_S$ to the input message $M_S$ to obtain ($M_S$, MAC(M), $T_S$)
4. Encrypt the result using public key of R
5. Encrypt ID with public key of R and add it to message
6. Output: Encrypted ($M_S$, MAC(M), $T_S$) + Encrypted (S)
```

Algorithm for Intermediate node I:

```
1. Input: Receiver R, Message = Encrypted (M, MAC(M), $T_S$) + Encrypted (L), Private key $P_r$,
   MAC Algorithm $M_{algo}$(Message, Key), Timestamp $T_{SI}$
2. Separate Message into two parts: $M_I$ = Encrypted (M, MAC(M), $T_S$) and Encrypted (L)
3. Apply $M_{algo}(M_I, P_r)$ algorithm to generate signature MAC($M_I$)
4. Add MAC (M) and $T_{SI}$ to $M_I$ to obtain ($M_I$, MAC($M_I$), $T_{SI}$)
5. Encrypt the result using public key of R
6. Add ID to Encrypted (L) and encrypt result using public key of R
7. Output: Encrypted ($M_I$, MAC($M_I$), $T_{SI}$) + Encrypted (Encrypted (L)+ I)
```

Algorithm for Receiver R:

```
1. Input: Sender S, Message = Encrypted (M, MAC(M), $T_S$) + Encrypted (L), Public key of S
   $P_u$, Public keys of Intermediate nodes $\{P_{UI}\}$
2. do:
3.     Separate message into 2 parts: $M_1$ = Encrypted (M, MAC(M), $T_S$) and $M_2$ = Encrypted
   (L)
4.     Decrypt $M_2$ using private key and extract ID of node $I$ from it
5.     Decrypt $M_1$ using private key and extract M, MAC(M), and $T_S$ from it, and Save $T_S$
6.     Apply $M_{algo}$(M +MAC(M), $P_{UI}$) to check the validity of MAC(M)
7.     If the MAC is not valid: output Integrity error and exit
8.     If MAC is valid continue with next iteration using M and Encrypted(L-1)
9. while (Encrypted(L) not empty)
10. Check values and sequence of all timestamps $T_S$
11. If one or more $T_S$ is old or $\{T_S\}$ are out of sequence, output validity error and exit
12. Pass final message $M_S$ to application
13. Output: Integrity is checked, Message $M_S$.
```

Figure 4. HiMAC algorithms for sender S, intermediate node I, and receiver R for scenario 2.

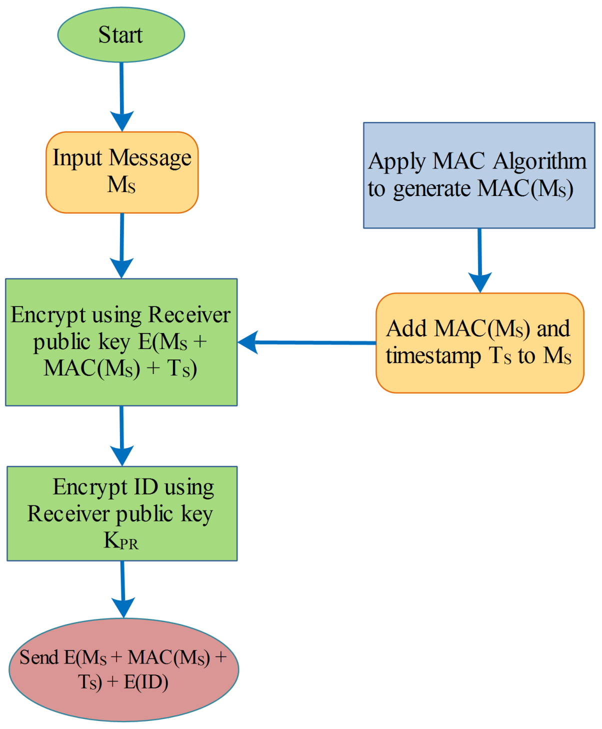

Start
Input Message M_S
Apply MAC Algorithm to generate $MAC(M_S)$
Encrypt using Receiver public key $E(M_S + MAC(M_S) + T_S)$
Add $MAC(M_S)$ and timestamp T_S to M_S
Encrypt ID using Receiver public key K_{PR}
Send $E(M_S + MAC(M_S) + T_S) + E(ID)$

Figure 5. Flowchart of HiMAC algorithm for sender S using scenario 2.

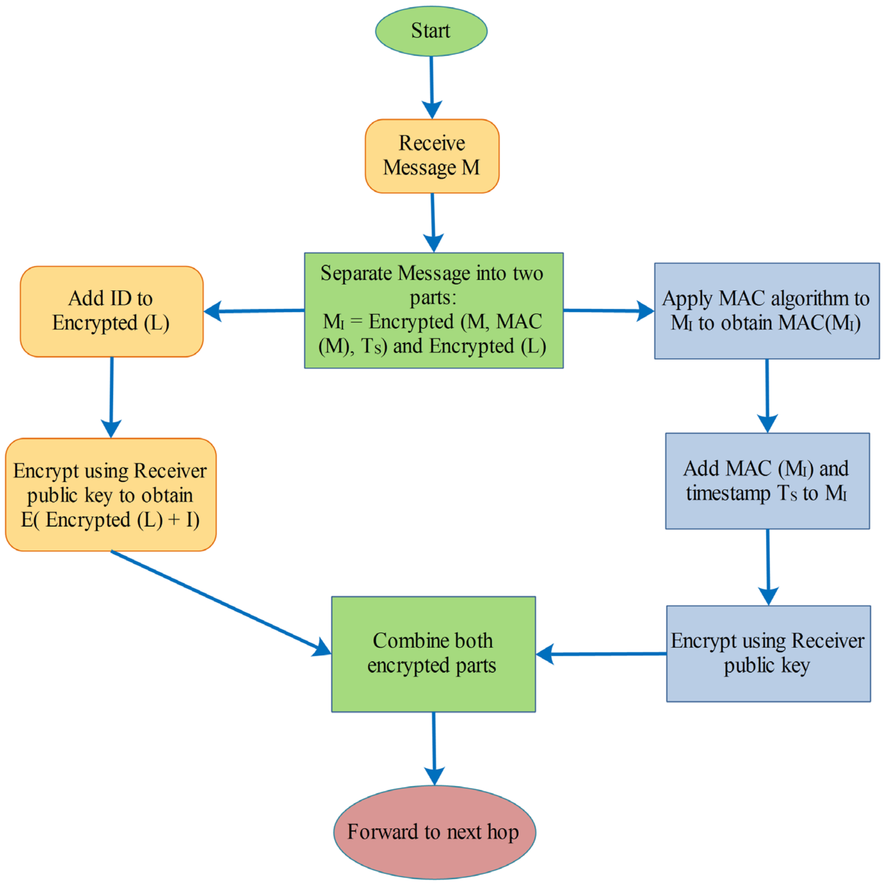

Start
Receive Message M
Separate Message into two parts: M_I = Encrypted (M, MAC (M), T_S) and Encrypted (L)
Add ID to Encrypted (L)
Apply MAC algorithm to M_I to obtain $MAC(M_I)$
Encrypt using Receiver public key to obtain E(Encrypted (L) + I)
Add MAC (M_I) and timestamp T_S to M_I
Combine both encrypted parts
Encrypt using Receiver public key
Forward to next hop

Figure 6. Flowchart of HiMAC algorithm for intermediate node I using scenario 2.

