PROCEEDINGS OF SPIE

SPIEDigitalLibrary.org/conference-proceedings-of-spie

An artificial immune system for securing mobile ad hoc networks against intrusion attacks

William S. Hortos

SPIE.

An artificial immune system for securing mobile ad hoc networks against intrusion attacks

William S. Hortos
Florida Institute of Technology, Orlando Graduate Center, 3165 McCrory Place, Suite 161, Orlando, FL 32803

ABSTRACT

To mitigate the problem of intrusion attacks by malicious nodes in mobile *ad hoc* networks (MANETs), security attributes and quantifiable trust levels, unique to the MANET's transient, self-organizing topology, augment or replace traditional protocol metrics of throughput, packet delay and hop-count in the *ad hoc* route discovery procedures. The new features are unique to the candidate security protocol, which views security as a quality metric to improve the relevance of the routes discovered by established reactive *ad hoc* routing protocols. Attributes of a secure route are identified in order to define the appropriate metrics to quantify the "level of security" associated with the protocol messaging and the detection of malicious activities by some intrusive nodes. A state vector of features and metrics based on the published Secure Routing Protocol (SRP) for MANETs is constructed to encode network security characteristics. This route discovery protocol mitigates the detrimental effects of various malicious behaviors to provide trustworthy connectivity information. The protocol ensures that fabricated, compromised, or replayed route replies would either be rejected or never reach the querying source node. In this paper, the pattern of values, taken by the state vector of the SRP features in the route request, discovery and reply operations, are analyzed to detect evidence of intrusion attacks by malicious nodes that could lead to denial of service and network shutdown. The pattern analysis applies a technique based on negative selection found in natural immune systems that can detect extraneous patterns in the (nonself) space that is the complement of vector values associated with correct route discovery and route maintenance. The immune system is well-suited to the distributed nature of the MANET. It does not rely on a central controller, but instead uses a distributed detection and response mechanism in order to respond to foreign invaders, mirroring the operation of the route discovery and selection process in the presence of intrusive or malicious nodes. Furthermore, this pattern detection approach is suitable for the difficult problem of passive or hidden security threats. Based on the SRP features of the state vector, an artificial immune system (AIS) is constructed as a hierarchy of rules to detect different types of intrusive activity within the MANET. The pattern detection rules in the complement (nonself) space are generated in an evolutionary manner using a genetic search algorithm. The effect of the genetic search is to discern the varying levels of abnormal behavior in the MANET protocol transactions. The efficacy of the AIS technique is compared to the positive characterization method based on nearest-neighbor classification. Initial evaluations of the detection scheme are performed to validate the AIS-based method using training and test data sets, generated from intrusion scenarios simulated from various threat models and security-aware modifications to reactive MANET routing protocols. These results are reported along with a performance analysis comparing the AIS approach with competing techniques. Conclusions about the AIS application to MANETs using the SRP are discussed.

Keywords: Artificial immune system, detection rules, dynamic routing protocols, genetic algorithms, mobile *ad hoc* networks, pattern recognition, intrusion detection, malicious nodes

1. INTRODUCTION

A mobile ad hoc network (MANET) has characteristics that contrast sharply with fixed networks or last-hop wireless networks. First, there is no infrastructure support. All routers are mobile and can communicate with each other only when they are in transmission range. Second, ad hoc wireless nodes are resource-constrained, with limited processing and memory capacity, and are usually powered with batteries. Finally, the wireless medium can be easily eavesdropped. No part of the network is dedicated to support individually any specific network functionality, with routing (topology

Intelligent Computing: Theory and Applications, Kevin L. Priddy, Peter J. Angeline, Editors, Proceedings of SPIE Vol. 5103 (2003) © 2003 SPIE · 0277-786X/03/$15.00

discovery, data forwarding) being the most prominent example. Other functions that cannot rely on a central server, yet are of high relevance to this work, are naming services, certification authorities (CA), directory and other administrative tasks. Even if such services were assumed, their availability could not be guaranteed, due to either the dynamically changing topology that could easily result in a partitioned network, or the congested links close to the node acting as the central security server. Moreover, MANET performance constraints, such as, delay in acquiring responses from the assumed infrastructure, would pose further challenges.

