Intrusion Detection and Response in Ad Hoc Networks

AIKATERINI MITROKOTSA[1], NIKOS KOMNINOS[2], CHRISTOS DOULIGERIS[1]

[1] Department of Informatics, University of Piraeus,
80 Karaoli & Dimitriou Str. Piraeus, 18534, GREECE
{mitrokat,cdoulig}@unipi.gr

[2] Athens Information Technology,
19002 Peania Attiki, GREECE
nkom@ait.edu.gr

Abstract. Ad hoc networks have received great attention in recent years, mainly due to their important advantages and their growing demand. Nevertheless, ad hoc networks present many inherent vulnerabilities and demand efficient security mechanisms in order to be safeguarded. An efficient intrusion detection approach combined with an authenticated intrusion response can act as an effective line of defense in ad hoc networks. In this paper, we present an intrusion detection engine based on neural networks and an authenticated intrusion response engine based on an innovative key agreement protocol. In particular, we exploit information visualization and machine learning techniques in order to achieve efficient intrusion detection and we propose an effective intrusion response engine that is based on secure communication principles in order to mitigate the impacts of possible attacks. The performance of the proposed model is evaluated under different traffic conditions and mobility patterns.

Key-Words: Intrusion Detection, Intrusion Response, Neural Networks, Ad Hoc networks, Key Agreement Protocols, Network Security

1. Introduction

Ad hoc networks are being adopted in a broad range of environments making imminent their rapid proliferation. Ad hoc networks can be defined as dynamic peer-to-peer networks (DPNs) that are composed of a collection of nodes, which employ a multi-hop information transfer in a self-organized way without relying on an a-priori infrastructure. In ad hoc networks all nodes are able to appear in and disappear from the network at any time. Because of the special advantages that ad hoc networks present, they are envisioned in many critical applications, including battlefields and disaster recovery applications.

There are many “flavours” of ad hoc networks. In this paper, we refer to an open ad hoc network composed of a set of nodes that are characterized by short membership duration since there is a large number of joining and leaving events. The ad hoc nodes have a short range and low power transmission. We should stress that the type of communication medium in an ad hoc network is not as important as the type of the network and the lack of infrastructure. The communication medium may vary from infrared, laser, radio to dynamic wire and fiber-optic links since even fixed wired nodes may join the ad hoc network.

Ad hoc networks, have many security requirements. The task of safeguarding them, is very challenging, since it demands efficient and effective protocols and mechanisms. Although ad hoc networks are characterized by great flexibility, they also present many inherent limitations including the lack of central control entities and the dynamically changing topology leading to a not well-defined boundary where access control mechanisms and firewalls can be applied. Their dynamically changing topology in combination with the lack of centralized control, the lack of trusted third parties, the resetting of connections for various reasons, the ease of interception of messages combined with the constrained resources are some of the security issues that we are called to combat. In order to achieve security in ad hoc networks apart from prevention techniques the use of reactive mechanisms, as a second wall of defense, is a necessity. Intrusion detection and intrsuion response systems are indispensable in order to achieve reliable communication.

Intrusion Detection is an invaluable mature arsenal with a long history of research for the defense of wired networks but it is still in its infancy in the area of ad hoc networks. In this paper, we present an intrusion detection module that is part of an Intrusion Detection System (IDS) architecture composed of a data collection engine, an intrusion detection engine and an Intrusion Response engine. The focus of this paper is the intrusion detection engine that is based on a type of neural networks known as emergent Self-Organizing Maps (eSOMs) and the design of the intrusion response engine. We exploit the great advantage of tolerance towards imprecise data of neural networks in order to classify normal against abnormal behavior. By combining machine learning, information visualization and key agreement techniques we are able to have a clear view of how secure an ad hoc network is against attacks. In particular, each node of the ad hoc network creates a map that depicts its security state and distributes this map to all its neighboring nodes. Thus, each node knows the security status of every neighbor by generating a global map. The global map is used in order to perform secure and efficient routing by avoiding paths that include nodes which are victims of attacks.

Neural Networks is a research area with many advantages that have not been exploited in ad hoc networks. We exploit their main advantages in the design of an intrusion detection engine, which is part of a local IDS agent. Using eSOMs, we have a visual representation of the normal-attack state in each node of the ad hoc network. Hence, each node can determine whether a neighboring node is under attack and forward its messages accordingly. Moreover, using an authenticated intrusion response engine based on an efficient key agreement protocol, we can instantly and securely notify the nodes of the ad hoc network about the possible existence of an attack and to mitigate the impact of the attack.

Following this introduction, the paper is organized as follows. Section 2 presents related work of intrusion detection approaches that have been proposed for ad hoc networks and approaches that use key agreement protocols. Section 3 discusses the intrusion detection model this paper is based on. Section 4 presents a functional description of the proposed intrusion detection engine and the classification algorithm that was used. Section 5 presents the intrusion response engine and a key agreement protocol that is necessary for the secure transfer of the eSOM maps produced by the ad hoc nodes. In section 6 the performance evaluation of the detection engine is presented, while section 7 concludes the paper and discusses future work.

2. Related Work

Intrusion Detection is an active and mature research area for wired networks but techniques designed for wired networks may not be efficient if applied to wireless ad hoc networks due to the stringent requirements these networks present. Compared to wired networks, where traffic monitoring is performed in gateways, routers and switches, wireless ad hoc networks lack centralized choke points at which it would be possible to monitor network traffic. Even if we could achieve the existence of such concentration points, their locations would continuously change due to mobility. This is the reason why the deployment of a distributed intrusion detection approach in wireless ad hoc networks is a necessity. Additionally, we should focus on security mechanisms keeping in mind the ease of listening to wireless transactions on which most ad hoc networks are based, the lack of a fixed infrastructure and the resource consumption characteristics of ad hoc networks. This means that it is better to use a periodic intrusion detection system (IDS) than an 'always-on' prevention mechanism. Moreover, the resource constraints that ad hoc networks face, including limited battery capabilities, strict bandwidth requirements and frequent miscommunication, complicate the discrimination between a new qualified operation after a disconnection and an intrusion, a fact that makes even more difficult the classification between normal and anomaly behavior ([1], [2]).

Zhang and Lee [3] proposed the first (high-level) IDS approach specific for ad hoc networks. They proposed a distributed and cooperative anomaly-based IDS, which provides an efficient guide for the design of IDS in wireless ad hoc networks. They focused on an anomaly detection approach based on routing updates on the MAC layer and on the mobile application layer.

Huang and Lee [4] extended their previous work by proposing a cluster-based IDS, in order to combat the resource constraints that MANET face. They use a set of statistical features that can be derived from routing tables and they apply the classification decision tree induction algorithm C 4.5 in order to detect normal vs. abnormal behavior. The proposed system is able to identify the source of the attack, if the identified attack occurs within one-hop.

Deng et al. [5] proposed a hierarchically distributed and a completely distributed intrusion detection approach. The intrusion detection approach used in both of these architectures focuses on the network layer and it is based on a Support Vector Machines (SVM) classification algorithm. They use a set of parameters derived from the network layer and suggest that a hierarchically distributed approach may be a more promising solution versus a completely distributed intrusion detection approach.

Kachirski and Guha [6] proposed a cluster-based intrusion detection system built on a mobile agent framework. The proposed system uses mobile agents, each performing a particular role, either monitoring, or decision or action. A few nodes are chosen by a distributed algorithm in order to host sensors for the monitoring of network packets and agents in order to make the decisions. Additionally, all the nodes host sensors for host-based monitoring. The main advantage of this approach is that the packet-monitoring task is limited in a few nodes and the IDS-related processing time by each node is minimized.

Liu et al. [7] proposed a completely distributed anomaly detection approach. They investigated the use of the MAC layer in order to profile normal behavior of mobile nodes and then apply cross-feature analysis [8] on feature vectors constructed from the training data.

Tseng et al. [9] proposed a distributed specification-based intrusion detection approach in order to detect attacks in the Ad-hoc On-demand Distance Vector (AODV) routing protocol. The approach involves the use of finite state machines. More specifically a correct AODV routing behavior is specified using finite state machines and the actual behavior of AODV flows is compared to these specifications. Any deviation from these specifications is recognized as an intrusion. Specification based techniques have the drawback that it is necessary to balance the tradeoff between complexity and accuracy.

Anjum et al. [10] proposed a signature-based intrusion detection approach for wireless ad hoc networks based on the assumption that attack signatures are completely known in an ad hoc network. This approach investigates the ability of various routing protocols to facilitate the intrusion detection procedure. The authors show that reactive ad-hoc routing protocols are less effective than proactive routing protocols in the detection of intrusions even in the absence of mobility.

Chen et al. [11] proposed a distributed intrusion detection approach based on the Dempster-Shafer theory. They exploit the main advantages of this theory and its ability to reflect uncertainty or a lack of complete information and the convenient numerical procedure for fusing together multiple pieces of data.

Effective intrusion detection should be combined with an efficient and secure intrusion response. Intrusion response in order to be efficient and secure should be based on security mechanisms such as group key management and agreement. Many group key management protocols ([12], [13], [14], [15]) have been proposed for wired networks, including a great number that rely of tree-based structure ([14], [16], [17], [18], [19]). However, most of these protocols can not be applied in an infrastructure-less or resource sensitive environment such as ad hoc networks.

For example, in the Octopus protocol [13], four nodes compose a 2^2 −cube and the rest members of the network are "tentacles" that are connected to one of the central nodes. In a dynamic ad hoc network, it is not easy to maintain such a topology. The Tree-Group Diffie-Hellman (TGDH) proposed by [14], which is based on a binary tree structure and improves the performances of the IKA1/2 [15], is based on modular exponentiation which is the most expensive operation thus, it might require O(n) exponentiations in order to compute the group session key.

Hwang and Chang [20] have proposed a key agreement protocol that is based on a shared group password, the exclusive-OR (XOR) operation and a binary tree structure. Although this approach is really efficient for ad hoc networks it is susceptible to password guessing and replay attacks. Lo [21] et. al. have improved the previous approach [16] by adding mutual authentication and adding a procedure for periodically session key updates. However, their approach is still susceptible to dictionary and brute force attacks since it is based on passwords.

In this paper, we use a key agreement protocol in combination with eSOMs in order to ensure that the exploitation of the information visualization that eSOM provide, in each node in the ad hoc network, will not be altered by malicious attackers. In ad hoc networks the response to possible attacks should be quick, because due to the aforementioned resource constraints the impact of a possible intrusion would be even more severe. Information visualization can help us in order to have a direct response to possible intrusions. We exploit the advantages of information visualization by proposing an Intrusion Detection engine and an Intrusion Response engine which cooperate efficiently for the effective combat of possible intrusions.