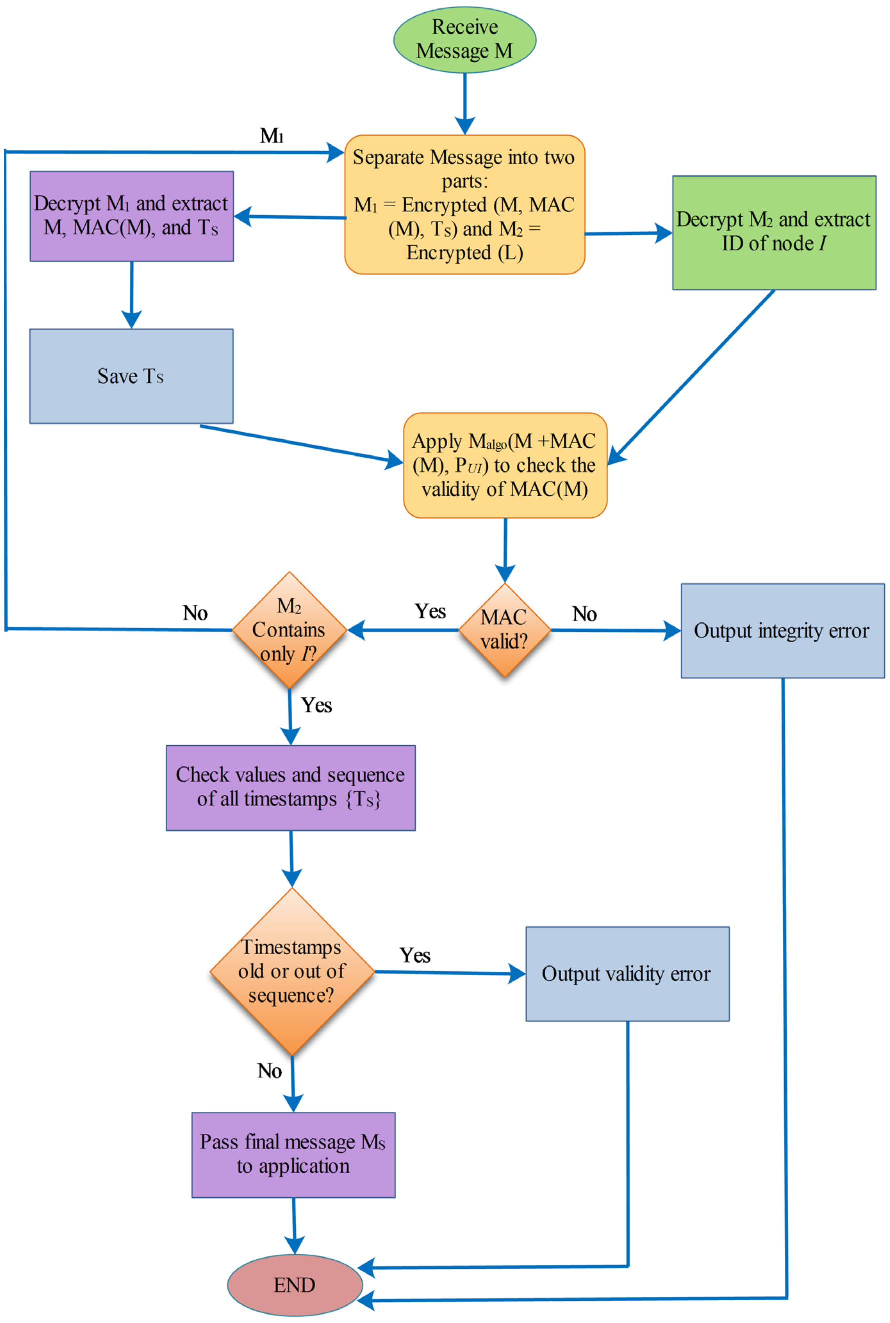

Receive Message M
M₁
Separate Message into two parts: M₁ = Encrypted (M, MAC (M), Ts) and M₂ = Encrypted (L)
Decrypt M₁ and extract M, MAC(M), and Ts
Decrypt M₂ and extract ID of node I
Save Ts
Apply Malgo(M +MAC (M), PUI) to check the validity of MAC(M)
No
M₂ Contains only I?
Yes
MAC valid?
No
Output integrity error
Yes
Check values and sequence of all timestamps {Ts}
Timestamps old or out of sequence?
Yes
Output validity error
No
Pass final message Ms to application
END

Figure 7. Flowchart of HiMAC algorithm for receiver R using scenario 2.

$$T_{AES-Enc}\left(S_d, C_p\right) = \frac{8 \times S_d}{128} \times \frac{T_{AES-Enc}}{C_p} \qquad (3)$$

where S_d is given in bits and C_p is given in MIPS. On the other hand, $T_{AES-Enc}$ is the number of processing cycles required to encrypt one block of data, which is equal to 6168 processing cycles for a 128 bit key (which is the size of the AES session key that we will use in the simulations). In a similar way, the authors in [34] calculate the number of processing cycles needed to decrypt one block of data ($T_{AES-Dec}$) and use it to calculate the total decryption delay of a packet, which is derived as:

$$T_{AES-Dec}\left(S_d, C_p\right) = \frac{8 \times S_d}{128} \times \frac{T_{AES-Dec}}{C_p} \tag{4}$$

In order to analyze the performance of HiMAC, we need to know the time that will be taken by both operations: 1) encrypting the two parts of the message (where the first part contains the message, signature, and timestamp and the second part contains list of IDs) using AES key, and 2) encrypting the AES key using the RSA public key (and decrypting these three parts at the receiver). The delay of the second operation is constant at all nodes, since they will be using the same size for RSA and AES keys. This delay will be equal to that of a single RSA encryption operation. As for the first part; since the encryption/decryption using AES depends on the message size (S_d), we need to derive the message size at the source node and at each intermediate node N_i, (where N_1 is the first intermediate node, N_2 is the second intermediate node, and so on).

