

A Hybrid Approach for Misbehavior Detection in Wireless Ad-Hoc Networks

S. Balachandran, D. Dasgupta and L. Wang
Intelligent Security Systems Research Lab
Department of Computer Science
The University of Memphis

Abstract—In wireless ad-hoc networks, the Dynamic Source Routing (DSR) protocol is sometime used in routing traffic among participating wireless nodes. Such networks are highly vulnerable to routing misbehavior due to malware, fault or compromised nodes. In a mobile ad-hoc environment, however, does not have (gateways or access points) traffic concentration points, thus requiring behavior monitoring of individual nodes. Intelligent machine learning techniques are therefore helpful for misbehavior detection that aims at keeping such vulnerable behavior under check. In this work, we used an anomaly detection algorithm, inspired by the biological immune system to detect such misbehaving nodes. This is a behavior-based anomaly detection technique, which assumes misbehavior identification by observing the deviations from the normal or expected behavior, of the nodes' protocol event sequence in wireless routing traffic. The objective is to build an effective technique, which like its natural counterpart, intelligently learns and detects unknown (and new) anomalous behavior with very low false positives. The classification of normal versus abnormal employs a hybrid negative selection for learning and adaptation. To evaluate the performance of the proposed technique, a simulated wireless network (GloMoSim) is employed for experiments. The proposed technique is designed to produce multi-set detectors using monitored behavior of neighborhood nodes. It is possible to distribute specialized detector sets to each network node thus fortifying the overall strength of the active wireless nodes.

1. INTRODUCTION

Ad-hoc wireless networks are peer-to-peer networks where each computer with a wireless interface can communicate directly with participating nodes. These nodes can self-organize without central management and special infrastructure. The network is established using (limited range) radio communication where each node acts as both data terminal and data transfer equipment. Moreover, nodes can move freely resulting in changes to the network topology and updated routing in order to forward the packets. The topology change depends on different factors such as mobility model, node speed etc. When old routes are broken and new ones are established there is always a risk of packet losses.

Security in ad-hoc networks [16] is of serious concern because of several reasons:

- Vulnerabilities associated with radio communications involve eavesdropping and spoofing of MAC address.

- Lack of infrastructure makes it impossible to assign well-defined roles such as trusted third parties and key management servers.
- Lack of dedicated routers and servers may hamper the performance of basic network operations (such as packet forwarding and routing) because of malicious network behavior or lack of cooperation.

There are some common wireless attacks such as man-in-the-middle, spoofing and Denial of service (DoS), which can be detected by examining MAC layer sequence numbers (refer [1] for details). The Sequence Number defines the frame and the Fragment Number identifies the fragment in each frame (section 3.3 briefly describes this packet format).

In this paper, we focus on routing misbehavior of wireless nodes (i.e. do not execute the DSR protocol properly) due to faulty software or malicious activities. Misbehavior detection aims at removing these vulnerabilities by examining the traffic pattern in the entire network. Traditionally, knowledge of known misbehavior patterns have been used [15] as signatures, and detected by monitoring at specific event sequences. Statistical techniques [4] can be utilized for such known misbehavior; however, learning and adapting to new misbehaviors are not usually possible. In all cases, it is necessary to correctly identify faulty nodes that, from time to time, suffer from node misbehavior or does not forward packets or are disastrously infected with Internet worms that maliciously attempt to bring the network down.

Behavior-based intrusion detection techniques assume that an intrusion can be detected by observing deviations from the normal or expected behavior of the system or the users. The intrusion detection system compares this behavior model with the current activities. When a deviation is observed, an alarm is generated. In other words, anything that does not correspond to a previously learned behavior is considered anomaly or intrusive.

Our behavior-based approach [1] uses negative selection algorithm to generate novel pattern detectors. These detectors are capable of distinguishing well-behaving

nodes from the misbehaving nodes with a good degree of accuracy. The False positives (or False alarms) could be minimized to good extent though some False Negatives exist because of subtle differences between good and bad behaviors. Countermeasures against misbehavior involve gradual isolation of misbehaving nodes, analyze and repair them.

The remainder of this paper is organized as follows: The next section revisits some related works in wireless networks and illustrates the problem scenarios and our prior works. Section 3 presents the details on immunity-based misbehavior detection, and section 4 provides simulation environment, data collection, preprocessing and simulation parameters with the experimental setup and performance metrics. Section 5 reports the experimental results with detailed analysis of the results.

2. BACKGROUND

2.1 Dynamic Source Routing (DSR) protocol

The DSR protocol [11] is a proposal for the future routing standard in mobile ad-hoc networks. It employs IP source routing wherein a list of nodes is included in the data packet header at the source, which are all the intermediate nodes to reach the destination. During route discovery by the source node, a **Route REQuest (RREQ)** packet is broadcasted locally with the address of the destination node. Those nodes having no information about the destination node or are first time recipients of the **RREQ** packets append their IP addresses to **RREQ** packets and re-broadcast it. The **RREQ** packets stop their journey when either the destination is reached or an intermediate node contains a route to the destination. On successful delivery, the destination node or the intermediate node may create a **Route REPLY (RREP)** packet containing the entire list of intermediate relaying nodes traversed by **RREQ** packet.

The DSR specification describes the source discovery, maintenance and usage of source routes. The nodes maintain a path cache scheme [10] wherein all the nodes forwarding (or promiscuously listening) data or control packets add sufficient information to their cache for limiting the route discovery for subsequent routing. For bidirectional links, the route replies are sent over the reverse collected routes. On the other hand, for unidirectional links, the route replies are sent to the source node initiating the corresponding **RREQ** packet at the beginning. As per the design implementation, the source node can accept one or more **RREP** packets from the same destination, while the destination node might respond to one or more incoming **RREQ** packets. The **Route REPLY** can be based on minimal hop count or latency. In the latency scheme, the replies that arrive earlier than others are indicative of better routes, because the waiting time for a node to send a route reply is directly proportional to the number of hops in the route leading to the end of a path. If during this wait, the node overhears a neighboring node's answer, it supposes that the

route it possesses is worse than one at its neighbor's and does not answer. This helps to avoid **RREP** storms and other unnecessary overheads.

Once the route is discovered, data packets are sent over the established route during which time the route is maintained. If for any reason a link breaks, the detecting node should send a route error message to the source (**RERR** packet). Depending upon the implementation, these nodes should salvage unsend packets and re-route them over alternate partial routes to the destination.