Absence of both a fixed infrastructure and the attendant authorization facilities impede the usual practice of establishing a line of defense, separating nodes into trusted and non-trusted classes. Such a distinction is based on a security policy, the possession of the necessary credentials, and the ability for nodes to validate them. In a MANET there may be no ground for *a priori* classification; all nodes are required to cooperate in supporting network operations, while no prior security association can be assumed for all the nodes. Furthermore, freely roaming nodes form transient associations with their neighbors, join and leave MANET sub-domains independently and without notice, making it difficult to have a clear picture of ad hoc network membership. In particular, any node may compromise the routing protocol functionality by disrupting the route discovery process. The mechanisms currently incorporated in MANET routing protocols cannot cope with disruptions due to such intrusive behavior. Thus, especially in networks of many nodes, no form of established trust relationships among the majority of nodes can be assumed.

In this paper, a route discovery protocol is first introduced that mitigates the detrimental effects of such intrusive and malicious behavior to provide accurate connectivity information. The routing protocol, Secure Routing Protocol (SRP) cited here has been developed to guarantee that fabricated, compromised, or replayed route replies would either be rejected or never reach back to the source node of the route request.[1] Furthermore, the protocol responsiveness is protected under different types of security attacks that exploit the routing protocol itself. The sole requirement of the SRP is the existence of a security association (SA) between the node initiating the query and its desired destination. No assumption is made regarding the intermediate nodes, which may exhibit arbitrary and malicious behavior. The scheme is robust in the presence of a number of non-colluding nodes and provides accurate routing information rapidly.

Since the level of trust in a traditional ad hoc network cannot be measured or enforced, intrusive or compromised nodes may participate directly in the route discovery process to intercept and filter routing protocol packets to disrupt communication. Compromised users may use the information gleaned from relayed packets to mount an attack or anticipate countermoves in order to gain advantage. Since there is no penalty or punishment for misbehavior, in general, nodes have no incentive to behave well. Malicious nodes can insert spurious data into routing packets that cause routing loops, induce long time-outs, advertise false or exaggerated metrics, replay old routing updates, etc.

Traditional MANET routing protocols place complete trust on the nodes and are thereby vulnerable to any or all of these attacks. Other attacks, e.g., physically destroying nodes or jamming broadcast signals, exist. A comprehensive solution to all security attacks addresses these issues as well. In this paper, the focus is on threats from intrusion attacks associated with routing protocols and routing misbehavior.

Following this introduction, Section 2 provides an overview of published approaches for protocols that secure MANET routing against attacks by malicious nodes. In Section 3, the structure and operation of a preferred approach to routing security, the Secure Routing Protocol (SRP), are described in detail. The features, with which SRP augments the packet headers of an underlying MANET routing protocol, are used to form the samples in the development of anomaly detection approaches. In Section 4, the anomaly detection problem for MANET intrusion attacks is defined. Two approaches to its solution are developed: self-nonself positive characterization, and the negative-characterization, artificial immune system (AIS). The generation and selection of rules by genetic algorithms (GAs) for the AIS method are presented in Section 4. In Section 5, the performance of the two anomaly detection methods is evaluated using data produced in various simulated scenarios of intrusion attacks on MANET routing. Section 6 summarized the performance evaluation. Lastly, Section 7 presents conclusions and suggestions for further research on intrusion detection.