At the source node, the two parts of the message will be encrypted separately using the AES key. The size of the first part will be equal to (S_{d0} + size-of-signature (S_{d0}) + size-of-timestamp). Since the RSA signature size is always equal to the size of the used key, hence we will obtain a signature of size 128 bytes when using a 1024 bit key. Also, considering that the timestamp has a size equal to 4 bytes, the total size of part 1 will be equal to (S_{d0} + 132) bytes. Knowing that AES adds padding to the message to make it a multiple of the AES block size, the resulting message size will become:

$$S_{d1} = \left\lceil \frac{\left(S_{d0} + 132\right)}{Bl} \right\rceil \times Bl \tag{5}$$

where Bl is the size of one encryption block in AES, which is equal to 128 bits. With respect to the second part of the total message, it will depend on the number of intermediate nodes. In this section, we will derive the average expected number of hops (*i.e.*, intermediate nodes) between any two nodes in the network E[H]. Suppose that the ID of each intermediate node requires 2 bytes, then the size of the second part will be equal to (2 × $E[H]$). After encryption, the size of the second part will be the same for all intermediate nodes, which is equal to Bl (since the size of 2 × $E[H]$ is less than Bl in normal cases).

Similar to the source node, each intermediate node will encrypt three things: the two parts of the message that were encrypted by the previous node will be encrypted using a 128-bit AES key (after adding necessary information as explained in Section 3), and the AES key using the receiver's public RSA key. The sizes of the second part of the message and the size of the encrypted AES key will be the same as those calculated previously for the source node. With respect to the first part of the message, we will derive it as follows: at the first intermediate node, the size of the first part of the message is S_{d1}. The first intermediate node will add the signature of this part and the timestamp to it. Hence, the total size will be equal to (S_{d1} + 132), then it will add the padding to make the final size multiple of AES block size, which will result in a message of size equal to:

$$S_{d2} = \left\lceil \frac{(S_{d1} + 132)}{Bl} \right\rceil \times Bl \tag{6}$$

In general, the size of the first part of the message that will be encrypted by Node N_i is:

$$S_{d(i+1)} = \left\lceil \frac{(S_{di} + 132)}{Bl} \right\rceil \times Bl \tag{7}$$

Before we derive the encryption/decryption delay, we calculate the average expected number of hops between two nodes in a MANET. Similar to [35], we assume a rectangular topology with area $a \times b$ and uniform distribution of nodes. Two nodes can form a direct link if the distance S between them is $\le r_0$, where r_0 is the maximum node transmission range. We seek to compute the expected number of hops between any two nodes. Using stochastic geometry, the probability density function of S is given in [35] as:

$$f(s) = \frac{4s}{a^2b^2}\left(\frac{\pi}{2}ab - as - bs + 0.5s^2\right) \tag{8}$$

for $0 \le s < b < a$. It is concluded that if two nodes are at a distance s_0 from each other, the number of hops between them, when there are a sufficient number of nodes to form a connected network, would tend toward s_0/r_0. Hence, $E[H]$, the expected minimum number of hops between any two nodes in the network, is equivalent to dividing $E[S]$, the expected distance, by r_0. It should be noted that the value of $E[H]$ represents a lower bound because when nodes are sparse in the network, the number of hops will inevitably increase due to having to route through longer distances to reach a certain node. When $a = b$, the expected number of hops, as depicted in [35], is

$$E[H] = 0.521a/r_0 \tag{9}$$

In order to calculate the average delay of the HiMAC encryption and decryption operations, we benchmark the delay of Crypto++ RSA using the same server that we will use to carry out the simulations in the next section. **Table 1** shows the benchmarking results, where TE_{RSA} and TD_{RSA} are the encryption and decryption delays of a single RSA operation (*i.e.*, delay for encrypting/decrypting one block of data), while TS_{RSA} and TV_{RSA} are the delays for a single RSA signing

Table 1. Benchmarking RSA on R1208WT2GSR Intel Server System with 18 cores and 128 GB RAM while running Ubuntu server 14.04. The RSA key size was varied between 512 and 4096 bits. Each result is the average of 100 operations.

Milliseconds/Operation	512	1024	2048	3072	4096
TE_{RSA}	0.0035	0.0186	0.0742	0.2484	0.5776
TD_{RSA}	0.0591	0.3712	2.731	6.629	12.438
TS_{RSA}	0.0587	0.3702	2.716	6.581	12.252
TV_{RSA}	0.0035	0.0185	0.0739	0.247	0.5441

and verification operation, respectively. We notice in Table 1 that all delays increase as the key size increases.