The proposed Intrusion Detection engine is based on eSOMs while the proposed Intrusion Response engine is based on a key agreement protocol. The proposed authenticated group key agreement protocol has some unique characteristics that include the use of a shared master key and the structure of a rooted tree which depends on the distance between the ad hoc nodes and consequently a unique protocol flow. Using this key agreement protocol we are able to generate Local Keys (LK) and a Global Key (GK) in order to secure the communication of the proposed Intrusion Response engine.

3. Intrusion Detection Model

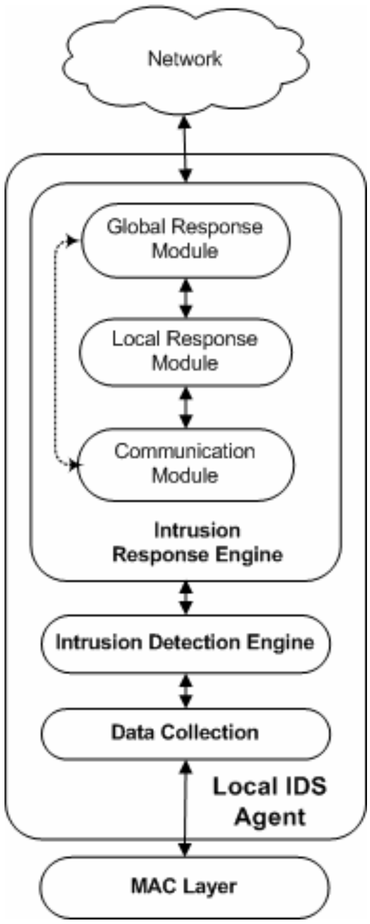


Fig 1. Intrusion Detection Architecture

The architecture of the IDS applied to ad hoc networks could be either distributed and cooperative or distributed and hierarchical. The lack of central monitoring nodes and the lack of trust between peer nodes of a wireless ad hoc network render a central intrusion detection system impractical. The distributed and hierarchical IDS are based on dividing the mobile ad hoc network in clusters. Although cluster-based IDS have the advantage of lower detection workload, the procedure of creating clusters and electing cluster heads may cause a large overhead. Moreover, the existence of cluster heads and the obvious possibility of their exploitation by malicious attackers weaken the expected security. Furthermore, the distributed hierarchical IDS are more efficient for ad hoc networks with low mobility. Thus, the cooperative and dynamic nature of ad hoc network implies that the intrusion detection system should be distributed and cooperative.

Each node of the ad hoc network must perform its local intrusion detection using local audit data. When the confirmation of other nodes to detect an attack is necessary, local intrusion detectors should cooperate. Furthermore, this cooperation between local IDS agents should be held through secure channels.

The IDS architecture we adopt is the one proposed in [3] and is composed of multiple local IDS agents, as illustrated in Figure 1, that are responsible for detecting possible intrusions locally. The collection of all the independent IDS agents forms the IDS system for the ad hoc network. Each local IDS agent is composed of the following components:

Data Collection: is responsible for selecting local audit data and activity logs.

Intrusion Detection engine: is responsible for detecting local anomalies using local audit data. The local anomaly detection is performed using the eSOM classification algorithm.

The procedure that is followed in the local detection engine is the one described below:

- Select labeled audit data and perform the appropriate transformations.
- Compute the classifier using training data and the eSOM algorithm.
- Apply the classifier to test local audit data in order to classify it as normal or abnormal.

We have to note here, that the integrity of the generated eSOM map is assured through the use of integrity mechanisms such as a MAC (Message Authentication Code) or a hash function. Thus, any possible attempt by malicious nodes to alter the eSOM maps will be detected.

Intrusion Response Engine: If the *Intrusion Detection Engine* detects an intrusion then the *Intrusion Response Engine* is activated. Every member node in the ad hoc network may participate in the *Intrusion Response Engine.* The *Intrusion Response Engine* is responsible for sending a local and a global alarm in order to notify the nodes of the ad hoc network about the incident of an intrusion. The proposed Intrusion Response Engine is composed of three main modules: the *Communication Module*, the *Local Response Module* and the *Global Response Module*. The *Local Response Module* is activated every time an intrusion is detected by the Intrusion Detection Engine while the *Global Response Module* is activated only in serious cases of attacks, i.e the eSOM map of a node (Fig. 2) is covered in its biggest part (over two thirds (2/3)) with signs of an attack (light color). Special attention should be paid on the function of the *Intrusion Response Engine* in order to avoid possible flooding, or Denial of Service attacks caused by the notification messages for likely intrusions. Thus, each notifying message generated by the Intrusion Response Engine is sent only once, while the Global Response Module is activated only in severe cases of attacks.

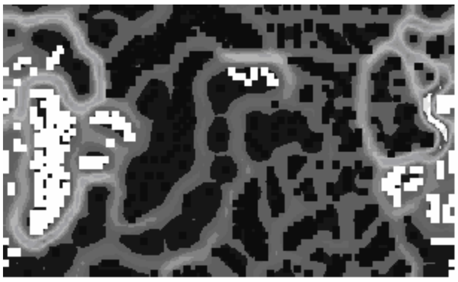


Fig. 2. Emergent SOM U-Matrix of a node of an ad hoc network

4. Proposed Intrusion Detection Engine

Kohonen's Self-Organizing Maps (KSOM) [22] have their base in biology. They belong in the category of unsupervised or competitive learning networks and produce a topological map, which illustrates the input data according to their similarity. The Self Organizing map is trained using only the characteristics of the trained data. The trained KSOMs create clusters of data, where similar vectors of features are located in a specific region in the output space. This is very useful for discovering clusters and relationships in data. The generated mapping is topology preserving. The algorithm of KSOM is described in the Appendix.

Although KSOMs present many advantages and have been extensively used in the research area of intrusion detection a special category of Self Organizing Maps called Emergent Self-Organizing Maps seem to provide much more advantages against Kohonen's SOM that can be exploited in order to achieve better results in the Intrusion Detection process.

Something that is often neglected in KSOM is that self-organization allows the emergence of structure in the data. According to [19], "Emergence is the ability of a system to produce a phenomenon on a new, higher level". In order to achieve emergence the existence and cooperation of a great number of elementary processes is necessary. Emergence may be presented not only in natural but also in technical systems. One of the basic disadvantages of SOM maps is that their abilities are limited to a few neurons. On the other hand, emergent Self-Organizing Maps may expand to many neurons. A large number of neurons in eSOM are necessary in order to achieve emergence. The cooperation of a big number of neurons leads to structures of a higher level, disregarding the elementary ones and allowing to consider structures that otherwise would be invisible. The clustering procedure in emergent SOMs is performed by observing the whole Emergent Self-Organizing Map and not by focusing on its neurons.

We have used the distance based (U-Matrix) method in order to visualize the structures generated by eSOM. According to this method [23] the sum (height) of distances between the neuron-weights is represented as the elevation of each neuron. If *n* is a neuron on the map, *NN(n)* is the set of immediate neighbors on the map and *w(n)* is the weight vector associated with neuron *n*, then the height *U-height* of each neuron *n* is given by (1) [24]:

$$U\text{-}height(n) = \sum_{m \in NN(n)} d(w(n) - w(m)) \qquad (1)$$

where *d(x,y)* is the distance used in the SOM algorithm to construct the map. The U-Matrix is a display of the U-heights on top of the grid positions of the neurons on the map. The input data set is displayed and depicted at a 3D landscape. The height will have a large value in areas of the map where one finds a few data points and small value in areas that represent clusters, creating hills and valleys correspondingly.

In our examples, we trained Emergent SOMs with logs of network traffic selected from a simulated mobile ad hoc network (using ns-2) and used eSOM U-Matrices [23] in order to perform intrusion detection. In our case, a vector represents each log of network traffic with some fixed attributes. Each vector has a unique spatial position in the U-Matrix and the distance between two points is the dissimilarity of two network traffic logs. The U-Matrix of the trained dataset is divided into valleys that represent clusters of normal or attack data and hills that represent borders between clusters. Depending on the position of the best match of an input data point that characterizes a connection this point may belong to a valley (cluster (normal or attack behavior)) or this data point may not be classified if its best match belongs to a hill (boundary). The map that will be created after the training of the Emergent SOM, will represent the network traffic. Thus an input data point may be classified depending on the position of its best match.

Considering the fact that image maps are exposed to the possibility of manipulation, we use a key agreement protocol in order to verify the authenticity of eSOM maps.

5. Proposed Intrusion Response Engine

The Intrusion Response engine and its components is illustrated in figure 3. The *Communication Module* is responsible for the agreement of the local keys (LK) and a global key (GK) that will be used in the local and global response modules, respectively.

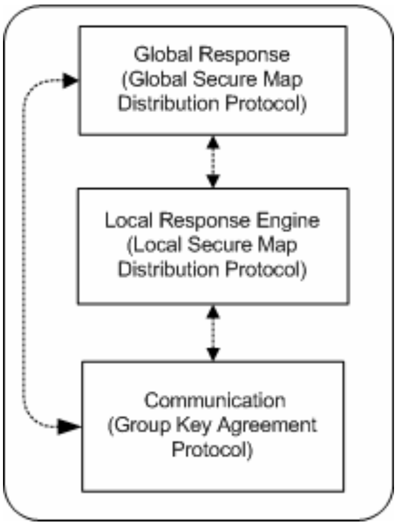


Fig. 3. Intrusion Response Engine

The *Local Response Module* is responsible for generating a global eSOM map, which is composed of the local eSOM maps of all one-hop away neighbors of a node. In order to verify the authenticity and integrity of the global eSOM map we apply a *Local Map Distribution Protocol*. The generated global eSOM map is mainly used for the visualization of the security status of the local ad hoc network composed of one-hop neighbors of a node. This map also helps nodes to select the most appropriate and secure neighboring node for message forwarding. The *Global Response Module* is responsible for the notification of all neighbors in the communication range of the attacked node.

5.1 Communication Module

The task of securing ad hoc networks is very complex and demanding, if we consider that their inherent vulnerable characteristics make them vulnerable to a broad range of attacks and the deployment of security mechanisms developed for networks based in a certain infrastructure is extremely difficult. *Key establishment* is a vital security mechanism that has been thoroughly investigated for securing structured networks. The purpose of a *group key* *establishment protocol* is the establishment of a group session key that would be used in the encryption/ decryption procedures in order to verify the authenticity of the transmitted messages over an open channel. The group key establishment protocols are divided into two main categories: the *group key distribution* and the *group key agreement* protocols [25].