2.2 Related Works on WLAN security

There exist some intrusion detection systems (IDS) that were designed to secure wireless networks (WLAN). One such IDS [23] suggests to detect intrusions such as abnormal routing table updates and attacks at the MAC layer. It also provides a multi-layer integrated intrusion detection and response. Another WLAN IDS [14] encompasses components such as data collection, intrusion detection and an additional secure database to log anomalies. Recent research on mobile ad-hoc networks (MANETs) focus on routing protocols because they are the most critical part in packet delivery and network control. In [16], authors specify the security requirements for ad-hoc network layer and corresponding solutions. Another work [18] investigates the effect of wormhole attacks in MANETs where the virtual tunneling to misbehaving nodes use to conduct a variety of attacks. Their work presents a trust-based scheme for identifying and isolating nodes that create wormhole in the network without using any cryptographic scheme. Our immune-inspired technique builds misbehavior detection system that learns and detects new misbehaviors [19] using a complete network profile.

3. NEGATIVE SELECTION APPROACH

The Negative Selection Algorithm (NSA), developed by Forrest et al. [24], is based on the principles of self/non-self discrimination in the immune system. It can be summarized as follows:

- Define self as a collection S of elements in a feature space X , a collection that needs to be monitored. For instance, if X corresponds to the space of states of a system represented by a list of features, S can represent the subset of states that are considered as normal for the system.
- Generate a set F of *detectors*, each of which fails to match any string in S .
- Monitor S for changes by continually matching the detectors in F against S . If any detector ever matches, then a change is known to have occurred, as the detectors are designed not to match any representative samples of S .

The negative selection algorithm has been studied for more than 10 years in different applications, and there exist several variations of it [25]. We previously developed a real-valued negative selection algorithm [2], which will be applied here in wireless network security. In this work, we propose a hybrid approach for misbehavior detector generation and the algorithm for evolving them.

3.1 Anomaly detection

The anomaly detection process aims at distinguishing a new pattern as either part of self or non-self, given a model of the self (normal data) set. The problem space, denoted by X , in an n -dimensional space; the self set is denoted as S and let N be the complementary space of S . It is assumed that each attribute is normalized to $[0, 1]$, then

$$S \subseteq [0,1]^n, S \cup N = X, S \cap N = \emptyset.$$

Given the normal behavior of a system, S the characteristic function of S , defined as $\chi_S(p) = \begin{cases} 1, & \text{if } p \in S \\ 0, & \text{if } p \in N \end{cases}$ is used to distinguish between self and non-self.

In order to generate detectors through an evolutionary process, we used a structured genetic algorithm (sGA), which is suitable for encoding different detector shapes [2].

3.2 The Structured GA

A structured GA (sGA) is a type of evolutionary algorithm [5] that incorporates redundant genetic material, which is controlled by a gene activation mechanism. It utilizes multi-layered genomic structures for its chromosome i.e. all genetic material (expressed or not) is ‘structured’ into a hierarchical chromosome. The activation mechanism enables and disables these encoded genes. The implicit redundancy has the advantages of maintaining genetic diversity necessary in solving complex search and optimization applications. The capacity to maintain such diversity however depends on the amount of redundancy incorporate in the structure.

The sGA interprets the chromosome as a hierarchical structure; thus, genes at any level can be either active or passive, and high-level genes activate or deactivate sets of low-level genes. Thereby, the dynamic behavior at any level, whether the genes will be expressed phenotypically or not, is governed by the high level genes. Figure 1 shows a multi-level genomic structure, which is organized as follows: the highest level control bits will activate different gene sets. A typical example is a multi-shaped detector scheme as shown in Figure 2.

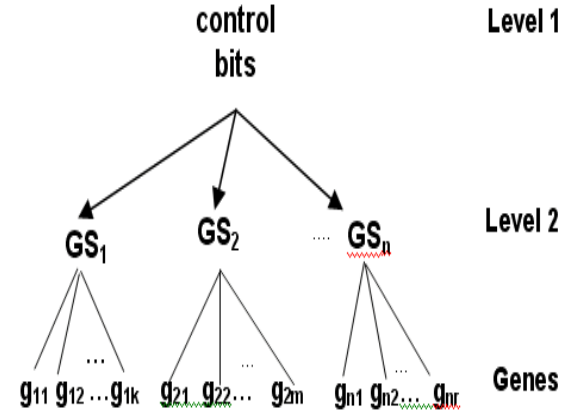


Figure 1. Structured GA representation of a chromosome with n different gene sets.

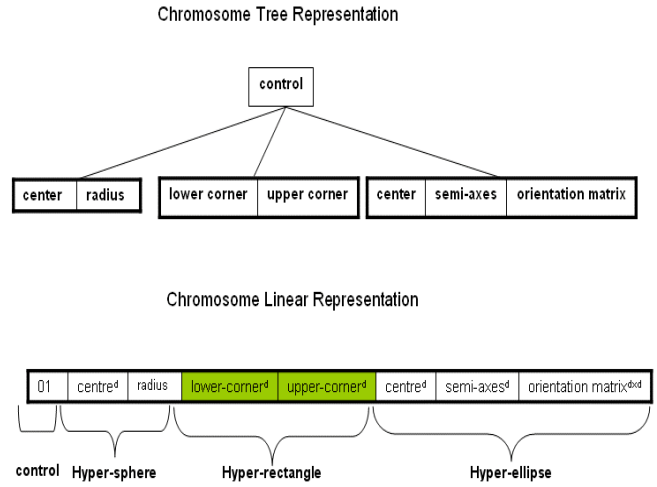


Figure 2. Encoding multi-shaped detectors: a chromosome having high level control and low level parameters for three different shapes: hyper-spheres, hyper-rectangles and hyper-ellipses detectors.

This illustration shows that the representation scheme for a chromosome tree has a control gene activating one of the shapes in phenotype space where each shape identifies a detector shape. A detector is defined in an n -dimensional space as a geometrical shape, such as a hyper-sphere, a hyper-rectangle or a hyper-ellipse. The *matching rule* is expressed by a *membership function* associated with the detector, which is a function of the detector-sample pattern distance [9] (Euclidean or any other distance measure). A set of good samples (also known as self), represented by n -dimensional points are given as inputs to the algorithm.

As depicted in Figure 3, the goal of the algorithm is to evolve a set of detection rules to cover the non-self space. The iterative process to generate a set of detectors (Figure 4) is driven by two main goals:

- Minimize overlap with self, and

- Make the detectors as large as possible and keep them separate from each other, in order to maximize the non-self covering (This is referred to as the coverage parameter in all our experiments).