Finally, we derive the average encryption and decryption delays of HiMAC. First, with respect to encryption, each node N_i (where N_0 is the source node and $\{N_i / i > 0\}$ is an intermediate node) will encrypt three different parts as explained before. The first part is the one that contains the message from the previous node (or the plain-text message at the source node), in addition to the signature and the timestamp. The size of this part was derived to be equal to $S_{d(i+1)}$. The second part is the list of IDs, which will have a size equal to ($2 \times E[H]$). As for the third part, which is the encryption of the AES session key with the RSA public key, it will take a delay equal to TE_{RSA}. Hence, the average encryption delay of HiMAC can be calculated as:

$$\begin{aligned} TE_{HiMAC} = TS_{RSA} &+ \frac{1}{(E[H]+1)} \sum_{i=0}^{E[H]} \left(\left\lceil \frac{(8 \times S_{d(i+1)})}{128} \right\rceil \times \frac{T_{AES-Enc}}{C_p} \right) \\ &+ \left(\left\lceil \frac{(8 \times (2 \times (E[H]+1)))}{128} \right\rceil \times \frac{T_{AES-Enc}}{C_p} \right) + TE_{RSA} \end{aligned} \tag{10}$$

With respect to the decryption operation, which will be performed at the destination node, its equation can be derived similarly to that of the encryption delay, with a main difference that the encryption delay is calculated as the average of the encryption operations at all intermediate nodes (in addition to the source node), while the decryption delay is calculated as the sum of all decryption operations at the destination node (where each decryption operation is the decryption of one of the encryption operations at a single node). Hence, the average expected decryption delay is:

$$\begin{aligned} TD_{HiMAC} = (E[H]+1) \times & \left(TD_{RSA} + TV_{RSA} + \left(\left\lceil \frac{(8 \times (2 \times (E[H]+1)))}{128} \right\rceil \times \frac{T_{AES-Dec}}{C_p} \right) \right) \\ &+ \sum_{i=E[H]}^{0} \left(\left\lceil \frac{(8 \times S_{d(i+1)})}{128} \right\rceil \times \frac{T_{AES-Dec}}{C_p} \right) \end{aligned} \tag{11}$$

As can be deduced from Equations ((10) and (11)), the encryption and decryption delays of HiMAC depend on several parameters: the plain-text message size S_{d0}, the network dimension a, the transmission range r_0 (from $E[H]$), $T_{AES-Enc}$, $T_{AES-Dec}$, and C_p, and $\{ TE_{RSA}, TD_{RSA}, TS_{RSA}, TV_{RSA} \}$. In Figure 8, we show the effect of varying S_{d0} between 0.1 and 100 Kilobytes on the encryption and decryption delays. To be consistent with the simulations in the next section, we set a to 1000 m and r_0 to 100 m. We also use a key size equal to 1024 bits for RSA and session key size equal to 128 bits for AES. The values of $T_{AES-Enc}$ and $T_{AES-Dec}$ were calculated similar to the approach in [34] for a 128-bit AES key. On the other hand, C_p was calculated for the R1208WT2GSR

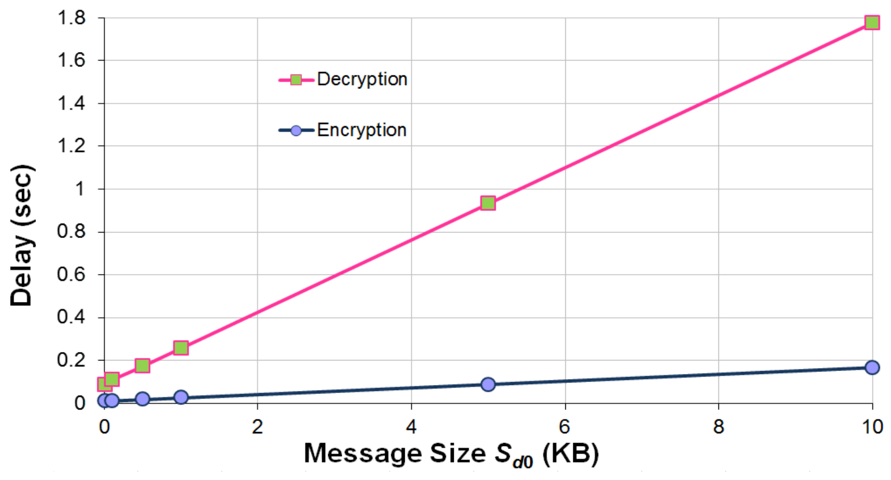


Figure 8. Encryption and decryption delays of HiMAC calculated using Equations (10) and (11) for different values of the plain-text message size S_{d0}.

Intel Server that we used in the simulations and found to be equal to 250 KMIPS.

We notice from Figure 8 that the encryption and decryption delays of HiMAC increase as the size of the plain-text message S_{d0} increases. This can be expected since in this case the message will be divided by AES into a higher number of blocks and each block will be separately encrypted/decrypted, which will increase the delay. We also notice that HiMAC decryption delay is much larger than the encryption delay, due to the fact that the encryption delay is calculated as the average of the encryption delays at the source and intermediate nodes. On the other hand, the decryption delay is calculated at the destination node and includes the decryption of all encryptions that were done by the source and intermediate nodes, which will be much greater than a single encryption at one node. Finally, we notice that the decryption delay starts to increase heavily when S_{d0} increases above 5 KB. Hence, we can deduce that when the size of the plain-text message is large, it is better to divide it into multiple messages and encrypt/send each message separately to avoid large decryption delay at the destination node. In the next section, we compare the experimental simulations encryption/decryption delays with the analytical delays from Equations ((10) and (11)).

5. Experimental Evaluation

We used the network simulation ns2 software (version 2.35) to test the performance of HiMAC. The operations of HiMAC were implemented as functions that were integrated within the AODV routing protocol, although these functions could be integrated within any MANET routing protocol. We choose AODV in order to compare the performance of our system with that of secure AODV (SAODV) [29], which is a secure data dissemination protocol that was extensively studied and enhanced in the literature. SAODV provides integrity, authentication and non-repudiation of routing information. Two mechanisms are used to secure AODV messages, digital signatures to authenticate the non-mutable fields of AODV messages so that every message is signed, and hash chains to secure the hop count information of AODV messages in such a way that allows every node that receives the message to verify that the hop count has not been decremented by an attacker. We implemented an enhanced version of SAODV, whose authors called adaptive SAODV or A-SAODV [36]. A-SAODV optimizes the performance of SAODV by performing the cryptographic operations via a dedicated thread to avoid blocking the processing of other messages. In A-SAODV, Every node has a queue of routing messages to be signed or verified and the length of the queue implies the load state of the routing thread. Whenever a node processes a route request and has enough information to generate a RREP on behalf of the destination, it first checks its routing message queue length. If the length of the queue is below a threshold, then it replies, otherwise, it forwards the RREQ without replying. The value of threshold can be modified during execution. This adaptive reply decision has a significant improvement on the performance of SAODV. Also, A-SAODV maintains a cache of latest signed and verified messages in order to avoid signing and verifying the same message twice.

5.1. Simulation Setup

We compare HiMAC with A-SAODV: both were implemented within AODV, and both were tested using the setup that will be explained next. The simulated network has a topography size equal to 1000 × 1000 m^2. The wireless bandwidth and the transmission range were assumed to be 6 Mbps and 100 m, respectively. The mobile nodes were randomly distributed in the topography and followed the Random Waypoint (RWP) movement model. We simulated six scenarios, in which the number of mobile nodes in the network was set to 50, 100, 200, 300, 400, and 500 respectively. Each scenario was repeated ten times for both protocols, and the reading of the ten repetitions were averaged to calculate the results. The mobility of the nodes in the network followed the RWP model in which the minimum and maximum speeds of the nodes (V_{min} and V_{max}) were set to 0.01 and 2.00 m/s, respectively, and the pause time to 100 seconds. Each scenario lasted for 4000 seconds, in which the first 200 seconds were used to setup the network, the security requirements (key exchange), and the routing tables. After the 200 seconds, each node in the network sends a data packet to another random node using the tested security protocol. The size of a data packet (payload) was set to 10 KB, and the time interval between two data packets was set to 100 seconds. A summary of the simulation parameters can be found in **Table 2**.

The cryptographic operations of HiMAC were implemented using the Crypto++ library [30]. The widely used RSA algorithm was used for the encryption and decryption of messages. The size of public and private keys used for encryption, decryption, signing, and verification was set to 1024 bits. The Crypto++ library was integrated into ns2 and its necessary classes were used within AODV. The attack model was simulated as follows: in each scenario, the nodes were divided randomly into 2 groups. The first group constitutes m percent of the total

Table 2. Simulation parameters.