In *group key distribution protocols* a member participant of the group selects the group session key and it is responsible for the secure distribution of this key in the other group members using a priori symmetric or asymmetric shared secrets. This scenario is not realistic and can not be applied in ad hoc networks since the existence of a Trusted Third Party (TTP) to distribute the session key is required. In *group key agreement protocols (GKA)* the group members agree on a common secret key, which is derived from each participant's public contribution. *Group key distribution protocols* are better suited to ad hoc networks with moderate sizes and with no central authority for the key distribution.

An *authenticated group key agreement protocol* provides the additional property of key authentication, which assures every group member that the agreed key will not be revealed to any other party. This authentication property is also called *implicit key authentication*. Our authenticated group key agreement protocol is based on the group key agreement protocols proposed by Lo et. al. [21] and Huang and Chang [20]. Our protocol adopts the exclusive-OR (XOR) operation proposed by [20] and [21] and not modular or exponential calculations that have extremely high computational cost. Furthermore, it employs some unique characteristics that make it even more secure. It uses a shared secret master key (K_M) and it is based on the structure of a rooted tree which depends on the distance between the ad hoc nodes. This results in a unique protocol flow.

The following security goals that are achieved through the proposed GKA protocol: k*ey secrecy*, *key independence*, *forward secrecy*, *backward secrecy* [21]. *Key Secrecy* is achieved since the key can be computed only by the group members. Moreover, since the disclosure of any set or group keys, does not lead to the disclosure of any other group key, *key independence* is also achieved. Furthermore, the disclosure of some long term secret does not lead to the disclosure of past group keys. Thus, the compromise of current session secrets does not imply compromise of future session's secrets. So a leaving group member is prevented from accessing the group communications and *forward secrecy* is achieved. In addition, the disclosure of current session secrets does not imply the disclosure of past session secrets, thus *Backward Secrecy* is also achieved. Through backward secrecy a new member is prevented from decrypting messages exchanged even if it has recorded earlier messages encrypted with the old key before it joined the group.

In the following paragraphs we give a short description of the way the GKA protocol functions. Table 1 provides the description for every notation used in the description of the GKA protocol and the local and global response modules.

Table 1. GKA Algorithm Notations Description

Notation	Description
M_i	Member *i*
M_d	The member in a descendant node
M_α	The member in an ascendant node
$M_{\alpha t}$	The member of the ad hoc network that is the victim of an attack
M_{C_j}	The member of the ad hoc network that is the jth child of its parent node in the tree structure
M_L	The member of the ad hoc network that leaves the ad hoc network (leave protocol)
ID_i	Member *i*'s identity
K_M	Master Key
$H()$	One – way hash function
S_i	Member *i*'s contributory key
z	Subkey generated by $M_1, \ldots, M_{n-1}$
K'_i, K_i	K'_i is the intermediate key and K_i is the session key hold by M_i
$nonce_i$	Member *i* generated random number
T_i	The tree path of member node M_i along its parent node to root node (ex. in Figure 2 the key path T_5 of node M_5 is $M_5 \rightarrow M_2 \rightarrow M_1$)
$\oplus$	XOR operation
$\|\|$	Concatenation
E_x	Encrypt data with key *x* by a symmetric algorithm
map_i	The eSOM map of member node *i*
LK	Local secret Key
GK	Global secret Key

Our authenticated key agreement protocol is not based on passwords as in [21] and [20], since such protocols are susceptible to dictionary and brute force attacks. In our scenario, there are *n* members sharing a secret master key K_M. K_M is embedded in the device of its member node and it is used for the initial communication. Although K_M can be used as a first step in order to setup a secure communication it is neither efficient nor sufficient to be used for a secure session communication. In addition, our proposed key agreement protocol is not based in a complete binary tree as in [21] but in a simple rooted tree.

We assume that there are n members $M_1, \ldots, M_n$, in an ad hoc network that want to have a secure communication. Each member of this group has a unique identity number *ID*. These members cooperate based on a rooted tree. In this rooted tree structure, every node is either a leaf or a parent of one or more children nodes. The nodes are denoted with each member's unique number. In this group, we assign member M_{Ch} to be a "checker". The checker is a group member that is randomly selected by the root node between its one-hop neighbors when the root structure is created. In case that the randomly selected checker M_{Ch} leaves the group of nodes due to mobility reasons, the root node is able to detect this movement and select another one-hop neighbor for the "checker". The checker does not participate in the tree structure, but it has the additional role of confirming the session key correctness. Furthermore, in order to select the "checker", it is not required the knowledge of the global static topology but just the one hop neighbors of the root node, something that is easily identified by the root node.

The rooted tree is constructed based on the distance between nodes and their unique numbers. If we assume that level 0 is the root of the tree and it is situated in node M_1 with ID number ID_1, then in level one will be situated its one-hop neighbors; in level two its two-hop away neighboring nodes until the last level where the most distant nodes of node M_1 that are in its transmission range will be situated.

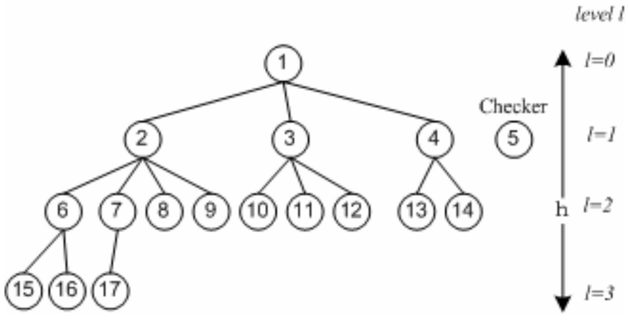


Fig. 4. Key structure illustrates membership

Figure 4 illustrates an example of a key tree with 17 members. In the key tree, its root is located at level *l=0* and its height is *h*, where *h=4*. All nodes in level one are one-hop away neighbors of root node M_1, all nodes in level two are two-hop away neighbors of node M_1 (root) and one-hop away neighbors of level one node parents (M_2, M_3, M_4 respectively). Moreover all nodes in level three are three-hop away neighbors (the last nodes in the communication range of node M_1). Moreover, member node M_{17} is the candidate node and member node M_5 is the checker. Our goal is all nodes in the ad hoc network to agree upon a session key K:

$$K = S_1 \oplus S_2 \oplus .... \oplus S_n , \qquad (2)$$

where S_i is contributed by M_i and randomly selected.

The protocol is divided into two phases: the *key initiation phase* and *the session key generation phase*. In the *key initiation phase*, M_1, M_2, ..., M_n cooperate to secretly construct a subkey z:

$$z = S_1 \oplus S_2 \oplus ... \oplus S_n \qquad (3)$$

We note here that in equation 3 the checker node M_{Ch} do not contribute in the construction of the subkey.

In the *session key generation phase*, each M_i (*i= 1, 2, ..., n* and *i≠Ch*) engages in a separate exchange with M_{Ch}. After this exchange all members have sufficient information to compute the session key *K*. Member node M_{Ch}, also verifies that the other members generated the same session key *K*. We introduce our method in detail in the following subsections.

5.1.1 Key initiation phase

Initially, all member nodes are considered to be malicious unless proven otherwise. A node that starts the normal communication process (Key initiation phase) is considered to be valid. The proposed key initiation phase follows a unique protocol flow based on the structure of a rooted tree depending on the distance of the ad hoc nodes. Each member M_i (for $i=1,2, \ldots, n$ and $i \neq Ch$) chooses a random quantity.

Similarly to [19], for $i \neq Ch$, all member nodes perform three steps to achieve mutual authentication.

In the *first step,* each descendant node (M_d) sends to his ancestor (M_a) a message that includes its ID (ID_d), the ID of its ancestor (ID_a) and an encrypted message with the master key (K_M) that is derived from the concatenation of the ID of the descendant (ID_d), the ID of the ancestor (ID_a) and some $nonce_d$ generated by M_d (i.e. $E_{K_M}(ID_d \| ID_a \| nonce_d)$).

In the *second step* the ancestor node (M_a) sends to the descendant node (M_d) a message that includes its ID (ID_a), the ID of the descendant (ID_d) and an encrypted message with the master key (K_M) that is derived from the concatenation of the ID of the descendant node (ID_d), the ID of the ancestor node (ID_a), the nonce generated by the descendant increased by one ($nonce_d+1$), and the $nonce_a$ generated by the ancestor (i.e. $E_{K_M}(ID_a \| ID_d \| nonce_d + 1 \| nonce_a)$).

In the *third step* the descendant node (M_d) sends to the ancestor node (M_a) a message that includes its ID (ID_d), the ID of the ancestor (ID_a) and an encrypted message with the master key (K_M) that derives from the concatenation of the ID of the descendant (ID_d), the ID of the ancestor (ID_a), the nonce generated by the ancestor increased by one ($nonce_a+1$), and the intermediate key K_i' (i.e. $E_{K_M}(ID_a \| ID_d \| nonce_a + 1 \| K_i')$).

The value that K_i' depends on the position of node i in the rooted tree.

- If M_i is a leaf node then $K_i' = S_i$, (4).
- If M_i is a parent node then $K_i' = S_i \oplus K'_{C_1} \oplus \ldots \oplus K'_{C_j}$, (5)

  where K'_{C_j} is the intermediate key of j child (C_j) of M_i node.
- If $i=1$, M_i is the root node and computes the values of z; where
  $z = S_1 \oplus K'_{C_1} \oplus K'_{C_2} \oplus \ldots \oplus K'_{C_j} = S_1 \oplus S_2 \oplus \ldots \oplus S_{n-1}$, (6).

The protocol flow of the *Key Initiation Phase* is the following:

For ($i \neq Ch$)**:** Steps 1 to 3 are used for mutual authentication

Step 1

$$M_d \xrightarrow{ID_d, ID_a, E_{K_M}(ID_d \| ID_a \| nonce_d)} M_a$$

Step 2

$$M_d \xleftarrow{ID_a, ID_d, E_{K_M}(ID_a \| ID_d \| nonce_d + 1 \| nonce_a)} M_a$$

Step 3

$$M_d \xrightarrow{ID_a, ID_d, E_{K_M}(ID_a \| ID_d \| nonce_a + 1 \| K_i')} M_a$$

- If M_i is a leaf node then $K_i' = S_i$. (7)
- If M_i is a parent node and has one or more children represented as C_j ($j= 1,\dots,l$), then:

$$K_i' = S_i \oplus K_{C_1}' \oplus \dots \oplus K_{C_l}'. \quad (8)$$

- If M_i is the root node ($i = 1$) then M_1 computes the subkey z

$$z = S_1 \oplus K_{C_1}' \oplus K'_{C_2} \oplus \dots \oplus K_{C_j}' = S_1 \oplus S_2 \oplus \dots \oplus S_{n-1}. \quad (9)$$

An example of the subkey *z* generation mechanism for nodes M_i for *i=1, 2, ..., 17* is illustrated in figure 5.