In this work, we assumed that the self data points representing the events of good traffic network behavior while non-self data points represent the misbehaving event sequences.

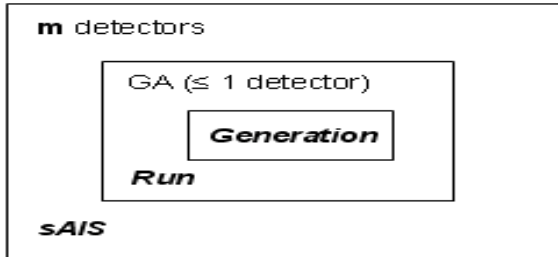


Figure 3. General framework for detector generation.

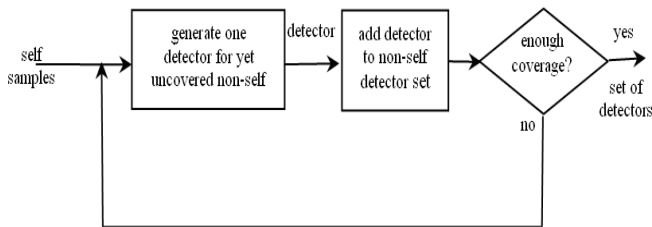


Figure 4. Modular subdivision of the detector generation process.

3.3 Detecting Flooding Attacks

From the work of [12] it was seen that 802.11b wireless networks suffers from some inherent flaws and are therefore prone to more attacks than wired networks because there is no need of any physical access to wireless networks. We first show how the above approach can be used to detect flooding attacks. We implemented two familiar attacks based on the guidelines in our previous work [12]. The two attacks are Denial of Service attack from an attacker machine outside the wireless network, and Denial of Service attack from a compromised machine inside the wireless network. The first attack was launched in a real environment consisting of some wireless stations and an Access Point (Figure 5), while the second one is implemented using a network simulator tool called NS2 (Figure 6). The technique mentioned underneath has been earlier used by us [1] in the study of some of the vulnerabilities [3, 21] and possible attacks on wireless networks (802.11b) to determine the efficiency of the novel detector generation system [1, 6] in detecting such attacks. The detection results indicate that in all the three cases the Negative Detectors were able to detect the attacks with good detection rate. We briefly illustrate the attack scenario and the results for the Ad-hoc network case. Two experimental setups for ad-hoc network attacks considered here are

similar to that described in [12], and the experiments are repeated here for a comparison only.

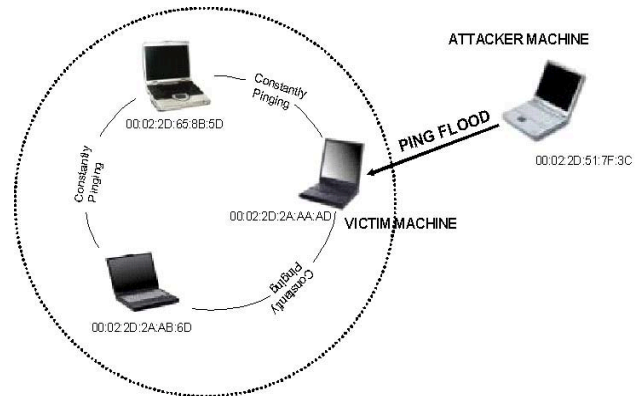


Figure 5. Ping Flood DoS in Ad-Hoc Network from an attacker machine outside the wireless network

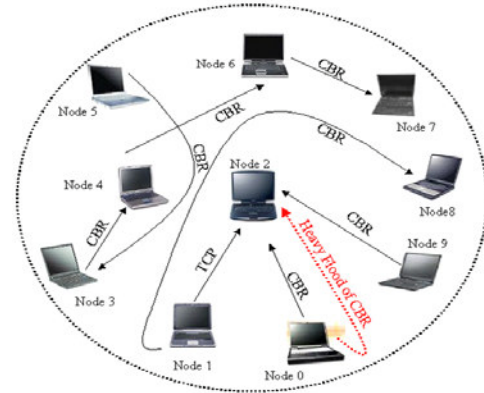


Figure 6. DoS attack by a compromised node in the WLAN implemented in NS-2 [12]

3.3.1 Scenario 1: Dos Attack from external compromised machine

The testbed comprises of an Ad-hoc WLAN with three wireless stations communicating with each other (Figure 5), and established normal traffic flow. An attacker machine launches a PING flood attack to a wireless node. This attack is launched by running the following command on the attacker machine:

```
$ ping -f <IPAddressofVictim> -s 1
```

where, *-f* option tells the machine to send flood of ICMP packets to a wireless node..

The ICMP (Internet Control Message Protocol) packet sequence numbers are collected at the Ad-hoc node different from the victim machine, which keeps pinging, on the victim machine throughout the experiments. The time taken for each packet to reach the destination node is noted when each ping (ICMP) packet is sent. It is observed that during the occurrence of an attack, there are some drops in

the ICMP packets. In addition, the time taken to reach the destination host increased when the attack was launched.

3.3.2 Scenario 2: DoS attack from an internal compromised machine

A network simulator (NS2 version 2.1b9a) is used to simulate an Ad-hoc network. Figure 6 shows a scenario with 10 nodes and the traffic flow among them. These nodes are labeled, ranging from Node 0 to Node 9. Constant Bit Rate (CBR) traffic is defined between Node 0 and Node 2, Node 3 to Node 4, Node 4 to Node6, Node 5 to Node 3, Node 6 to Node 7, Node 1 to Node 8, Node 9 to Node 2, Node 8 to Node 7 and File Transfer Protocol (FTP) traffic flows between Node 1 and Node2. The start times for all these traffics are preset. The attack is launched from Node 0 to Node 2. The attack is simulated as DoS with heavy traffic flow in a short time interval. During these periods when the attack was launched, the number of legitimate packets received by the victim node (Node 2) was reduced. The sequence numbers resulting from the connection between different nodes and Node 2 were collected.

We are using this approach in order to test our detection approach, although a more realistic approach (the complete network profile rather than monitoring a single node) is used in this work for misbehavior detection. The primary objective was to evaluate our model in terms of good detection rates and low alarm rates for wireless ad-hoc network’s MAC layer. Figure 7 and Figure 8 are our previously published results [12], reproduced here for showing the detection levels produced for the entire test data set, using a different detection rule set. The detection rates in that case is good for the second scenario whereas detection levels for the first scenario have high deviations. Similarly, in our results we could achieve very good detection rates for the second scenario with a few misses (false negatives) while considerably high detection rates in the first scenario as well. With further tuning of our algorithm, better results could be achieved.