Simulation Parameter	Parameter Value
Simulation Time	4000 (seconds)
Network Dimensions	1000×1000 m^2
Wireless Bandwidth	6 Mbps
Number of nodes	50 → 500
Packet Request Rate (at each node)	1 Packet/100 sec
Data Packet Payload Size	10 KB
Node Transmission Range	100 m
Routing Protocol	HiMAC enhanced AODV
Node mobility model	Random Way Point
Node Speed (v)	0.01 → 2 m/s
Maximum Queue Size	200

number of nodes (N), and its members are malicious nodes who listen to packets and forward them without applying the security mechanism. The second group constitutes (N-m) percent of all nodes, and whose members are recognized as trusted by all nodes in the second group. Each time an attacker N_A listens to a packet that is sent to a destination D, it adds random data to the packet, increment the hop count, and forward the packet to one of the trusted node. When D receives the packet, it will detect that the packet has been tampered since the decryption algorithm will fail to decrypt the data that was added by the attacker. In the simulations, the value of m was set to 10%.

The metrics used for evaluating the two compared protocols are:

1) Latency: we tested three different delays: a) Encryption delay (D_{enc}), which is the average time needed to perform a single encryption operation in HiMAC or A-SAODV, b) Decryption delay (D_{dec}), which is the average time needed to perform a decryption operation in HiMAC or A-SAODV, and c) End-to-End delay (D_{EtE}), which is the time between the instance a node generates a new packet (before encrypting it) to the instance that the destination decrypts the packet and recovers the data. The definitions of D_{enc} and D_{dec} for HiMAC are the same as those of TE_{HiMAC} and TE_{HiMAC} in Section 4.

2) Success ratio (R_S): the average percentage of data packets (per node) that are successfully received at their destinations.

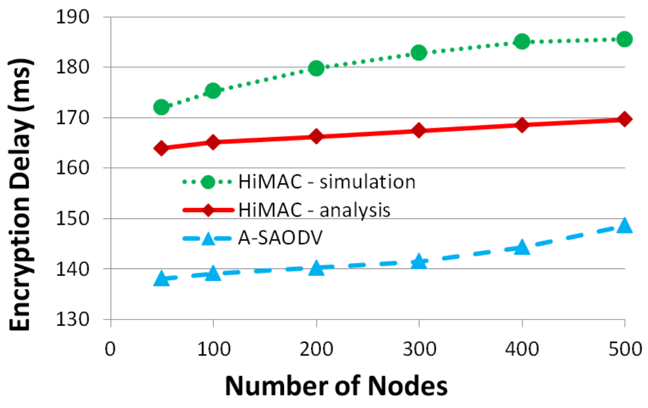
3) Hop count (H_C): the average number of hops a data packet traverses before it reaches its destination (*i.e.*, the number of times a data packet is forwarded by intermediate nodes).

4) Queue Length (L_Q): the average length of the queue of packets at each node (the queue will contain packets that are waiting to be forwarded to the next hop).

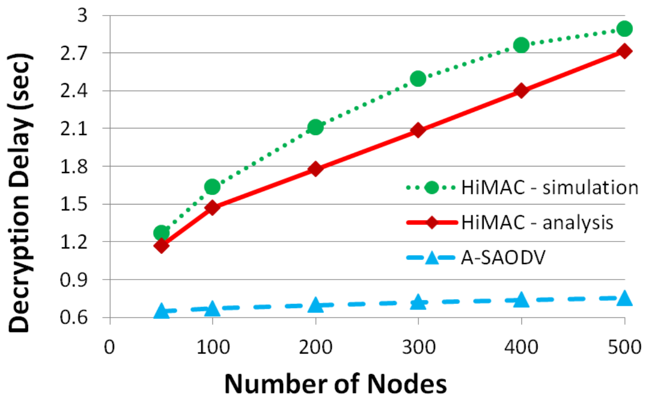
5) Network Traffic (Traff): the total number of data packets sent, forwarded, or received per node during the simulation time.

5.2. Simulation Results

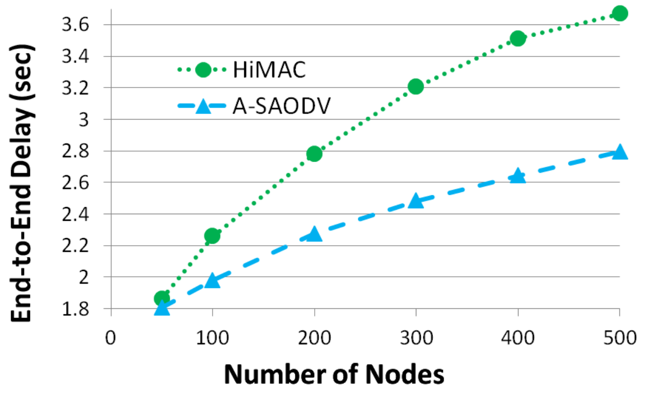
Figure 9 presents the encryption, decryption, and End-to-End delays of HiMAC and A-SAODV. The encryption delay was calculated as follows: each time a node generates a data packet, it encrypts both ($M_p \| MAC_{KPR_S}(M_p) \| T_s$) and (ID_S). The total time of both encryptions is saved to a log file. Also, each time an intermediate node encrypts both parts of the packet as explained in Section 3, the total encryption time is also recorded into the log file. At the end, the average



(a)



(b)



(c)

Figure 9. Latency of HiMAC and A-SAODV: (a) Encryption delay; (b) Decryption delay; and (c) End-to-End delay while varying the number of nodes in the network.

of all encryptions is calculated for each scenario. With respect to decryption, each time a node receives a packet destined to it; it executes the hierarchical decryption algorithm described in Section 3. The decryption delay D_{dec} is the total time from the instance the destination node starts decrypting the outer encrypted layer of the packet body, until it recovers the original data message M_p. As for the End-to-End delay, it is calculated as follows: each time a source node generates a new data packet, it records the time at the instance before encrypting the packet, and includes this time (T_i) in the packet (unencrypted). After the destination decrypts the packet and recovers M_p, it subtracts T_i from the current time to calculate D_{EtE}. Similarly in A-SAODV, the encryption and decryption delays were calculated for each instance a node performs an encryption or decryption of the packet payload, and D_{EtE} was calculated similar to HiMAC.