2. BACKGROUND ON SECURE ROUTING PROTOCOLS FOR MANETS

Despite the fact that security of MANET routing protocols is seen to be a major roadblock to commercial application of this technology, only a limited number of solutions that target route discovery have been based on approaches for fixed-infrastructure networks, ignoring the distinctive MANET challenges. Some secure protocols proposed in the literature

address the problem of secure data forwarding, using two mechanisms that (i) detect misbehaving nodes and report such events and (ii) maintain a set of metrics, reflecting the past behavior of other nodes have been proposed to alleviate the detrimental effects of packet dropping.[2] Unfortunately, the path metrics and tokens still allow malicious nodes to hide their activity and issue fake alerts. Others issue tokens or currency to prevent route breaks and packet dropping, but rely on an on-line certification authority, tamper-resistant modules and the high overhead of hop-by-hop public key encryption.[3]

In other approaches, the protection of the route discovery process has been regarded as an additional quality-of-service (QoS) issue, where routes are selected based on certain quantifiable security criteria.[4] Nodes in a MANET subnet are classified into different trust and privilege levels. A node initiating a route discovery sets the desired security level for the route; i.e., the required minimal trust level for nodes participating in the query/reply propagation. Nodes at each trust level share symmetric encryption and decryption keys. Intermediate nodes of different levels cannot decrypt in-transit routing packets, or determine whether the required QoS parameter can be satisfied, and simply drop them. Although this scheme provides protection (e.g., integrity) of the routing protocol traffic, it does not eliminate false routing information maliciously provided by intrusive nodes. Moreover, the proposed use of symmetric cryptography allows any node to corrupt the routing protocol operation within a level of trust, by mounting virtually any attack that would be possible without the presence of the scheme. Finally, the assumed supervising organization and the fixed assignment of trust levels does not apply to the MANET paradigm. In essence, the proposed solution transcribes the problem of secure routing in a context where nodes of a certain group are assumed to be trustworthy, without actually addressing the global secure routing problem.

The SRP was selected as the basis for the detection approach, as it is the only protocol found thus far in published literature that ensures that a node initiating a route discovery will be able to identify and discard replies providing false topological information, or, avoid receiving them entirely. In this way, SRP approach is distinct from the Internet-related solutions [5], which require the existence of a trust structure that encompasses *all nodes* participating in routing, and may rely on network management operations to detect routing instabilities. The novelty of SRP, as compared with other MANET secure routing schemes, is that false route replies, as a result of malicious node intrusions, are discarded partially by benign nodes while in-transit towards the querying node, or deemed invalid upon reception. Most importantly, the above-mentioned objectives are achieved with the existence of a security association between the source and destination nodes alone, *without* the need for intermediate nodes to cryptographically validate control traffic.works has been published on the subject. Such efforts have mostly concentrated on the aspect of data packet forwarding, disregarding the critical issue of route discovery. Conversely,

3. SECURE ROUTING PROTOCOL

The widely accepted technique in MANET route discovery of broadcasting query packets is the basis of the SRP. Specifically, as routing-query packets traverse the network, the intermediate relaying nodes append their identifiers (e.g., IP address) in the query packet header. When one or more queries arrive at the desired destination, replies that contain the accumulated routes are returned to the querying source node; the source then may use one or more of these routes to forward its data. Reliance on this basic route-query broadcasting mechanism allows SRP to be applied as an extension of any one of the existing MANET routing protocols. In particular, the Dynamic Source Routing (DSR) [6] and the interzone routing protocol (IERP) [7] of the Zone Routing Protocol (ZRP) [8] framework are two protocols that can be extended directly to incorporate SRP. Other protocols, such as, Associativity-based Routing (ABR) [9], can be combined with SRP with minimal modifications to achieve the security goals of the SRP protocol.

In SRP, route replies that are validated and accepted by the querying node provide accurate connectivity information, despite the presence of powerful adversaries. The protocol has been shown to be robust against a set of attacks that attempt to compromise the route discovery, under the assumption of non-colluding adversarial nodes.[1]

3.1 Basic assumptions

It is assumed that the networks use bi-directional communications between a pair of nodes. A security association (SA) between the source node *S* and the destination node *D* is assumed. The trust relationship could be established, for example, through knowledge of the public key of the other communicating end. The two nodes can negotiate a shared

secret key, e.g., via the Elliptic Curve Diffie-Hellman algorithm [10, 11], then, using the SA, verify that the principal that participated in the exchange was indeed the trusted node. In the remaining discussion, the existence of a shared key $K_{S,D}$ is assumed. The SA is bi-directional in that the shared key can be used for control traffic flow in both directions, provided relevant state information is maintained for each direction.