Table 1. The Detection result for mixed shaped detectors for various self threshold parameters. Here **D.R.** is detection rate, **F.A.** is False Alarm rate, **F.N.** is False Negative is expressed as a percentage over the entire data points, **F.P.** is False Positive.

Ratio	Coverage	D.R.	F.A.	F.N.	F.P.
0.01	97.11	95.1	0	4.75%	0
0.02	96.4	96.45	0	4.05%	0
0.03	97.28	92.05	0	7.75%	0
0.05	98.2	87.1	0	11.57%	0

Table 2. The Detection result for mixed shaped detectors for various self threshold parameters. Here **D.R.** is detection rate, **F.A.** is False Alarm rate, **F.N.** is False Negative, **F.P.** is False Positive

Ratio	Coverage	D.R.	FA	F.N	F.P.
0.01	99.9	92.75	0	6.50%	0
0.02	99.86	91.1	0	8.50%	0
0.03	99.775	87.1	0	14.10%	0
0.05	99.77	86.9	0	13.70%	0

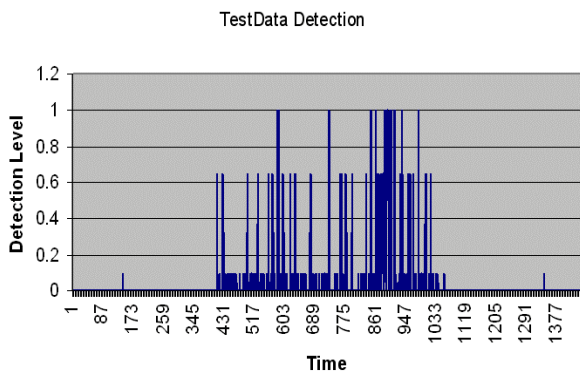


Figure 7. Detection level produced by the detection system for the test data set generated in scenario 3.1, indicating the abnormality in *icmp* sequence number.

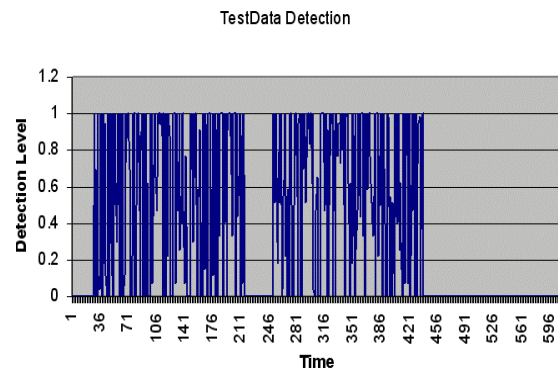


Figure 8. Detection level provided by the system for the test data set generated in Scenario 3.2

4. DETECTING ROUTER MISBEHAVIOR

In wireless ad-hoc networks, the Dynamic Source Routing (DSR) protocol is sometime used by the participating wireless nodes. Such networks are highly vulnerable to (packet) routing misbehavior due to malware, faulty or compromised nodes. We describe the simulation environment, the experimental setup, and results with performance metrics.

4.1 Simulation Environment

Global Mobile Information System Simulator (GloMoSim) provides a scalable simulation environment for large wireless and wireline communication networks. Its scalable architecture supports up to thousand nodes linked by a heterogeneous communications capability that includes multi-hop wireless communications using ad-hoc networking. The GloMoSim supports various settings including Scenario Topology, Mobility Radio and Propagation Models, MAC Protocol, and Routing Protocol. Using the application configuration file, the following traffic generators are supported: TELNET and CBR. More details in [17].

The visualization tool (Figure 9a) indicates packet transmissions among nodes within the power range. Yellow lines indicate links within the range, green indicates successful reception and red line is unsuccessful reception.

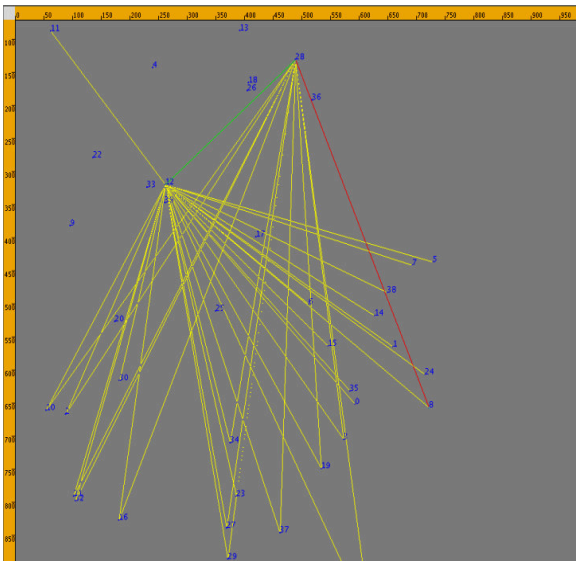


Figure 9a: Shows a simulated WLAN environment

Our simulated network comprises of mobile nodes acting both as terminals and information relays with sufficient infrastructure for radio communication. This setup relates to an ad-hoc network that usually utilizes multi-hop routing scheme. Each of these nodes participates in a common routing protocol, such as DSR (which is being considered for standardization). The elements of each of

these nodes are as shown in Figure 9b. As illustrated, each of these nodes in the ad-hoc network has a set of protocol layers clearly defined. In each of these layers, important protocol events are generated whose sequences are of particular interest in this work. For the DSR protocol block, main events of concern are the data packets sent/request, route packet sent/request, broken link error packets received, etc.

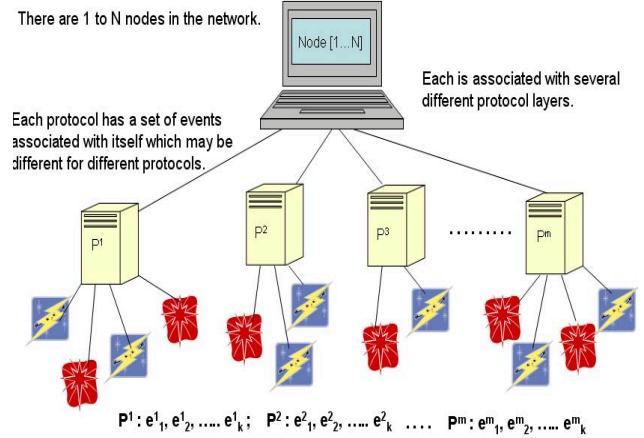


Figure 9b. A simplified scenario of a multi-hop ad-hoc wireless network nodes (a single node enlarged showing the basic elements for consideration in a wireless network for attack detection).