From **Figure 9**, we notice that the encryption and decryption delays of HiMAC are much higher than those of A-SAODV. However, the End-to-End delay of HiMAC is slightly higher than that of A-SAODV. First, the encryption delay of HiMAC is higher than that of A-SAODV since the former involves two encryption parts: that of $(M_p \| MAC_{KPR_S}(M_p) \| T_s)$ and of (ID_S), while that of A-SAODV doesn't include the second part. As we stated in Section 3, we encrypted the list of traversed nodes to prevent an attacker from tampering the hop count without being detected by the destination. The decryption delay of HiMAC is also much higher than that of A-SAODV (**Figure 9(b)**) since HiMAC decryption process includes a hierarchy of decryption operations, while that of A-SAODV includes a single decryption operation. The overhead in encryption and decryption delays that occurs in HiMAC is compensated by the fast communication delay that can be deduced from **Figure 9(c)**. The reason why the End-to-End delay of HiMAC is only slightly greater than that of A-SAODV although it is cryptographically much slower is that it requires much less communication time and overhead. This happens because A-SAODV doesn't include a trust mechanism in which the "next hop" decision is based on trust in addition to route efficiency. In addition, the adaptive "*double signature*" process in A-SAODV required a lot of time to obtain the destination signature before a secure route can be established and before forwarding the data packet. Also, the separation of routing and data packets queues requires that a data packet will wait for the route to be established before it can be forwarded, which will take a lot of time if the routing queue is overwhelmed. In general, the slight overhead in D_{EtE} is compensated in HiMAC by the advantages that each node on the message path is trusted and that the destination nodes makes sure of the integrity of each intermediate node that forwarded the message. Finally, we notice from **Figure 9** that the analysis encryption/decryption delays that were calculated from Equations ((10) and (11)) have similar values to the simulation encryption/decryption delays, which reflects the accuracy of the analysis equations in depicting the most important parameters that affect the delays.

These observations are verified by the results in **Figure 10** which show that

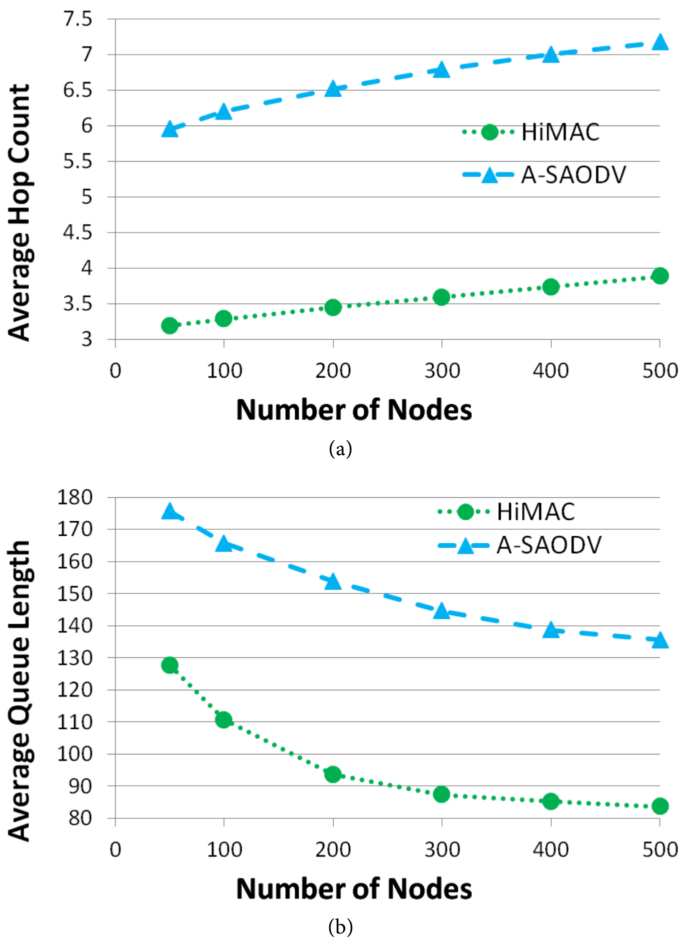
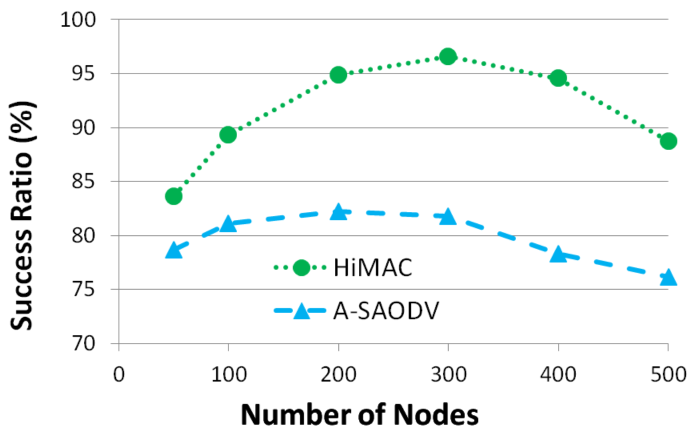


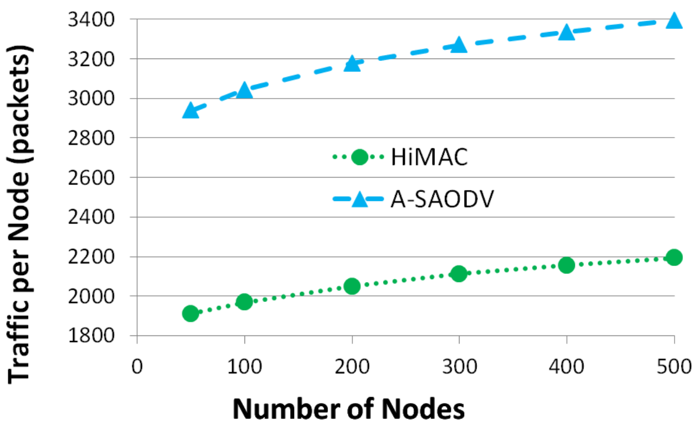
Figure 10. (a) Average hop count and (b) Average queue length of HiMAC and A-SAODV while varying the number of nodes in the network.

HiMAC has a less hop count and smaller packet queue size on average than A-SAODV. While A-SAODV relies on securing RREQ and RREP to establish secure path, which requires routing overhead; HiMAC establishes secure routing dynamically using the proposed trust mechanism that was described in section 3. The process of choosing the next hop based on "trust" value and on ordinary AODV routing data, reduces the overhead required to establish secure route and decreases the number of hops to the destination, as the packet will remain a much less time waiting in the queue, which gives it a higher probability to find the optimal route and at the same time reaches the destination faster. From Figure 10(a), the average hop count of HiMAC is about 3.5 while that of A-SAODV is about 6.6. Also, from Figure 10(b), the average queue length in HiMAC is 98 packets while that in A-SAODV is 152 packets, which shows that a packet will stay in the queue less time in HiMAC as compared to A-SAODV.