The existence of the SA is justified, because the end hosts chose to use a secure communication scheme and, thus, should be able to authenticate each other. For example, such a pair of nodes could have performed a secure key exchange, or an initial distribution of credentials.[12] However, the existence of SAs with any of the intermediate nodes is unnecessary. Finally, it is required that the end nodes be able to use static or non-volatile memory.

Intrusive nodes may attempt to compromise network operation by exhibiting arbitrary, Byzantine behavior.[13] They can corrupt, replay, and fabricate routing packets. They may attempt to misroute them in any possible manner and, in general, they cannot be expected to properly execute the routing protocol. Although a set of intrusive nodes may mount attacks against the protocol concurrently, it is assumed that nodes are not capable of colluding within one step of the protocol execution; that is, within the period of broadcasting one query and reception of the corresponding replies.

The underlying data link layer (e.g., IEEE 802.11b [14]) provides reliable transmission on a link basis, without any requirement of data link security services, such as the Wired Equivalent Protocol (WEP) function. Moreover, links are assumed to be bi-directional, a condition satisfied by the proposed medium access control (MAC) protocols, especially those employing the Request-to-Send (RTS)/Clear-to-Send (CTS) dialogue. It is assumed that a one-to-one mapping between MAC and IP addresses exists. Finally, the broadcast nature of the radio channel requires that transmissions are received by all neighbors of the transmitting node.

3.2 Functional overview

The SRP scheme combats attacks that disrupt the route discovery process and guarantees, under the aforementioned assumptions, the acquisition of correct topological information. It incorporates mechanisms that protect network functionality from intrusion attacks that exploit the protocol itself, in order to degrade network performance and possibly lead to denial of service (DoS).

The source node S initiates the route discovery, by constructing a route request (RREQ) packet identified by a pair of identifiers: a *query sequence number* and a *random query identifier*. The source and destination nodes and the unique (with respect to the pair of end nodes) query identifiers are the input for the calculation of the Message Authentication Code (MAU) [15], along with $K_{S,D}$. In addition, the identities (IP addresses) of the traversed intermediate nodes are accumulated in the RREQ packet.

Intermediate nodes relay RREQs, so that one or more query packets arrive at the destination, and maintain a limited amount of state information regarding the relayed queries, so that previously seen RREQs are discarded. Moreover, the intermediate nodes provide feedback in the event of a path break, and they may provide route replies in some cases.

The RREQs reach the destination D, which constructs the route replies (RREPs), calculates a MAU covering the RREP contents, and returns the packet to S over the reverse of the route accumulated in the respective request packet. The destination responds to one or more request packets of the same query, so that it provides the source S with as diverse a topological view as is available in the network. The number of replies and the time-window D allocates for replies to a specific query are design parameters. Moreover, S could provide an indicator of the required diversity, so that D can regulate the number of replies. The querying node validates the replies and updates its topological view.

As an illustrative example, consider the topology of ten nodes in Figure 1. S queries the network to discover one or more routes to D. The nodes M_1 and M_2 are two malicious intermediate nodes. The query request is denoted as a list $\{Q_{S,D};\ n_1,n_2,..,n_k\}$, with $Q_{S,D}$ denoting the SRP header for a query searching for D and initiated by S. The $n_i, i \neq \{1,k\}$, are the IP addresses of the traversed intermediate nodes and $n_1 = S$, $n_k = D$. The route reply is denoted as $\{R_{S,D};\ n_1,n_2,..,n_k\}$. A number of scenarios of possible security attacks by the two malicious nodes can be considered to explain the response by the SRP.[1] Due to the scope and goals of this paper, that discussion is omitted here.