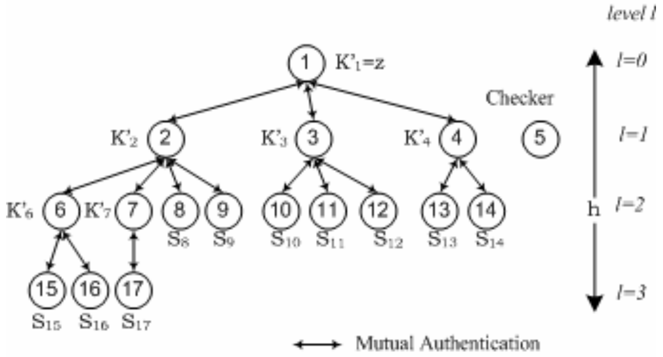


Fig. 5. Subkey z generation by M_1 to M_{17}

At the end of this phase the local keys LK of the one-hop neighbors of the root node are calculated and are used in the Local Response module:

$$LK = LK_j = z \oplus S_j \quad (10)$$

where j are all the one-hop neighbors of the root node and consequently belong in the first level of the tree.

5.1.2 Session key agreement phase

Our session key agreement phase uses a shared master key and not a password. The subkey $z(= S_1 \oplus S_2 \oplus \dots \oplus S_n)$ generated by M_1 at the end of the key initiation phase is used in the session key agreement phase.

Similarly to [21], the protocol flow of the *Key Agreement Phase* is the following:

Step 1

$$M_1 \xrightarrow{Broadcasts\ ID_1, E_{K_M}(ID_1 \| z \| nonce_1)} M_i, for\ i = 2,3,\dots,n$$

Step 2

M_{Ch} $\xrightarrow{\textit{Broadcasts}\ ID_{Ch}, E_{K_M}(ID_{Ch} \| S_{Ch} \| nonce_1 + 1 \| nonce_{Ch})}$ M_i*, for i= 1, 2, ..., n* and *i≠Ch*

Step 3

M_{Ch} $\xleftarrow{ID_i, ID_{Ch}, H(ID_{Ch} \| nonce_{Ch} + 1 \| K_i)}$ M_i*, for i= 1, 2, ..., n* and *i≠Ch*

$$\textit{and}\ K_i = z \oplus S_{Ch} \tag{11}$$

Step 4

M_{Ch} checks session key K.

In the *first step* of the session key agreement phase, the root of tree M_1 broadcasts subkey z to all members M_i, for $i = 1, 2, ..., n$ and $i \neq Ch$. In particular, M_1 broadcasts a message which includes its ID (ID_1) and an encrypted message with master key K_M, which includes the concatenation of the ID of node M_1 (ID_1), subkey z, and $nonce_1$ generated by node M_1 (i.e. $E_{K_M}(ID_1 \| z \| nonce_1)$).

In the *second step,* checker M_{Ch}, after receiving subkey z from M_1, generates a random quantity S_{Ch}, which will be used in the session key agreement phase and a random quantity $nonce_{Ch}$ for mutual authentication. Then, M_{Ch} broadcasts to all nodes M_i, for i = 1, 2, …, n and $i \neq Ch$, its ID (ID_{Ch}) and an encrypted message with master key K_M which includes the concatenation of its ID (ID_{Ch}), the generated random quantity S_{Ch}, the nonce generated by node M_1 increased by one $(nonce_1 + 1)$ and the $nonce_{Ch}$ generated by M_{Ch} (i.e. $E_{K_M}(ID_n \| S_n \| nonce_1 + 1 \| nonce_n)$).

In the *third step* the members M_i, for i = 1, 2, …, n and $i \neq Ch$, unbind the quantity received from M_{Ch} and construct a session key $K_i = z \oplus S_{Ch}$. Then, M_i, for i = 1, 2, …, n and $i \neq Ch$, send a message that includes their ID (ID_i), the ID of member node M_{Ch} (ID_{Ch}) and the hash produced by hash function H of a message derived by the concatenation of ID_{Ch}, the random quantity $nonce_{Ch}$ increased by one and K_i (i.e. $H(ID_n \| nonce_n + 1 \| K_i)$).

In the *fourth step,* member M_{Ch} verifies that each member has generated the same session key $K= (K_1= K_2= ... =K_n)$ and notifies all members M_i, for i = 1, 2, …, n and $i \neq Ch$, that the session key has been established successfully. The session key K agreed in this phase is the Global Key (GK) that will be used in the Global Response module:

$$GK = K = z \oplus S_{Ch} \tag{12}$$

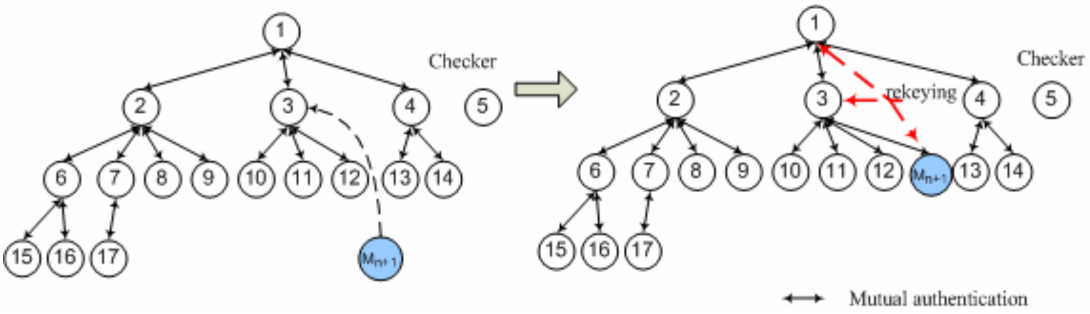
5.1.3 Membership Events

In a dynamic ad hoc network, member nodes may have short membership duration. New members may join or leave the group, after the session key is generated. The new members are not authorized to know the session key agreed before they join the ad hoc network. Thus, the member nodes of the ad hoc network should change the agreed session key and the shared secret master key also. In the same manner, in case a node leaves the ad hoc network the session key should be changed in order to verify the secure communication of the remaining nodes. This section describes two protocols, that achieve the above, namely the *Member Joining Protocol* and the *Member Leaving Protocol*.

A. Member Joining Protocol

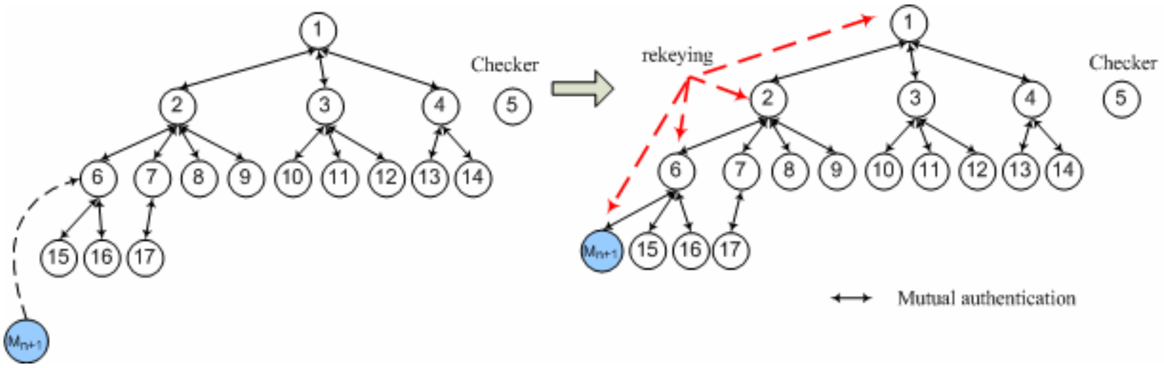
The proposed member joining protocol follows the protocol flow implied by the structure of the rooted tree. Let us assume that the group has *n* members: $M_1, M_2, \ldots, M_n$ and a new member M_{n+1} wants to join a group of nodes. The new member sends a joining request message which includes its ID (ID_{n+1}). The new member will be allowed to join the group when the members in the group receive this message and accept it. Furthermore, the members of the group must change the master key from K_M to K'_M and reconstruct the session key.

The procedure that will be followed depends on how close to the root member node M_1 will be the new member node M_{n+1} will be. Thus, if the new member node M_{n+1} is a one-hop neighbor of the root node M_1 then it will be situated in level one, if it is a two-hop neighbor of the root node M_1, and consequently one-hop neighbor node of a parent node in level one, it will be situated in level two. If we suppose that the new member node M_{n+1} is *l*-hops away from root node M_1, then its position in the rooted tree will be in the l $^{\text{th}}$ level and would be a child of the node of which it is a one-hop neighbor. Then the key path $M_{n+1} \rightarrow M_{l-1} \rightarrow M_{l-2} \rightarrow \ldots \rightarrow M_1$ where M_{l-1} (M_{l-2}) represents the member node that is situated in *l-1* (*l-2*) level and is parent node one (two)-hop neighbor of the new member node (respectively).



(a) **(b)**

Fig. 6. Key tree after a new node joins in the second level.



(a) **(b)**

Fig. 7. Key tree after a new node joins in the third level.

Two examples of a new member node M_{n+1} joining a group of n members $M_1, M_2, \ldots, M_n$ are illustrated in figures 6 and 7 where the new member will become an internal node or a leaf node respectively. The new member node M_{n+1} is a three-hop neighbor of a root node M_1, a two-hop neighbor of member node M_2 and a one-hop neighbor of member node M_5. Each of the members M_i, except the root M_1on the key-path T_{n+1}, $M_{n+1}\rightarrow M_5\rightarrow M_2\rightarrow M_1$ performs the key initialization phase. Particularly, if M_d is a descendant node and M_α is an ancestor node in the key path then the following protocol flow of the *Tree Path Key Initiation phase* is performed*:*

Step a

$$M_d \xrightarrow{ID_d, ID_a, E_{K'_M}(ID_d \| ID_a \| nonce''_d)} M_a$$

Step b

$$M_d \xleftarrow{ID_a, ID_d, E_{K'_M}(ID_a \| ID_d \| nonce''_d + 1 \| nonce''_a)} M_a$$

Step c

$$M_d \xrightarrow{ID_a, ID_d, E_{K'_M}(ID_a \| ID_d \| nonce''_a + 1 \| K_i'')} M_a$$

- If M_i is a leaf node then $K''_i = S''_i$. (13)
- If M_i is a parent node and has one or more children represented as C_j ($j= 1,\ldots,l$), then $K''_i = S''_i \oplus K''_{C_1} \oplus \ldots \oplus K''_{C_l}$. (14)
- If M_i's child M_{C_j} is not in the key path, then $K''_{C_j} = K'_{C_j}$, (15)

- If M_i is the root node ($i=1$), then M_1 computes the subkey z' using the K_i'' of the members in the key path and the K_i of the other members of the tree and computes the new sub key:

$$z' = S_1 \oplus K'_{C_1} \oplus K'_{C_2} \oplus ... \oplus K'_{C_j} \ . \qquad (16)$$

Finally, M_1 performs the algorithm of the key agreement phase (section 5.1.2) to reconstruct and verify a new session key.