The fact that a packet stays in the queues for less time on average (Figure 10(b)) and traverses less nodes on its way to the destination (Figure 10(a)), leads to a higher success ratio, as can be seen in Figure 11(a). The success ratio of HiMAC reaches a maximum of 96% when the number of nodes is equal to



(a)



(b)

Figure 11. (a) Average success ratio and (b) Average traffic per node of HiMAC and A-SAODV while varying the number of nodes in the network.

300 and has an average value of 91%, as compared to A-SAODV which has an average success ratio of 79% and a maximum value of 82%. A packet doesn't reach its destination when it is dropped for any reason, such as when it stays for a long time in the queue, when the queue is full, or when no route can be found to the destination. The last reason is common between the two protocols, which makes us deduce that the first two reasons cause the difference between the success ratios of the two protocols. The packet queues in A-SAODV will be more full on average than those in HiMAC, which leads to more packets dropped for this reason. Also, a packet will stay in the A-SAODV queue for longer periods than HiMAC, which causes more packets to be dropped for queue TTL expiry. These two reasons explain the higher success ratio of HiMAC.

Finally, Figure 11(b) illustrates that A-SAODV creates much more traffic in the network than HiMAC. The traffic shown in the figure is the average number of data packets created (and sent), forwarded, or received at a node during the simulation time (average for all nodes). The main difference which causes A-SAODV to create much more traffic is the number of forwarded packets, which was calculated in the simulation scenarios and found to be much higher in A-SAODV than HiMAC. This is mainly because the number of hops traversed by a data packet is higher by 2.5 (on average) in A-SAODV than HiMAC. Hence, the total number of forwarded packets is higher. Also, the success ratio is lower in A-SAODV, which causes more data packets to be resent after they fail to reach the destination, which creates an additional number of forwarded packets.

In general, we deduce that HiMAC produces very good performance although it contains heavy cryptographic operations. The high processing overhead caused by multiple encryption and decryption operations per packet is compensated by the trust mechanism, which enables a node to dynamically choose the next node on the path, without maintaining a secure route all the time which puts additional overhead on the routing protocol. This overhead is eliminated in HiMAC by means of dynamically established secure-route based on trust mechanism. In addition, HiMAC eliminates the possibility of any attack by an intermediate node on the routing path, by authenticating each intermediate node at the destination. HiMAC will operate well as long as the data message size is not very large. As can be deduced from the analysis and simulation results, the encryption/decryption and End-to-End delays of HiMAC will increase with the message size. After a certain size limit, the high security advantages offered by HiMAC will not compensate the increased overhead delay of the encryption/decryption operations.

6. Conclusion and Future Work

We presented a trust-based mechanism with hierarchical Message Authentication Code solution for securing data dissemination in Mobile Ad hoc Networks. Our approach depends on establishing the trust between mobile nodes based on the knowledge and expertise of users and based on the nodes' behaviors. Our proposed HiMAC prevents attackers from tampering data packets or modifying their hop count by means of signing and encrypting the packet at each intermediate node. Our system dynamically establishes a route based on the "trust" of neighbors and also based on securing routing data in order to prevent malicious nodes from inserting false data into trusted nodes' routing tables. We tested our proposed system by comparing its performance to another security mechanism for MANETs. The results illustrated that HiMAC performs better than A-SAODV, although it has high overhead in cryptographic operations. For future work, we will study how to decrease the cryptographic overhead produced by HiMAC while maintaining secure data transmission. One possible direction is to find the optimal key size that reduces the encryption and decryption delays and at the same time keeps the message secure; since a hierarchical encryption operation is done, there is no need for a very large key since the attacker needs several decryption stages to reach the original message.

References

[1] Hu, Y.C., Perrig, A. and Johnson, D.B. (2002) Ariadne: A Secure On-Demand

Routing Protocol for Ad Hoc Networks. *Wireless Networks*, **11**, 21-38. https://doi.org/10.1145/570645.570648

[2] Chlamtac, I., Conti, M. and Liu, J.J.N. (2003) Mobile Ad Hoc Networking: Imperatives and Challenges. *Ad Hoc Networks*, **1**, 13-64. https://doi.org/10.1016/S1570-8705(03)00013-1

[3] Panagiotis, P. and Zygmunt, J.H. (2003) Secure Data Transmission in Mobile Ad Hoc Networks. *Proceedings of the* 2003 *ACM Workshop on Wireless Security*, 41-50.

[4] Papadimitratos, P. and Haas, Z.J. (2003) Secure Message Transmission in Mobile Ad Hoc Networks. *Ad Hoc Networks*, **1**, 193-209. https://doi.org/10.1016/S1570-8705(03)00018-0

[5] Hu, Y.C., Johnson, D.B. and Perrig, A. (2003) SEAD: Secure Efficient Distance Vector Routing for Mobile Wireless Ad Hoc Networks. *Ad Hoc Networks*, **1**, 175-192. https://doi.org/10.1016/S1570-8705(03)00019-2

[6] Wu, B., Chen, J., Wu, J. and Cardei, M. (2007) A Survey of Attacks and Countermeasures in Mobile Ad Hoc Networks. In: *Wireless Network Security*, Springer, Boston, 103-135. https://doi.org/10.1007/978-0-387-33112-6_5

[7] Li, J.S. and Lee, C.T. (2006) Improve Routing Trust with Promiscuous Listening Routing Security Algorithm in Mobile Ad Hoc Networks. *Computer Communications*, **29**, 1121-1132. https://doi.org/10.1016/j.comcom.2005.06.025