This collected sequence, stated as S in Figure 10, could be time-sliced ($\Delta t = 10s$, for example) over a period of time to aggregate (or log) active/non-active observed sequences per monitored node. These are categorized separately under each of the observer node. This simple scheme is applied to collect good and bad data in two separate simulations: one for learning and generating detectors while the other for performing misbehavior detection. Such profiles after assembling through proper pre-processing provides an overall behavior of the network routing traffic.

Node 0		Δt	Δt	Δt	Δt	Δt	Δt	Δt	Δt	Δt
Node 1	S	S	-	-	S	-	S	-	-	-
Node 2	-	S	S	-	-	S	-	-	S	S
.
Node N	S	-	-	S	-	-	-	-	-	S
Node 1	Node 0	-	-	-	-	S	S	-	-	-
Node 2	S	-	-	S	S	-	-	-	-	-
.
Node N	-	-	-	S	-	-	-	-	-	S
.
Node N	Node 0	S	-	-	-	-	S	-	S	S
Node 1	-	-	-	-	S	-	S	-	S	-
.
Node N-2	-	-	-	S	S	-	-	S	S	-
Node N-1	-	-	-	-	-	-	S	S	S	S

Figure 10. Shows a snapshot of a profile (logical) collected during the simulation run. S denotes an event sequence creation while “-” denotes an absence of any sequence. A short time interval Δt is used for data sampling.

A simplified scenario involving the multi-hop ad-hoc wireless network nodes assumes the Dynamic Source routing protocol that all the participating nodes are involved in commonly. The active nodes in the network are connected in bold lines. Certain nodes in this scenario are assumed to be sources of traffic initiation. Such nodes are specifically made observer nodes that monitor the activity of all the other active nodes based on the **RREQ** packets sent and/or received, **RERR** sent and/or received and **DATA** sent (IP source address not of monitored node) and or/received (IP destination address not of the monitored node). The profile of the monitored nodes is built up by the observer nodes thus forming the basis of a reputation system [4]. Such profiles can be assembled as shown in Figure 10. This will require proper pre-processing and a generalized assumption for aggregating a complete behavior of the nodes in the entire network. As the nodes might misbehave, it results in an Alert signal to propagate along the Source Route to reach the final originator.

As illustrated in Figure 9, these wireless elements are responsible for certain sequences of events in the routing protocol can lead us to build a useful profile of the Node under preview. Such profiles can be aggregated as a complete behavior sketch for the network system. Over the time, such profiles could either adapt or develop into new profile sets. Such a development can trigger the modification of the pattern detectors and hence lead to a highly stable and robust detection system. Each node in the system captures the events of interest (only at routing level associated with DSR protocol) about the monitored nodes in its neighborhood and collects series of small event sequences comprised of only the events of interest. These sets of event sequences are collected by each observing nodes for the monitored neighbor nodes for all the 40 nodes (except its own). The aggregation of these sequence sets by all the observer nodes over a monitored node helps to build its event profile. This is done as explained earlier and thus aggregates to build up the complete network profile. The events of interest or protocol events [19] recorded by a node are labeled for both transmitted and received cases as shown below (Table 3).

TABLE 3
Labeling different event types

Packets Transmitted	
Label	Event Type
A	RREQ
B	RREP
C	RERR
D	DATA sent and IP address not of monitored node
E	RREQ
F	RREP
G	RERR
H	DATA received and IP destination address is not of monitored node

The sequence could look somewhat like this for an event sequence **T**: (EAFBHHHEDEBHDHDDHDDH). A set of commonly occurring events' subsequence is used to produce a four dimensional vector denoting the number of occurrences for each of the four elements. To achieve this, four different subsequence expressions have been used.
 Expression1:- number of E in sequence (**T**); #E
 Expression2:-number of E with 1or no event followed by either an A or a B in sequence (**T**); #(EX(A|B)).
 Expression3:- number of H in sequence (**T**); #H
 Expression4:- number of H with one or none event followed by a D in sequence (**T**); #(H?D)

These expressions are evaluated one after the other on the event sequence such as **T** to produce a 4 dimensional vector as shown: **S = (3 2 7 6)**. Our aim is to collect such series of vectors both during the normal operation (training data) as well as during detection phase.

The schematic of data collection is shown in the Figure 10. This represents the initial preprocessed profile set as a matrix of nodes versus events trace that are sampled over the events' time series. The Nodes in the 1st column, labeled Node 0 to Node N (There are total of N +1 nodes in this assumption) are all behaving as observers for each of their neighbors except themselves. Each of these nodes build a small events' profile of their monitored neighbors (it could be some or all), during the activity time (only Δt times the total active monitored time). All these profiles are later aggregated for each monitored node to yield an events' dataset respectively, which is later used by the learning process. The **S** in this illustration refers to a short, heterogeneous event sequence of variable length collected in a short interval of time, Δt . It is illustrative of the generation of a small window of event sequence generally spoken to of as a trace in Δt period. Each **S** is more likely to be quite different from the other **S**'s, most of which belonging strictly to one of the classes - either well-behaving or misbehaving. There are certain intervals of observations where no sequences might be observed. These are represented as "-" in the above table. This generally happens for no or low activity for the concerned neighboring monitored node or even for monitored nodes beyond the listening range of the observer node. It is to be noted that the event trace sequence for a long observation interval typically 2 minutes or more might generate a single sequence that may well be over 1 Giga-bit long [13], thus making the generation of a large number of distinct patterns difficult. This will further make our task of generating appropriate detectors during the behavior learning process almost fruitless. Thus it helps to keep the observation window period small (about 10s - 20s for Δt).

The data set composed of series of **S** is normalized in each element of the vector, where the actual values of the variables are scaled to fit a defined range [0, 1] using 40% of maximum (addition factor) and 40% of minimum

(subtraction factor) value for each dimension in the data set. Any value in this normalization above the maximum and minimum is taken as 1.0 and 0.0 respectively. This normalization is done for training and testing alike.