B. Member Leaving Protocol

Let us assume that there are *n* members in the group of the ad hoc network and that member M_L wants to leave. In this case, the agreed session key should be changed and the key structure should be altered. In order to maintain the security of the group ad hoc nodes, the shared master key (K_M) should also change to K'_M which is derived from the XOR operation between the K_M and the old session key K_{old}

$$K'_M = K_M \oplus K_{old} \qquad (17).$$

The leaving member M_L initiates the *Member Leaving Protocol* by sending a leave message to all group members. The *Member Leaving Protocol* follows the protocol flow implied by the structure of the rooted tree and uses the new shared master key K'_M . The *Member Leaving Protocol* depends on the M_L's location in the key tree. Thus, we divide two cases depending on if M_L is a leaf or an internal node.

Case 1: M_L is a leaf node

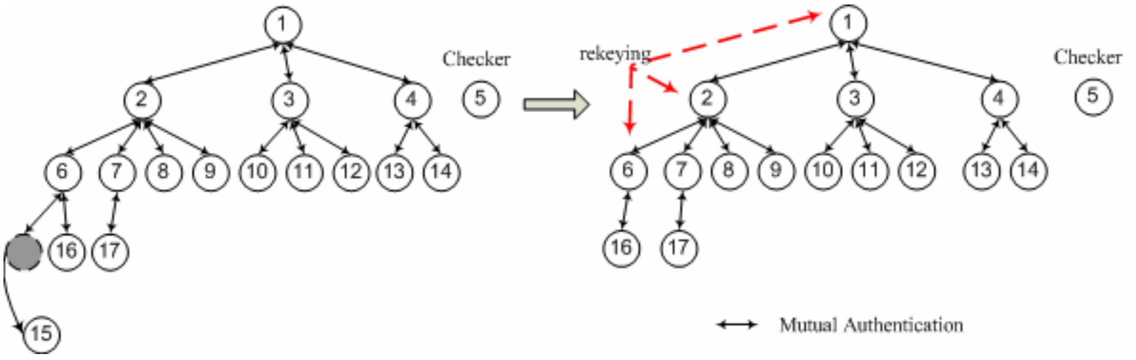
Figure 8 (a) shows an example of this case clearly. In this case, the following two steps have to be performed.

Step 1

If M_{15} leaves the group, the new key tree structure is shown in figure 8 (b).

Step 2

Each of the members M_i except the root M_1, on the key-path $M_6 \rightarrow M_2 \rightarrow M_1$ perform the key initialization phase described in section 5.1.1. Finally, M_1 performs the algorithm of the key agreement phase (section 5.1.2) to reconstruct and verify a new session key.



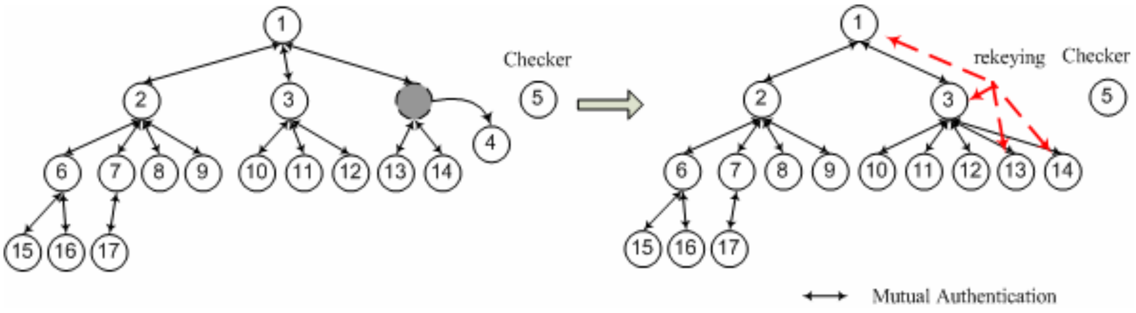
(a) **(b)**

Fig. 8. Key tree after the departure of member M_{14}

Case 2: M_L is an internal node.
Figure 9 (a) shows an example of this case. In this case the following two steps are performed.

Step 1
If M_4 leaves the group, then the one-hop neighbor of root node that is closer to the children of the leaving node will become their parent node. In our example in Figure 9 when the member node M_4 leaves, then the node that will replace it is either node M_2 or node M_3. If we suppose that node M_3 is closer to M_4's children then M_3 will become their new parent node and the key tree is reorganized. The new key tree structure is depicted in figure 9 (b).

Step 2
Each of the members M_i, except the root M_1 on the key-path $M_3 \rightarrow M_1$ and the two children $M_{13,}$ M_{14} of node M_3 perform the key initialization phase. More specifically, for nodes M_3,, M_{13},, M_{14} the protocol flow of the *Tree Path Key Initiation phase* is performed (described in section 5.1.3 A) in order to compute the new subkey z'. Finally, M_1 performs the algorithm of section 5.1.2 to reconstruct a new session key.



(a) **(b)**

Figure 9. Key tree after the departure of member M_2

5.1.4 Periodic Session Key Update

If no member joining or leaving events take place for a long period of time, the session key K, must be changed in order to prevent it from possible exposure and to verify its security strength. Same as [21], the steps that have to be performed are the following.

In the *first step,* member node M_{Ch} broadcasts a message to member nodes M_i, for $i=1, 2, 3, \ldots, n$ and $i \neq Ch$. The broadcasted message includes the ID of the member node M_{Ch} (ID_{Ch}) and an encrypted message with the old key (K_{old}) which includes the concatenation of the ID of the member node (ID_{Ch}), a new random quantity (S''_{Ch}) generated by M_{Ch}, and $nonce_{Ch}$ generated by M_{Ch} ($i.e.(E_{K_{old}}(ID_{Ch} \| S_{Ch}'' \| nonce_{Ch}))$).

In the *second step,* member nodes M_i, for $i=1, 2, 3, \ldots, n$ and $i \neq Ch$**,** decrypt the message sent by M_{Ch} and compute the new session key

$$K_{new} = K_{old} \oplus S''_{Ch} . \qquad (18)$$

In the *third step,* member nodes M_i, for $i=1, 2, 3, \ldots, n$ and $i \neq Ch$, send a message to member node M_{Ch} which includes the identity (ID_i) of each member node M_i, the identity (ID_{Ch}) of member node M_{Ch} and the hash of the concatenation which includes the identity of (ID_{Ch}) of member node M_{Ch}, the $nonce_{Ch}$ generated by M_{Ch} increased by one and the new session key K_{new}.

In the *fourth step,* member node M_{Ch} verifies that that each member has generated the same session key $K_{new}(=K_1=K_2=\ldots=K_n)$ and notifies all members M_i, for $i=1, 2, 3, \ldots, n$ and $i \neq Ch$, that the session key is established successfully.

We note here that the new session key K_{new} is the new Global Key (GK_{new}) that will be used in the Global Response Module:

$$GK_{new} = K_{new} = K_{old} \oplus S''_{Ch} \qquad (19)$$

The protocol flow of the *Periodic Global Session Key Update* [21] is the following:

Step 1

$M_{Ch} \xrightarrow{Broadcasts\ ID_{Ch}, E_{K_{old}}(ID_{Ch} \| S_{Ch}'' \| nonce_{Ch})} M_i$, for $i=1, 2, 3, \ldots, n$ and $i \neq Ch$

Step 2

M_i , for $i=1, 2, 3, \ldots, n$ and $i \neq Ch$ computes the new session key, $K_i = K_{old} \oplus S''_{Ch}$. (20)

Step 3

$M_{Ch} \xleftarrow{ID_i, ID_{Ch}, H(ID_{Ch} \| nonce_{Ch} + 1 \| K_i)} M_i$, for $i=1, 2, 3, \ldots, n$ and $i \neq Ch$

Step 4

M_{Ch} checks the new session key $K_{new} = K_i$.

Except of the periodic update of the Global Key, the Local Keys used by the Local Response module should also be updated after a long period of time.

The steps that are performed for the periodic update of the Local Keys are the following:

In the *first step*, member nodes M_j that are situated in the first level of the tree and are one-hop neighbors of M_1 send to root node M_1 a message that includes their IDs (ID_j) and an encrypted message with the old local keys ($LK_{old}=LK_j$) which includes the concatenation of their IDs (ID_j), a new random quantity (S''_j) generated by M_j, and $nonce_j$ generated by M_j, (i.e. ($E_{LK_{old}}(ID_j \| S_j'' \| nonce_j)$)).

In the *second step*, member nodes M_j compute the new Locals Keys $LK_{new} = LK'_j = LK_{old} \oplus S''_j$ (21).

In the *third step*, member nodes M_j send to the root node M_1 a message that includes their IDs (ID_j) and the hash of the concatenation which includes the identity (ID_j) of member nodes M_j, the $nonce_j$ generated by M_j increased by one and the new Locals Keys $LK_{new} = LK'_j$.

In the *fourth step*, root node M_1 checks the new local keys.
The protocol flow of the *Periodic Local Keys Update* is the following:

Step 1

$$M_j \xrightarrow{ID_j, E_{LK_{old}}(ID_j \| S_j'' \| nonce_j)} M_1$$

where j are all the nodes in level 1 of the tree

Step 2

M_j, computes the new Local Keys, $LK_{new} = LK'_j = LK_{old} \oplus S''_j$

Step 3

$$M_j \xrightarrow{ID_j, H(ID_j \| nonce_j + 1 \| LK'_j)} M_1$$

Step 4

M_1 checks the new Local Keys $LK_{new} = LK'_j$

5.2 Local Response Module

The *Local Response Module* is responsible for generating the eSOM map of the one-hop node connectivity network. More precisely, in a local ad hoc network each node creates its local eSOM map through the *Intrusion Detection Engine*. Each node transfers its local eSOM map, through the *Local Map Distribution Protocol*, to all its one-hop neighbors. The way the *Local Map Distribution Protocol* functions is illustrated in Figure 10.