3.3 Protocol description

The SRP introduces a set of new features that can be incorporated in the underlying routing protocol with low overhead. In principle, the underlying protocol can retain mechanisms, such as the control of the query propagation, the rate of query generation, and the neighbor discovery protocol, if present. SRP extends the basic routing protocol by enforcing rules on the format and propagation of RREQ, RREP, and error messages, by introducing additional functionality.

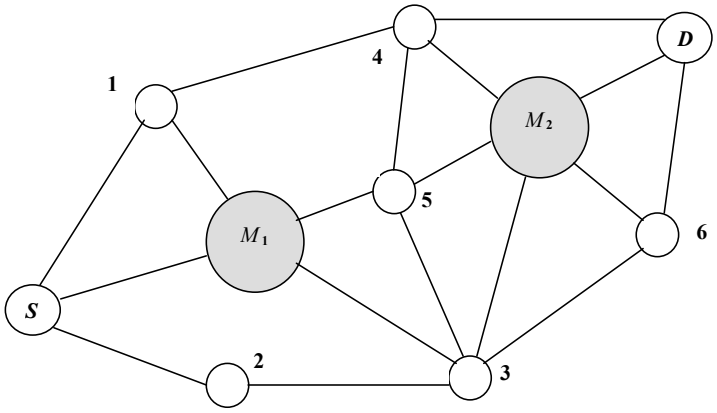

Figure 1. Example topology: S wishes to discover a route to D in the presence of two malicious nodes, M_1 and M_2.

In short, SRP makes efficient use of the SA between the communicating nodes S and D. RREQ packets verifiably propagate to the destination (in the general case) and RREP packets are returned to S strictly over the reverse route, accumulated in the RREQ packet. Similarly, route error messages can be generated only by nodes that lie on the route reported as broken. In order to ensure this critically important function, SRP explicitly determines the interaction with the network layer; i.e., the Internet protocol (IP) function. Furthermore, it provides an innovative way of query identification, which protects the query propagation and the end nodes from DoS attacks. Finally, propagating query packets are handled locally by a priority scheme that enhances the robustness and the responsiveness of the protocol.

The SRP parameters are used to extract features used in the samples input to the AIS algorithms. The features introduced by SRP require the addition of a six-word header. The SRP header is integrated into the structure of the underlying routing protocol header as an additional IP option, and covers most portions of the routing protocol datagram. Different types of SRP messages are distinguished with the help of the 1-byte Type field. Primary considerations are the augmentation of RREQ and RREP packets and each message type is described individually.[1] It is also possible for SRP to operate in a more general setting, where, for example, a route reply is appended to a *data packet*.

Route request. A source node S maintains a *Query Sequence Number,* Q_{seq}, for each destination with which it securely communicates. This 32-bit sequence number increases monotonically, for each RREQ generated by S, and allows D to detect outdated RREQs. The Q_{seq} is initialized at the establishment of the SA and although it is not allowed to wrap around, it provides approximately a space of four billion query requests per destination. If the entire space is used, a new security association has to be established.

For each outgoing RREQ, S generates a 32-bit random *Query Identifier*, Q_{ID}, which is used by intermediate nodes as a means to identify the request. Q_{ID} is the output of a secure pseudo-random number generator [16]; its output is statistically indistinguishable from a truly random one and is unpredictable by an adversary with limited computational power. Since intermediate nodes have limited memory of past queries, uniqueness and randomness can be efficiently achieved, by using a one-way function, e.g., a secure hash standard (SHA-1 [17]), with a small random seed as input. This renders the prediction of the query identifiers practically impossible, and combats an attack where malicious nodes simply broadcast fabricated requests only to cause subsequent legitimate queries to be dropped.