4.2 Experimental Details

The incorporation of misbehavior into the network is the same as done in [19]. We reiterate for clarity. The nodes can be set to misbehave as a Boolean parameter. It can be set or reset. Using this implementation capability we could have different numbers of misbehavior set up (In our experiments, 5 10 and 20 were involved). The misbehavior are implemented in two ways - (1) Nodes neither forward route requests nor answer the route replies from their route cache. (2) The nodes do not forward data packets. The misbehavior probability is a control parameter such that the misbehaving nodes behave badly only during certain times in the simulation run. The same or different probabilities could be utilized in either case.

We used an implementation [20] of the routing packets traffic data using DSR protocol, in GloMoSim [22] that provides an excellent environment for wireless mobile ad-hoc networks. For data set collection, detection and analysis, crucial simulation and detection parameters, as defined in Table 4 were used. As we employ a simulated environment, certain designated nodes were labeled as misbehaving with a certain probability. High Telnet traffic and low CBR rates were tried to generate the overall traffic in this simulation. A configuration file for setting up these traffic types is provided with the simulator. The Global system configuration for the simulation is defined in a

separate file, which has support for DSR protocol in promiscuous mode. Table 4a enlists this information.

During the learning phase, all the nodes were made to behave well (we restricted any node from misbehaving) so that a generalized profile for the entire network traffic could be built and our potential detectors could be generated and trained. The simulation time was set for a long duration, so as to capture a considerably large and proper profile of the system as also to collect sufficient good behavior, which is an important concern for Anomaly detection problems.

During the detection phase, certain numbers of nodes (with known addresses) were made to misbehave. Our trained detectors were tested against this behavior towards the end of the simulation for the rates of detection and false positives in misdetection of the well behaved nodes. Different set of parameters (as given in Table 4b) were used for the detection phase in deciding the true positive (Detection Rate) classification and the false positive (False Alarm Rate) classification rates. A critical parameter called the “learning data threshold” was used for our purpose to keep a balance between high detection rate and low false alarm rate. This parameter is applied to each data point (a data point is a vector of 4 dimension data) so as to generalize the overall profile (as not all possible good behavior can be captured easily in any given profile, or in other words, no profile is ‘so called’ complete). In the results we could see the significant effects of this parameter with relation to the change in detection rates as well as false alarm rates. For our analysis, a threshold for detection was used, where a detection rate below 25% is not considered for detecting the misbehavior.

TABLE 4
Tabulation for Simulation and Detection parameters

4a. Simulation System parameters		4b. Detection System Parameters	
Parameter	Default value(s)	Parameter	Default value(s)
Routing protocol	DSR	Upper limit for Events sequence sets of a Monitored Node for learning	500
Simulation time	60 mins	Number of subsequences in a sequence set	4
Simulation area in meters	800x1000	Upper limit for the number of events in a sequence set.	40
Number of nodes	40	Upper limit for a sequence set collection	10s
Radio range	380 m	Misbehavior probability	0.8
Propagation Pathloss model	Two-ray	Learning data threshold	0.001 - 0.1
Mobility model	Random way point	Threshold for detection (% of Detection rate)	0.25
Mobility speed (no pauses)	1m/s	Mutation probability	0.05-0.1
Misbehaving nodes	5, 10, 20	Crossover probability	0.6
Traffic type	telnet, CBR	Normalized space range	[0.0, 1.0]
Payload size	512 bytes	Number of dimensions	4, 2
Frequency/rate	0.2-1s		
Radio-Bandwidth/link speed	2Mbps		

We ran two set of experiments:

- *Using All 4 Event types* (all 4 parameters): - This analysis involves the misbehavior detection for the nodes that are neither replying from its route cache nor forwarding route requests as well as non forwarding data packets.
- *Using more prominent event types* (last 2 parameters):- This analysis is more significant and pronounced if the initial route set up for the monitored node is not taken into account for misbehavior. It concerns more those misbehaviors that happen during the data packets' transfer among the nodes. This is more crucial for detecting those misbehaving nodes that do not forward data packets. Unless this assumption is made in the simulation, this analysis will fail.

It is to be noted that for this case, the first initial stages of route set-up event sequences data collected were disregarded (not collected). Our belief is that these events' subsequences are more effective for difference analysis however we need to perform more sets of rigorous experimentation and analysis to prove our claim. The results section however indicated better results than that involving all the 4 parameters.

Ten independent replications of all experiments were performed in order to obtain high degrees of confidence intervals. The Average, best and worst cases were recorded in each case.

5.1 Using All 4 parameters:

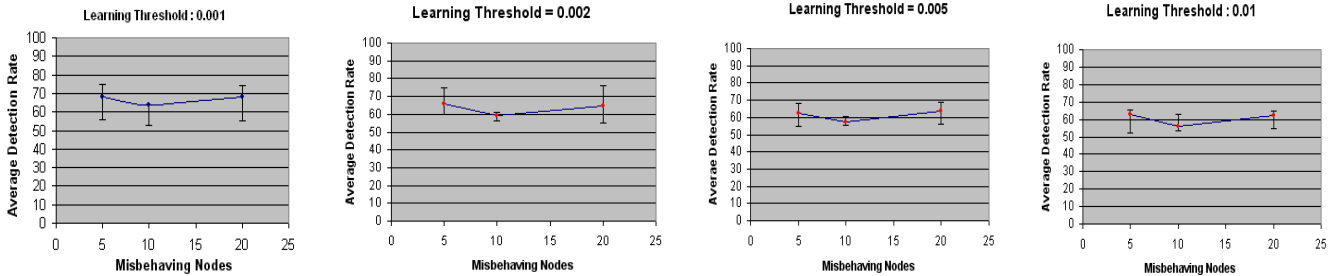


Figure 11. Average Detection Rates for all misbehaving nodes - 5, 10 and 25 with learning thresholds of 0.001, 0.002, 0.005 and 0.01. The Learning Threshold parameter has a significant effect on the detection rate in all three cases.

4.3 Performance measures

The experimental simulation aims at finding and reporting the detection behavior of the generated nodes in correctly identifying the misbehaving nodes as well as how well it could identify such deviations in behavior. Our experimental results are based on the following metrics:

1. Average detection rates for the misbehaving nodes; defined as detection rate (D.R.) = (true positives)/(true positives + false negatives).
2. Average False Alarm rates for misclassifying the well behaving nodes; defined as false alarm rate (F.A.R) = (false positives)/(false positives + true negatives).

where, true positives refer to the number of abnormal cases identified as abnormal in the given data set of vector points; false positives refers to the number of normal cases mistakenly identified as abnormal; true negatives refer to the number of normal event sequences (normal cases) in the entire sequence correctly identified as normal while false negatives are the count of those abnormal sequences that the detector set classified as normal.