If we suppose that we have a group of ad hoc nodes $M_1, M_2, \ldots, M_n$ and M_i, for $i=2, 3, \ldots, n$**,** that are one-hop away neighbors of node M_1**,** then in the *first step, M_1* broadcasts a message to all nodes M_i, for $i=2, 3, \ldots, n$**.** The broadcasted message includes the ID (ID_1) of member node M_1, its map (map$_1$) and the hash of a message generated by the concatenation of its ID (*ID₁*), its map (map₁) and a *nonce₁* generated by M₁ (i.e. $H_{LK}(ID_1 \| map_1 \| nonce_1)$). The keyed hash function uses the local key LK generated by the communication module:

$$LK = LK_i = z \oplus S_i \qquad (22)$$

generated by the communication module.

In the *second step* all nodes M_i, for $i=2, 3, ..., n$, send to member node M_1 a message which includes their IDs (ID_i), their maps (map_i) and the keyed hash of a message generated by the concatenation of their IDs (ID_i), the nonce generated by node M_1 increased by one ($nonce_1+1$) and their maps (map_i) (i.e. $H_{LK}(ID_i \| nonce_1 + 1 \| map_i)$).

In the *third step*, the member node M_1 is able to compute the global map using the maps (map_i) sent by the other members of the one-hop connectivity network M_i, for $i=2, 3, ..., n$.

In the *fourth step*, the member node M_1 broadcasts a message to all nodes M_i, for $i=2, 3, ..., n$. The message includes its ID (ID_1), the GlobalMap generated and a hash of a message which includes the concatenation of its ID (ID_1), the GlobalMap and the nonce generated by M_1 (i.e. $H_{LK}(ID_1 \| GlobalMap \| nonce_1)$).

The protocol flow of the *Local Map Distribution Protocol* is the following:

Step 1

$$M_1 \xrightarrow{\textit{Broadcasts } ID_1, map_1, H_{LK}(ID_1 \| map_1 \| nonce_1)} M_i, \textit{for } i=2, 3, ..., n$$

Step 2

$$M_1 \xleftarrow{ID_i, map_i, H_{LK}(ID_i \| nonce_1 + 1 \| map_i)} M_i, \textit{for } i=2, 3, ..., n$$

Step 3

M_1 creates the local map of one-hop neighbors

Step 4

$$M_1 \xrightarrow{\textit{Broadcasts } ID_1, GlobalMap, H_{LK}(ID_1 \| GlobalMap \| nonce_1)} M_i, \textit{for } i=2, 3, ..., n$$

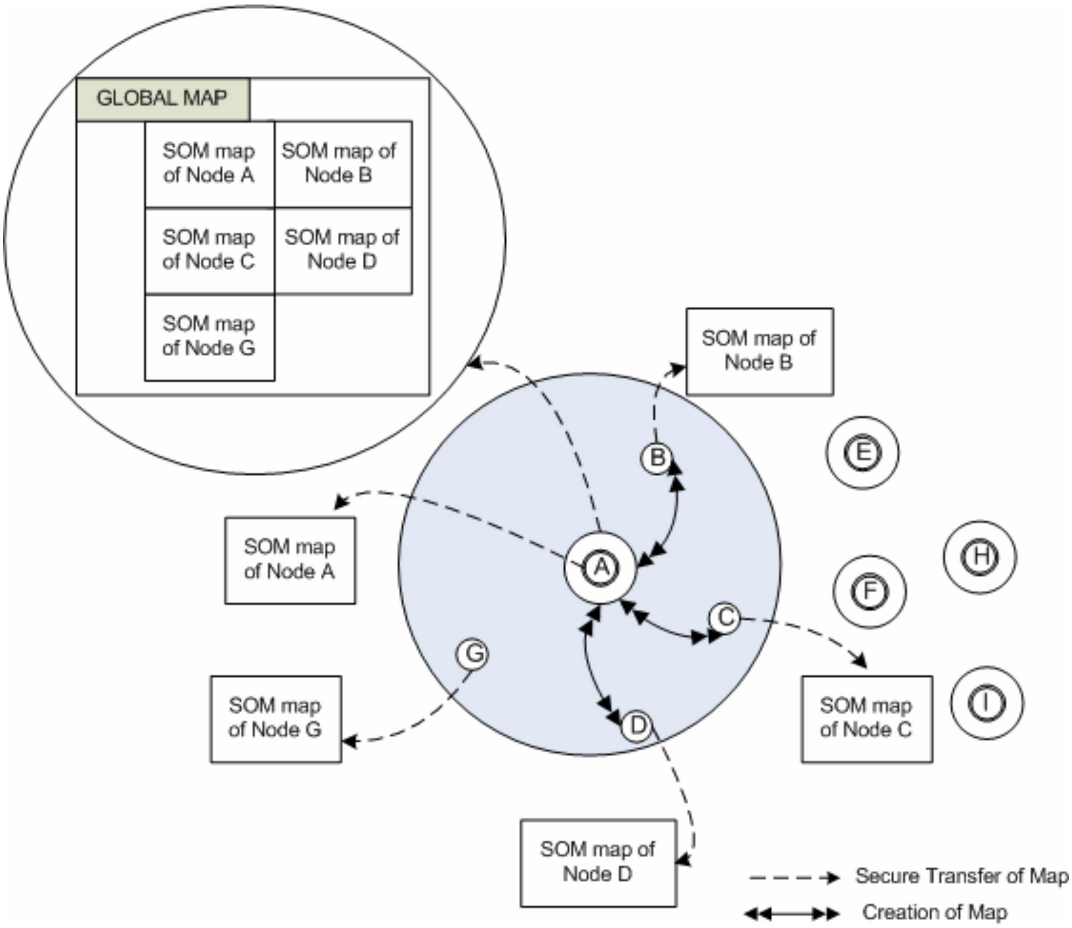


Fig 10. Functioning of Local Response Module

In Figure 10, nodes C, B, D and G are one-hop neighbors of node A. Nodes B, C, D and G create their own eSOM U-Matrix and use the local map distribution protocol in order to transfer securely their map to node A. Node A selects the local authenticated eSOM U-Matrices from its neighbors, creates its own map and then creates the global map of its local network. By observing the global map of its local network, node A is able to have a view of the security status of its neighboring nodes. Based on this information node A selects the appropriate node in order to forward its messages. By observing the local maps of all its neighboring nodes and by considering as secure the nodes that are not victims of attacks, node A selects an appropriate node for message forwarding. In case all nodes are victims of an attack, the node that is considered to be able to forward information is the one that forwards the messages.

5.3 Global Response Module

The *Global Response Module* is responsible for notifying all neighbors of the attacked node M_{at}, not only the ones that are one-hop away but all nodes in its transmission range (M_i**,** for $i=1, 2, ..., r$ and $i \neq at$), that there is a possible intrusion in this node. The notification of all nodes in the transmission range of the attacked node is performed using the *Global Response - Map Distribution Protocol*.

If we suppose that we have a group of ad hoc nodes M_i, for $i=1, 2, 3, ..., r$, that are all nodes in the transmission range of member node M_{at} in the *First step* of *Global Response - Map Distribution Protocol* the member node M_{at} which is a possible victim of the attack, broadcasts a message to all nodes M_i, for $i=1, 2, 3, ..., r$ and $i \neq at$, in the transmission range of node M_{at}**.** The broadcasted message includes the ID of the attacked node (ID_{at}) and a hash of a message produced by the concatenation of its ID (ID_{at}), its map (map_{at}) and nonce generated by node M_{at} (i.e. $H_{GK}(ID_{at} \| map_{at} \| nonce_{at})$). The keyed hash function uses the global key GK generated by the communication module: $GK = K = z \oplus S_{Ch}$ (23)

In the *second step*, all the nodes M_i, for $i=1, 2, 3, ..., r$ and $i \neq at$, remove M_{at} from their routing tables.

The protocol flow of the *Global Response - Map Distribution Protocol* is the following:

Step 1

$$M_{at} \xrightarrow{\text{Broadcasts, } ID_{at},\, H_{GK}(ID_{at} \| map_{at} \| nonce_{at})} M_i, \text{ for } i=1, 2, ..., r \text{ and } i \neq at$$

Step 2

M_i, for $i=1, 2, ..., r$ and $i \neq at$, removes M_{at} from its routing tables

Figure 11 depicts an example of how the *Global Response Module* functions. In this figure three local ad hoc networks are illustrated. One with one-hop neighbors for node C (including nodes A, B, D, E, F, G, H), one with one-hop neighbors for node I (including nodes J, K, L, M, N, O), and one with one-hop neighbors for node P (including nodes Q, R, S, T, U, V, W). We suppose that the *Intrusion Detection Engine* has detected node D as the victim of an attack. Then the Global Response Module is activated and the local eSOM map of node D is distributed using the global distribution protocol in order notify all neighbors in the communication range of node D about the attack so that all neighbors remove the attacked node from their routing tables and update the paths they use for their message transmissions.

We must note here that the *Global Response Module* cannot be compromised. We may discriminate two cases that a malicious node may attempt to compromise the Global Response Module. In the first case, a malicious node falsely activates the *Global Response Module* by spreading the rumor that it is a victim of a severe attack. In this case the malicious node will acquire no benefit for itself since it will isolate itself and it will trigger its removal from the routing tables of its neighbors. In the second case, a malicious node might attempts to spread fake rumors that another legitimate node is a victim of an attack. But this event is prevented since the integrity of its eSOM map is assured through an integrity mechanism such as a MAC or hash function and the communication between member nodes is mutually authenticated.