[8] Komninos, N., Vergados, D. and Douligeris, C. (2006) Layered Security Design for Mobile Ad Hoc Networks. *Computers & Security*, **25**, 121-130. https://doi.org/10.1016/j.cose.2005.09.005

[9] Xiaopeng, G. and Wei, C. (2007) A Novel Gray Hole Attack Detection Scheme for Mobile Ad-Hoc Networks. *Proceedings of the IFIP International Conference on Network and Parallel Computing Workshops*, 209-214. https://doi.org/10.1109/NPC.2007.88

[10] Su, M.Y. (2011) Prevention of Selective Black Hole Attacks on Mobile Ad Hoc Networks through Intrusion Detection Systems. *Computer Communications*, **34**, 107-117. https://doi.org/10.1016/j.comcom.2010.08.007

[11] Bhalaji, N. and Shanmugam, A. (2012) Dynamic Trust Based Method to Mitigate Greyhole Attack in Mobile Ad Hoc Networks. *Procedia Engineering*, **30**, 881-888. https://doi.org/10.1016/j.proeng.2012.01.941

[12] Mavropodi, R., Kotzanikolaou, P. and Douligeris, C. (2007) SecMR—A Secure Multipath Routing Protocol for Ad Hoc Networks. *Ad Hoc Networks*, **5**, 87-99. https://doi.org/10.1016/j.adhoc.2006.05.020

[13] Komninos, N., Vergados, D.D. and Douligeris, C. (2007) Authentication in a Layered Security Approach for Mobile Ad Hoc Networks. *Computers & Security*, **26**, 373-380. https://doi.org/10.1016/j.cose.2006.12.011

[14] Zhao, S., Aggarwal, A., Liu, S. and Wu, H. (2008) A Secure Routing Protocol in Proactive Security Approach for Mobile Ad-Hoc Networks. *Proceedings of the IEEE Wireless Communications and Networking Conference*, 2627-2632. https://doi.org/10.1109/WCNC.2008.461

[15] Marchang, N. and Datta, R. (2008) Collaborative Techniques for Intrusion Detection in Mobile Ad-Hoc Networks. *Ad Hoc Networks*, **6**, 508-523. https://doi.org/10.1016/j.adhoc.2007.04.003

[16] Yeun, C.Y., Han, K., Vo, D.L. and Kim, K. (2008) Secure Authenticated Group Key Agreement Protocol in the MANET Environment. *Information Security Technical*

Report, **13**, 158-164. https://doi.org/10.1016/j.istr.2008.10.002

[17] Cordasco, J. and Wetzel, S. (2008) Cryptographic versus Trust-Based Methods for MANET Routing Security. *Electronic Notes in Theoretical Computer Science*, **197**, 131-140. https://doi.org/10.1016/j.entcs.2007.12.022

[18] Kim, J. and Tsudik, G. (2009) SRDP: Secure Route Discovery for Dynamic Source Routing in MANETs. *Ad Hoc Networks*, **7**, 1097-1109. https://doi.org/10.1016/j.adhoc.2008.09.007

[19] Dutta, R., Mukhopadhyay, S. and Collier, M. (2010) Computationally Secure Self-Healing Key Distribution with Revocation in Wireless Ad Hoc Networks. *Ad Hoc Networks*, **8**, 597-613. https://doi.org/10.1016/j.adhoc.2009.11.005

[20] Su, M.Y. (2010) WARP: A Wormhole-Avoidance Routing Protocol by Anomaly Detection in Mobile Ad Hoc Networks. *Computers & Security*, **29**, 208-224. https://doi.org/10.1016/j.cose.2009.09.005

[21] Makri, E. and Konstantinou, E. (2011) Constant Round Group Key Agreement Protocols: A Comparative Study. *Computers & Security*, **30**, 643-678. https://doi.org/10.1016/j.cose.2011.08.008

[22] Khalil, I., Awad, M. and Khreishah, A. (2012) CTAC: Control Traffic Tunneling Attacks' Countermeasures in Mobile Wireless Networks. *Computer Networks*, **56**, 3300-3317. https://doi.org/10.1016/j.comnet.2012.06.003

[23] Singh, G., Kumar, N. and Verma, A.K. (2012) Ant Colony Algorithms in MANETs: A Review. *Journal of Network and Computer Applications*, **35**, 1964-1972. https://doi.org/10.1016/j.jnca.2012.07.018

[24] Omar, M., Boufaghes, H., Mammeri, L., Taalba, A. and Tari, A. (2016) Secure and Reliable Certificate Chains Recovery Protocol for Mobile Ad Hoc Networks. *Journal of Network and Computer Applications*, **62**, 153-162. https://doi.org/10.1016/j.jnca.2016.01.007

[25] Borkar, G.M. and Mahajan, A.R. (2017) A Secure and Trust Based On-Demand Multipath Routing Scheme for Self-Organized Mobile Ad-Hoc Networks. *Wireless Networks*, **23**, 2455-2472. https://doi.org/10.1007/s11276-016-1287-y

[26] Dhyani, I., Goel, N., Sharma, G. and Mallick, B. (2017) A Reliable Tactic for Detecting Black Hole Attack in Vehicular Ad Hoc Networks. In: *Advances in Computer and Computational Sciences*, Springer, Singapore, 333-343. https://doi.org/10.1007/978-981-10-3770-2_31

[27] Mershad, K. and Artail, H. (2013) A Framework for Secure and Efficient Data Acquisition in Vehicular Ad Hoc Networks. *IEEE Transactions on Vehicular Technology*, **62**, 536-551. https://doi.org/10.1109/TVT.2012.2226613

[28] Jan, M., Nanda, P., Usman, M. and He, X. (2017) PAWN: A Payload-Based Mutual Authentication Scheme for Wireless Sensor Networks. *Concurrency and Computation: Practice and Experience*, **29**, 64. https://doi.org/10.1002/cpe.3986

[29] Zapata, M.G. (2002) Secure Ad Hoc On-Demand Distance Vector Routing. *ACM SIGMOBILE Mobile Computing and Communications Review*, **6**, 106-107. https://doi.org/10.1145/581291.581312

[30] Crypto++ Library (2017). http://www.cryptopp.com/

[31] Crypto++ Benchmarks (2017). https://www.cryptopp.com/benchmarks.html

[32] Hoffmann, J., Uellenbeck, S. and Holz, T. (2012) SMARTPROXY: Secure Smartphone-Assisted Login on Compromised Machines. *Proceedings of the International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, 184-203.

[33] Doomun, M.R. and Soyjaudah, K.M.S. (2009) Analytical Comparison of Cryptographic Techniques for Resource-Constrained Wireless Security. *International Journal of Network Security*, **9**, 82-94.

[34] Xenakis, C., Laoutaris, N., Merakos, L. and Stavrakakis, I. (2006) A Generic Characterization of the Overheads Imposed by IPsec and Associated Cryptographic Algorithms. *Computer Networks*, **50**, 3225-3241. https://doi.org/10.1016/j.comnet.2005.12.005

[35] Bettstetter, C. and Eberspacher, J. (2003) Hop Distances in Homogeneous Ad Hoc Networks. *Proceedings of the* 57*th IEEE Semiannual Vehicular Technology Conference*, Vol. 4, 2286-2290. https://doi.org/10.1109/VETECS.2003.1208796

[36] Cerri, D. and Ghioni, A. (2008) Securing AODV: The A-SAODV Secure Routing Prototype. *IEEE Communications Magazine*, **46**, 120-125. https://doi.org/10.1109/MCOM.2008.4473093