Whenever and wherever we refer to positive detection and misclassifications, we refer to these metrics respectively.

5. EXPERIMENTAL RESULTS

Based on the Performance metrics, we present the following results here for clarity. We present two series of experiments for this simulation environment. Other dependences like the effect of mobility, speed and simulation area were not considered in these experiments.

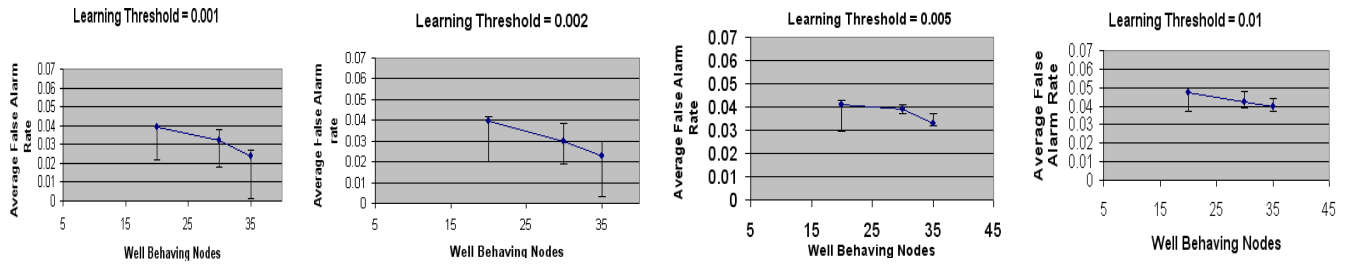


Figure 12. The average misclassification (false alarm rates) for well behaving nodes - 20, 30 and 35 with learning thresholds of 0.001, 0.002, 0.005, 0.01. The false positives are within the upper limit of 0.03 in most cases.

5.2 Using last 2 parameters:

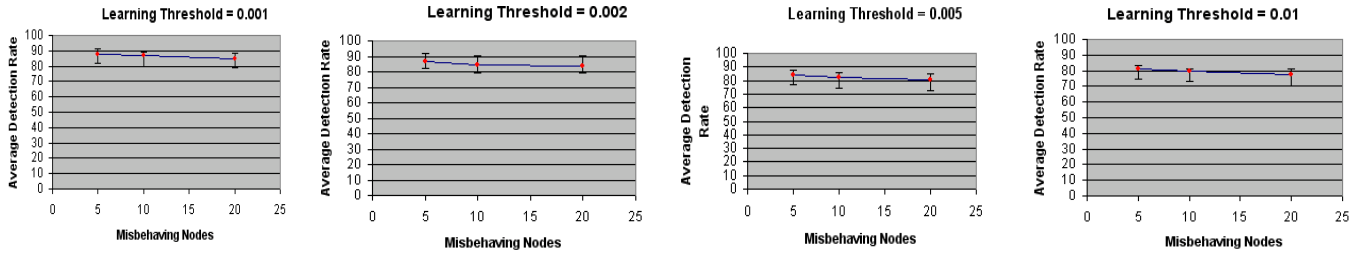


Figure 13. Average Detection Rates using partial parameters for learning (ignoring the routing setup data) misbehaving nodes - 5, 10 and 25 with learning thresholds of 0.001, 0.002, 0.005 and 0.01. The Learning Threshold parameter has a significant effect in each of the cases.

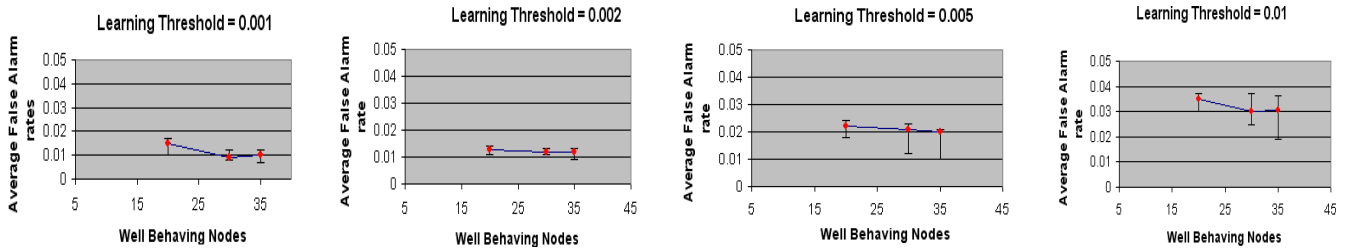


Figure 14. The average misclassification or false alarm rates for the partial events' sequence are within an upper limit of 0.02 in most cases.

5.3 Analysis of Results

- **Detection capabilities:** In all these experiments, average detection rates were substantially higher than the detection threshold. This demonstrates that all misbehaving nodes were detected and classified as misbehaving while the well-behaving nodes were classified as normal. Depending on the detector sets the average rates of detection and misclassifications were close to the median of the best and worst cases. This highlights the average good coverage of the multi-shaped detectors. The misclassification values for the good nodes were within a small range of 0.001 and 0.03. We observed that the mixed representation scheme using structured hybrid representation produces more stable and better detection rates [2].
- **Impact of the learning threshold parameter:** It is clearly discernable from the experimental results that most of the misbehaviors are subtle and hence difficult to distinguish from the benevolent behaviors. This results in high false negatives thus lowering the

detection rates. Thus a lower threshold value (0.001) has a higher detection rate compared to the higher ones. For a given learning threshold value, the number of misbehaving nodes also play a distinguished role. As the number of misbehaving nodes increases, the detection rates decrease slightly.

- **Effect of using partial events' subsequence:** As already discussed earlier, this criteria (using the last two parameters in the events' sequence) had an improved effect in our detection results though detailed analysis needs to be performed in real network environment. It is observed however that subsequence patterns have a obvious effect in better differentiating the behavior than that involving all the four.

6. CONCLUSIONS AND FURTHER WORK

We have implemented a hybrid learning technique in which learning of good behavior and detection of misbehavior is carried out in a simulated ad-hoc wireless environment. The hybrid approach generated multi-shaped

detectors (a set of rules) defining a boundary for the unknown regions of the behavior space. In certain cases, it would require a large amount of training (good) data as well as the number of efficient detectors. Sometimes, for a very subtle misbehavior detection, more detectors with better coverage are necessary. We used a small subset of protocol event sequences, which represents a subset of the entire traces of observed protocol events, so the dataset in this work was somewhat within limits. With the use of learning threshold for the detectors' learning, certain subtle abnormalities are supposedly captured. We have found that the performance of the system is very sensitive to some parameters that require careful tuning. The preprocessing of raw data for matching the misbehavior needs to be carefully analyzed.