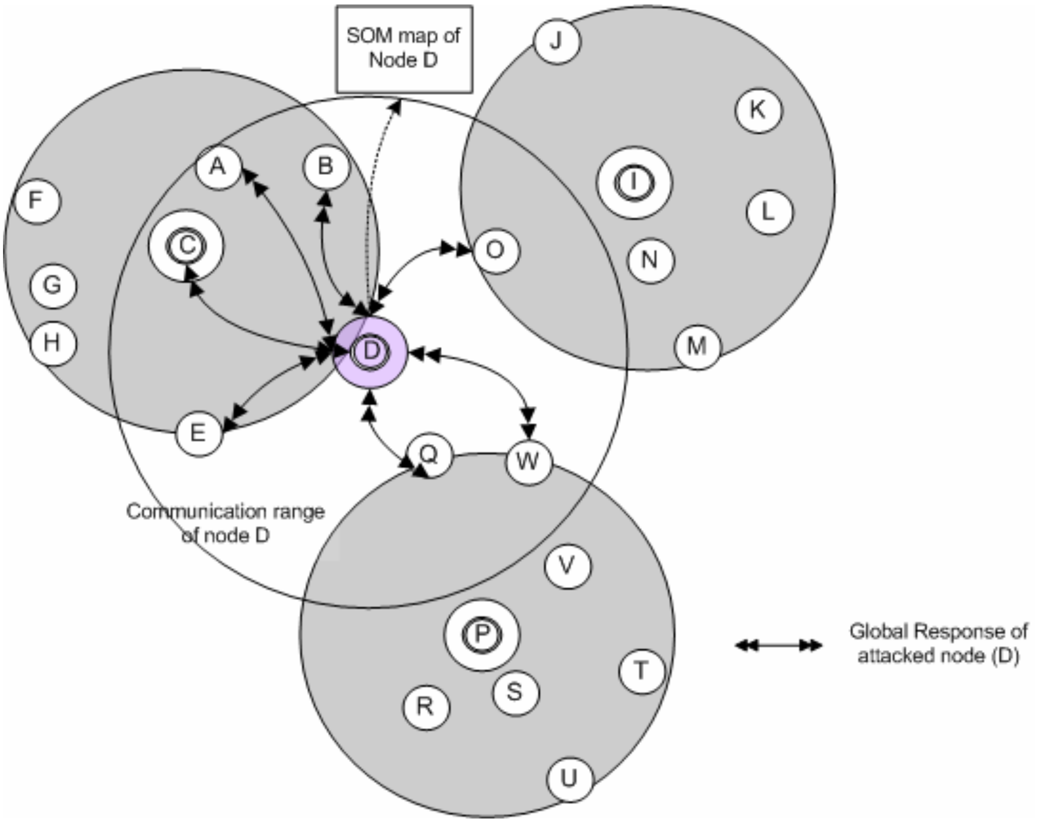

SOM map of Node D
J
K
A
B
I
F
L
C
O
N
G
H
D
M
E
Q
W
Communication range of node D
V
P
Global Response of attacked node (D)
T
S
R
U

Fig. 11. Functioning of Global Response Engine

6. Performance Evaluation

In order to evaluate our approach we have simulated a type of ad hoc networks, a mobile ad hoc network. Malicious nodes in a mobile ad hoc network may target to exploit features of the physical, MAC or network layers. The majority of the so far proposed security approaches in such networks has focused in the network layer, while little research has been done on the MAC layer security. The role of the MAC layer in wireless ad hoc networks is substantial as it is responsible for maintaining the communication between nodes and the scheduling of the access in a shared radio channel.

The MAC layer is directly affected by almost every intrusion [7], since it is placed in the first layers of the protocol stack. Indeed, the data delivery ratio, or the throughput, may be affected by the malicious behavior or by the misuse of the shared medium (e.g., selfishness) due to increased routing load. The control overhead for each delivered data packet may also increase. Thus, intrusion detection mechanisms that are based on features selected in the MAC layer are faster regarding the detection delays and the response time. Furthermore, these features make the discrimination between normal and abnormal behavior easier.

To evaluate the feasibility of our intrusion detection engine we have conducted a series of experiments. For our experiments we have made the assumption that the network has no preexisting infrastructure and that the employed ad hoc routing protocol is AODV. We implemented the simulator within the ns-2 library. Our simulation modeled a network of 50 hosts placed randomly within an 1800 × 1000m^2 area. Each node has a radio propagation range of 250 meters and the channel capacity was 2 Mb/s. The nodes in the simulation move according to the 'random way point' model. At the start of the simulation, each node waits for a pause time, then randomly selects and moves towards a destination with a speed uniformly lying between zero and the maximum speed. On reaching this destination it pauses again and repeats the above procedure till the end of the simulation. The minimum and maximum speed is set to 0 and 10 m/s, respectively, and pause times at 0, 20, 50, 70 and 200 sec. A pause time of 0 sec corresponds to the continuous motion of the node and a pause time of 200 sec corresponds to the time that the node is stationary.

The training dataset we have used is derived from a simulation in ns-2 where among the 50 nodes that participate in the network the 15 are malicious, the nodes move according to the 'random way point model' and the pause time is equal to 50 sec. We evaluated the performance of our proposed intrusion detection module for 5, 10, 15 and 20 malicious nodes. In each case the number of all nodes in the network is set to 50. The malicious behavior is carried between 50 and 200 sec. The nodes perform normally between 0 and 50 sec. These parameters result in a network with rather high mobility and high traffic activity.

On average, twenty traffic generators were developed to simulate Transmission Control Protocol (TCP) data rate to ten destination nodes. This traffic pattern results in twenty connections among source and destination nodes. The sending packets have random sizes and exponential inter-arrival times. The sources and the destinations are randomly selected with uniform probabilities. The mean size of the data payload was 512 bytes. Each run is executed for 200 sec of simulation time with a feature-sampling interval of one sec. We used the IEEE 802.11 Distributed Coordination Function (DCF) as the medium access control protocol. The mobility of the nodes is random determined by scenario files that are generated by the scene generator of ns-2. A free space propagation model with a threshold cutoff was used in our experiments. In the radio model, we assumed the ability of a radio to lock onto a sufficiently strong signal in the presence of interfering signals, i.e., radio capture.

In our experiments, we simulated a constant selective packet-dropping attack where the attacker simply discards all data packets while it functions legitimately concerning routing and MAC layer packets. This type of attack is extremely difficult to detect if we consider that packet dropping is due to a malicious behavior or mobility. To add to the problem, we let the malicious node exhibit malicious behavior when it is most advantageous to him and not from the beginning of the traffic.

The statistical features we have used have been introduced by Liu et al. [7] in their proposed approach for performing intrusion detection in the MAC layer. These features are as follows:

- *Network allocation vector (NAV):* it is a node specific characteristic, which depicts the time that the node will occupy the medium for sending its messages.
- *Transmission traffic rate:* indicates the rate of the transmitted packets.
- *Reception traffic rate:* indicates the rate of the received packets.
- *Retransmission rates of RTS packets:* indicates the rate of the ReadyToSend packets that are retransmitted by the monitoring node. A high value of this feature suggests a possible packet dropping attack.

- *Retransmission rates of DATA packets:* indicates the rate of the data packets that are retransmitted by the monitoring node. A high value of this feature suggests a possible packet dropping attack.
- *Active neighbor node count:* represents the number of neighbor nodes that have data transmission activities.
- *Forwarding node count:* represents the number of neighbor nodes that communicate directly with the monitoring node.

In order to avoid having a great influence of the attributes of some input vectors it is necessary to normalize the input data. We have normalized the data with mean zero and variance one, a technique that produces very good results in most cases as reported in the literature. For the evaluation we have used the Databionics eSOM tool ([26], [27]).

The presented evaluation proves that we can achieve a differentiation between normal and abnormal behaviors concerning packet-dropping attacks. In order to perform clustering with eSOM U-Matrices we followed the proceeding procedure. The best matches of the trained dataset and thus the corresponding dataset are manually grouped into clusters representing normal and attack behavior. Thus, we identify the regions of the map that represent a cluster that can be used for the classification on new datasets. The eSOM of a trained dataset is depicted in Figure 2. As it can be clearly seen the training data set has been divided in two classes that are very well distinguished, normal data class (dark color) and packet dropping data class (light color). In order to make sure that our intrusion detection engine will always provide efficient and accurate results we should update our trained eSOM U-Matrix according to the new conditions concerning mobility.

In order to evaluate the efficiency of the proposed intrusion detection engine we use two measures the *Detection rate* and the *False alarm rate*:

$$Detection\ rate = \frac{TP}{TP+FN} \tag{24}$$

$$False\ alarm\ rate = \frac{FP}{TN+FP}, \tag{25}$$

where *TP* is the number of true positives (attack logs classified as attacks), *TN* the number of true negatives (normal logs classified as normal), *FP* the number of false positives (normal logs classified as attacks) and *FN* the number of false negatives (attack logs classified as normal). The most effective approach should reduce as much as possible the *False alarm rate* and at the same time increase the *Detection rate.*

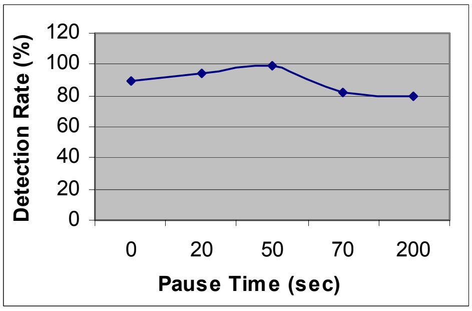


Fig. 12. Detection Rate vs. Pause Time

Figure 12 presents the average Detection rate of all source nodes that present traffic activity and are recognized as normal or attack by eSOM regarding the used pause times. The detection rate seems not to be influenced by the mobility and in all cases to be over 80%. For long pause times the rate slightly reduces which is due to the TCP traffic and the degradation of the mobility of the network. Indeed, a TCP agent stops sending data packets when it doesn't receive acknowledgment. Even after AODV discovers a new path to that destination, the agent keeps sending data packets through the malicious node, as the latter responds normally to control packets. As the network exhibits a rather low mobility, traffic always is rejected by the malicious node and it is soon stopped by the TCP agent, which degrades the audit data fed to eSOM.

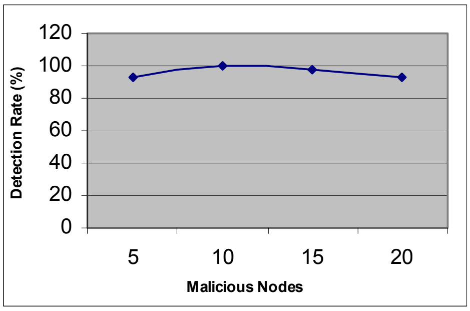


Fig. 13. Detection Rate vs. Number of Malicious Nodes

The detection rate as a function of the number of malicious nodes is presented in Figure 13. The rate is rather high and, as in the previous figures, always over 80%. When few malicious nodes exist in the network, the connections that are influenced by them are also a few, since source nodes move randomly in the network. This results in duplicated lines in the audit data set which is fed to eSOM, and consequently the detection rate is decreased. When the number of malicious nodes is high compared to the number of source nodes, the TCP connections generated automatically by NS are a few, which leads to multiple duplicate lines in the audit data that is fed to eSOM, which explains the decrease in the detection rate.

Table 2. False Alarms vs. Pause Time

Pause time (sec)	**False Alarm (%)**
0	21
20	20
50	22
70	20

Table 3. False Alarms vs. Number of Mobile Nodes

Malicious nodes	False Alarm (%)
5	26
10	22
15	17
20	21

Table 2 and Table 3 present the average false alarm rate as a function of the paused times used and the number of malicious nodes, respectively. When a source node generates traffic to different destinations and one of these connections is influenced by malicious nodes, then eSOM finds it difficult to distinguish among normal and abnormal traffic. If this is combined with multiple duplicate lines in the audit data due to mobility, the malicious node number produces rather high false alarm rates. The high false alarms are mainly caused because of the difficulty that the classifier (eSOM) faces to discriminate the change in the behavior of a node (depicted in the selected features) caused either due to mobility reasons or due to malicious behavior.