Since we could successfully detect the misbehavior by using the complete network behavior, further work should utilize and maintain separate collections of detectors in the mobile wireless nodes. The network nodes in this case monitor and produce the profile of their neighbors, those that are within their listening range and could listen to sufficient number of packets from neighbors denoting enough activity data in such nodes so as to be able to generate individual sets of detectors for each node. Later these detectors can be used to detect (if any) misbehaviors among the neighboring nodes (within listenable range with sufficient activity). Each of these nodes detection rates could be used to score individual nodes detection capability. As we have an established technique set forth, the new task would be our next logical step for implementation.

REFERENCES

- [1] S. Balachandran. *Multi-shaped Detector generation using Real-valued representation for Anomaly Detection*, Masters Thesis, Advisor: Dr. Dipankar Dasgupta, The University of Memphis December 2005.
- [2] S. Balachandran, D. Dasgupta, F. Nino, D. Garrett. *A General Framework for Evolving Multi-Shaped Detectors in Negative Selection*. Submitted to the IEEE Transactions on Evolutionary Computation, January 2006.
- [3] N. Borisov, I. Goldberg and D. Wagner. 2001. *Intercepting Mobile Communications: The Insecurity of 802.11*. DRAFT.
- [4] S. Buchegger and J.Y. Le Boudec. *The Effect of Rumor Spreading in Reputation Systems for Mobile Ad-hoc Networks*. In Proceedings of WiOpt '03: Modeling and Optimization in Mobile, Ad-hoc Networks, As Hoc and Wireless Networks", Sophia-Antipolis, France, March 2003.
- [5] D. Dasgupta and D. R McGregor. *sGA: A structured Genetic Algorithm*. Research Report IKBS-11-93, April 1993.
- [6] D. Dasgupta and F. Gonzalez. *An Immunity-Based Technique to Characterize Intrusions in Computer Networks*. In the Journal IEEE Transactions on Evolutionary Computation, Volume: 6, Issue: 3, Page(s):281-291, June, 2002.
- [7] H. Debar. IBM Zurich Research Laboratory. http://www.sans.org/resources/idfaq/behavior_based.php
- [8] J. Gomez, F. Gonzalez, M. Kaniganti and D. Dasgupta. *An Evolutionary Approach to Generate Fuzzy Anomaly (attack) Signatures*. In the proceedings of the Fourth Annual IEEE Information Assurance Workshop, page(s): 251-259, West point, NY June 18-20, 2003.
- [9] F. González. *A study of Artificial Immune Systems Applied to Anomaly Detection*. PhD. Dissertation, Advisor: Dr. Dipankar Dasgupta, The University of Memphis, May 2003.
- [10] Y. C. Hu, A. Perrig, and D.B. Johnson. *Packet leases: A defense against wormhole attacks in wireless networks*. In proceedings of the 6th Annual International Conference on Mobile Computing and Networking (MobiCom), pp. 231-242, 2000.
- [11] D. B. Johnson, D.A. Maltz, and Y. Hu. *The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR)*. IETF MANET Working Group, Internet Draft, July 2004.
- [12] M. Kaniganti. M. *An Agent-Based Intrusion Detection System for Wireless LANs*, Masters Thesis, Advisor: Dr. Dipankar Dasgupta. The University of Memphis, December 2003.
- [13] J. Kim and P.J. Bentley. *The Artificial Immune Model for Network Intrusion Detection*: 7th European Conference on Intelligent Techniques and Soft Computing (EUFIT'99), Aachen, Germany.
- [14] Y. Lim, T. Schmoyer, J. Levine and H. L. Owen. June 2003. *Wireless Intrusion Detection and Response*. In Proceedings of the 2003 IEEE workshop on Information Assurance United States Military Academy, NY: West Point.
- [15] S. Marti, T.J. Giuli, K. Lai, and M. Baker. *Mitigating routing misbehavior in mobile ad hoc networks*. In proceedings of MOBICOM 2000, pages 255-265, 2000.
- [16] P. Michiardi, and R. Molva. *Ad Hoc Network Security*. In the HANDBOOK OF INFORMATION SECURITY, Volume 1, pp. 787-806, ed. Hossein Bidgoli.
- [17] J. Nuevo. *A Comprehensible GloMoSim Tutorial*, INRS Université du Québec, March 4, 2004.
- [18] A. A. Pirezada and C. McDonald. *Detecting and Evading Wormholes in Mobile Ad-hoc Networks*. In the proceedings of the International Journal of Network Security, Vol.3, No.2, pp.191-202, September, 2006.
- [19] S. Sarafijanovic and J.Y. Le Boudec. *An Artificial Immune System for Misbehavior Detection in Mobile Ad-Hoc Networks with Virtual Thymus, Clustering, Danger Signal and Memory Detectors*. In Proceedings of ICARIS-2004 (Third International Conference on Artificial Immune Systems), pp. 342-356, September 13-16, 2004, Catania, Italy.
- [20] Simulation Code: <http://lcawww.epfl.ch/ssarafij/ais-code>.
- [21] A. Stubblefield, J. Ioannidis, and A. Rubin. *Using the Fluhrer, Mantin, and Shamir attack to break WEP*. 2002. In Proceedings of the 2002 Network and Distributed Systems Security Symposium.
- [22] X. Zeng, R. Bagrodia, and M. Gerla. *GloMosim: A library for parallel simulation of large scale wireless networks*. In the proceeding s of the 12th workshop on Parallel and Distributed Simulations-PDAS'98, May 26-29, Banff, Alberta, Canada, 1998.
- [23] Y. Zhang and W. Lee. August 6-11, 2000. *Intrusion Detection in Wireless Ad-Hoc Networks*. In Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking, Boston: Massachusetts.
- [24] S. Forrest, A.S. Perelson, L. Allen, R. Cherkuri. *Self-nonsel discrimination in a computer*. In the proceedings of the 1994 IEEE Symposium on Research in Security and Privacy, IEEE Computer Society Press. 1994.
- [25] Sanjay Goel and Stephen F. Bush. *Kolmogorov Complexity Estimates For Detection Of Viruses In Biologically Inspired Security Systems: A Comparison With Traditional Approaches*, Complexity Journal, vol. 9, issue 2, Nov-Dec 2003, Special Issue: "Resilient & Adaptive Defense of Computing Networks".