Two representative works in the area of anomaly detection is Deng et. al. [8] and Liu et. al. [7]. Deng et. al. [5] in their anomaly detection approach in MANET propose the use of SVM (Support Vector Machines) in a completely distributed architecture. The false alarm rate ranges from 3.5±5.8% to 20.85±8.03% for the Black hole attack and the Frequent False Routing Requesting (FFRR) attack.

Moreover, Liu et al. [7] in their approach for packet dropping attack using cross feature analysis although the false alarm rate is low 0.29% the detection rate is also rather low 72%. As the packet-dropping attack is a rather difficult attack to combat the low false alarm is combined with a low detection rate.

Our intrusion detection engine presents a rather high detection rate for the specific attack under study. Its main advantage is the visual representation of the normal-attack state in a mobile ad hoc network. Moreover, our intrusion detection engine has the ability to immediately respond in the case of a possible intrusion by selecting the more secure node as indicated by its u-Matrix map for forwarding the information. In order to verify the reliability and avoid possible alteration of the maps the proposed key agreement protocol must be used.

7. Conclusions and future work

In this paper, we have presented an intrusion detection engine that is part of a local IDS agent that exists in every node of an ad hoc network. The collaboration of all the local IDS agents composes an IDS for ad hoc networks. The proposed intrusion detection engine is based on emergent SOMs, a special and efficient class of neural networks that generates as an output a map and provided visual representation of the classification performed. We have examined how eSOM performs in classifying normal and abnormal behavior in ad hoc networks and we exploited the advantage of visualizing network traffic. Using eSOM, each node of the ad hoc network creates its local eSOM map as well as the global map of its local ad hoc network. The local and global eSOM maps provide us the important advantage of being able to have a visual representation of the security status of each ad hoc node as well as its local ad hoc network. Thus, each node has the option to select a secure routing path for packet forwarding by avoiding compromised neighbors.

In addition, we have proposed an Intrusion Response Engine composed of three modules: the Communication module, the Local Response module and the Global Response module. The intrusion response should be as quick as possible and reliable in the transmission of notifying messages. Thus, we propose an authenticated key agreement protocol. The Local Response module creates a global map through an authenticated protocol, for the one-hop neighborhood of each ad hoc node. The global map depicts the security status of each node's one-hop neighborhood. The Global Response module notifies every node in the transmission range of the attacked node about the incident of the attack and initiates the remove of the attacked node from the routing tables.

We exploit the significant advantages of visual representation and key agreement protocols in ad hoc networks. Special attention should be paid to the fact that the detection engine could be employed to various routing protocols. Regarding possible extension of this work, we plan to select features from other layers (e.g. network layer) in order to examine the performance of the proposed approach for the detection of other type of attacks. Furthermore, in our performance experiments TCP traffic is used as a more realistic one. Another data traffic type (e.g. Constant Bit Rate (CBR)) is under future investigation.

ACKNOWLEDGEMENTS

This work has been partially supported by the Greek Research and Technology Secretariat under a PENED grant.

References

[1] S. Makki, N. Pissinou, H. Huang, "The Security Issues in the Ad-Hoc on Demand Distance Vector Routing Protocol (AODV)", In Proceedings of the 2004 International Conference on Security and Management (SAM'04), pp. 427-432.

[2] N. Komninos, D. Vergados, and C. Douligeris, "Detecting Unauthorized and Compromised Nodes in Mobile Ad-Hoc Networks", Journal in Ad Hoc Networks, Elsevier, Volume 5, Issue 3, April 2007, pp. 289-298.

[3] Y. Zhang, W. Lee, Y. Huang, “Intrusion Detection Techniques for Mobile Wireless Networks”, Wireless Networks Vol. 9, No. 5, (September 2003), pp. 545-556.

[4] Y. Huang, and W. Lee, “A Cooperative Intrusion Detection System for Ad Hoc Networks”, in Proceedings of the 1st ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN’03), October (2003), Fairfax, VA, USA, pp. 135-147.

[5] H. Deng, Q. Zeng, and D. P. Agrawal, “SVM-based Intrusion Detection System for Wireless Ad Hoc Networks”, In Proceedings of the IEEE Vehicular Technology Conference (VTC’03), October (2003), Orlando, Florida, USA, pp. 2147-2151.

[6] O. Kachirski, and R. Guha, “Intrusion Detection Using Mobile agents in wireless Ad hoc Networks”, in Proceedings of the IEEE Workshop on Knowledge Media Networking, July (2002), Kyoto Japan, pp.153-158.

[7] Y. Liu, Y. Li and H. Man, “MAC Layer Anomaly Detection in Ad Hoc Networks”, In Proceedings of 6th IEEE Information Assurance Workshop, June (2005), West Point, New York, USA.

[8] Y. Huang, W. Fan, W. Lee and P. Yu, “Cross-Feature analysis for Detecting Ad-Hoc Routing Anomalies”, In Proceedings of the 23rd International Conference on Distributed Computing Systems, May (2003), Rhode Island, USA, pp. 478.

[9] C.Y. Tseng, P. Balasubramanyan, R. Limprasittiporn, J. Rowe, and K. Levitt, “A specification-based Intrusion Detection system for AODV”, In Proceedings of the 1st ACM Workshop on Security of Ad hoc and Sensor Networks (SASN’03), October (2003), Fairfax, Virginia, USA, pp. 125-134.

[10] F. Anjum, D. Subhadrabandhu, S. Sarkar, “Signature-based Intrusion Detection for Wireless Ad-Hose Networks”, In Proceedings of Vehicular Technology Conference (VTC’03), Wireless Security Symposium, October (2003), Orlando, Florida.

[11] T. M. Chen and V. Venkataramanan, “Dempster-Shafer Theory for Intrusion Detection in Ad Hoc Networks”, IEEE Internet Computing, Vol. 9, No. 6, November (2005), pp. 35-41.

[12]M. V. D. Burmester and Y. Desmedt. "A Secure and Efficient Conference Key Distribution System", In A. D. Santis, editor, Advances in Cryptology -- EUROCRYPT '94, volume 950 of Lecture Notes in Computer Science,. Springer-Verlag, 1995, pp. 275—286.

[13] K. Becker and U.Wille. “Communication complexity of group key distribution”. In Proceedings of 5th ACM Conference on Computer and Communications Security, ACM Press, 1998, pp. 1–6

[14] Y. Kim, A. Perrig, and G. Tsudik. “Tree-based group key agreement”. ACM Transactions on Information and System Security, Vol.7, No.1, February (2004), pp.60–96.

[15] Michael Steiner, Gene Tsudik and Michael Waidner. “CLIQUES: a new approach to group key agreement”. In proceeding of the 18th international conference on distributed computing systems (ICDS’98), May (1998), pp.380-387.

[16] R. Barua, R. Dutta, P. Sarkar, “Extending Joux Protocol to Multi Party Key Agreement”, In Proceedings of Indocrypt 2003, LNCS 2904, pp. 205-217, Springer-Verlag, 2003.

[17] R. Dutta, R. Barua and P. Sarkar, “Provably Secure Authenticated Tree Based Group Key Agreement”, In Proceedings of ICICS 2004, LNCS 3269, pp. 92-104, Springer-Verlag, 2004.

[18] R. Dutta and R. Barua, “Dynamic Group Key Agreement in Tree-based Setting”, In Proceedings of ACISP 2005, LNCS 3574, pp. 101-112, Springer-Verlag, 2005.

[19] D. Nalla and K. C. Reddy, “Identity Based Authenticated Group Key Agreement Protocol”, In Proceedings of Indocrypt 2002, LNCS 2551, pp. 215-233, Springer-Verlag, 2002.

[20] R. J. Hwang and R. C. Chang, “Key Agreement in Ad Hoc Networks”, Lecture Notes in Computer Science Volume 2745/2003 Parallel and Distributed Processing and Applications, pp.382-390.

[21] C. Lo, C. Huang and Y. Huang, “A Key Agreement Protocol Using Mutual Authentication for Ad-Hoc Networks”, Proceedings of International Conference on Services Systems and Services Management 2005 (ICSSSM’05), Vol. 2, June (2005), pp. 814-818.

[22] S. Haykin, “Neural Networks: A Comprehensive Foundation”, Prentice-Hall, New Jersey, USA, 2nd edition (1999).

[23] A. Ultsch, “Data Mining and Knowledge Discovery with Emergent SOFMs for Multivariate Time Series”, In Kohonen Maps, Elsevier Science,(1999) pp. 33-46.

[24] A. Ultsch, “Maps for visualization of high-dimensional Data Spaces”, In Proceedings of Worshop on Self-Organizing Maps (WSOM ’03), September (2003), Kyushu, Japan, pp. 225-230.

[25] A.J Menezes, S. A. Vanstone, P.C. Van Oorschot, “Handbook of Applied Cryptography”, 1st Edition, 1996, CRC Press, Inc., Florida, USA.

[26] A. Ultsch, F. Moerchen “ESOM-Maps: tools for clustering, visualization, and classification with Emergent SOM”, Tech. Report Dept. of Mathematics and Computer Science, University of Marburg, Germany, No. 46 (2005).

[27] Databionic ESOM Tools, Available from <http://databionic-esom.sourceforge.net/devel.html>

Appendix

The learning procedure for the Kohonen Self Organizing Maps is composed of the following steps:

a. Initialize the random weights w_{ij} (also known as codebook vectors of the neurons) with small random values.

b. Use an input pattern x.

c. Calculate the Euclidean distance (eq. 1 [16]) between input data sample x, and each neuron weight w_{ij}. The winner (Best Matching Unit) is chosen as $o(x)$:

$$o(x) = \arg\min_{j} \left\| x - w_{ij} \right\|, \quad j=1,2,\ldots,l, \tag{26}$$

where is l the total number of neurons.

d. Adjust all the weights in the neighborhood, in order to achieve the topological mapping, depending on their distance from the winning neuron according to the following equation [8]:

$$\forall j : w_{ij}(t-1) + a(t)\eta(t') \cdot (x_i(t) - w_{ij}(t-1)) \,, \tag{27}$$

where α is the learning rate, η the neighborhood function and t' the time that was spent in the current context. The neighborhood function η decreases as t' increases.

e. Repeat steps b, c, d until convergence