Both Q_{ID} and Q_{seq} are placed in the SRP header, along with appropriate *Type* value and the *Request Message Authentication Code* (MAU). The MAU is a 96-bit long field, generated by a keyed hash algorithm [15], which calculates the truncated output of a one-way or hash function (e.g., SHA-1 or MD5 [18]). Input to this one-way function is the entire IP header, the underlying routing protocol RREQ packet, and most importantly, the shared key $K_{S,D}$. The RREQ fields

that are updated as the packet propagates towards the destination, i.e., the accumulated addresses of the 8 intermediate nodes in Figure 1, and the IP-header mutable fields are excluded from this encryption.

Query handling/propagation. Intermediate nodes parse the received RREQs in order to determine whether an SRP header is present. If not, they process the packet as described in the underlying routing protocol specification. Otherwise, the intermediate nodes extract the Q_{ID}. The S and D addresses are also extracted in order to create an entry in the query table. Queries with Q_{ID} matching one of the table entries for the same pair of end nodes are discarded. Otherwise, the intermediate nodes re-broadcast the RREQ.

Intermediate nodes also measure the *frequency of queries* received from their neighbors, in order to regulate the query propagation process. All nodes self-regulate the generation of new RREQs, in order to keep the control traffic overhead low. However, malicious nodes probably act selfishly and avoid backing off before generating a new RREQ, or generate queries at the highest possible rate, thereby consuming network resources and degrading the routing protocol performance.

In order to ensure the responsiveness of the routing protocol, each benign node maintains a *priority ranking* of its neighbors according to the corresponding observed rate of queries. The highest priority is assigned to nodes generating (or relaying) requests with the lowest rate, the lowest priority to neighbors that generate queries more frequently. Then, quanta are allocated proportionally to the priorities and, within each class, queries are serviced in a round-robin manner.

As immediate neighbors of a malicious node observe a high rate of incoming queries, they update the corresponding priority. Moreover, low-priority queries not serviced are eventually discarded. In this way, non-malicious queries are only affected for a time period equal to the time it takes to detect and update the priority assigned to a misbehaving neighbor. At the same time, the round-robin operation provides additional assurance that benign requests will propagate as well. More importantly, the filtering of suspected requests will be performed close to the potential source of misbehavior, and benign nodes farther away from the adversary will not be affected, as they will have to relay fabricated queries at a lower rate.

Route reply. Destination D validates the received RREQ packet, by first verifying that it has originated from a node with which it has a SA. Then, Q_{seq} is compared to Q_{max}, the maximum query sequence number received from S, within the lifetime of the SA. If $Q_{seq} \leq Q_{max}$, the request is discarded as outdated or replayed. Otherwise, D calculates the keyed hash of the request fields. If the output matches the SRP header MAU, the integrity of this request is verified, along with the authenticity of its origin.

Node D generates a number of replies to valid requests, at most as many as the number of its neighbors, to disallow a possibly malicious neighbor from controlling multiple replies. For each valid request, D places the accumulated route in the RREP packet and the Q_{ID} and Q_{seq} of the RREQ in the corresponding SRP header fields, so that S can verify the timeliness of the reply. The MAU covers the underlying protocol RREP and the rest of the SRP header, protects the integrity of the reply on its way to the source and offers evidence to S that the request has indeed reached the destination.

An alternate, more efficient implementation would be for D to source-route a reply with an empty payload. The SRP header *Type* indicates that the packet is a reply, the source-route of the datagram contains the desired route in reverse, and the MAU covers the IP source-route, as created by D. If the reply is validated, S extracts the node sequence from the reply IP source-route and reverses it, in order to create the $S \rightarrow D$ route, or, decomposes it into its constituent links.

Route reply validation. On reception of a RREP, S checks the source and destination addresses, Q_{ID} and Q_{seq} and discards the RREP if it does not correspond to the currently pending query. Otherwise, it compares the reply IP source-route with the reverse of the route carried in the reply payload. If the two routes match, S calculates the MAU using the replied route, the SRP header fields, and $K_{S,D}$. Upon successful verification, S is assured that its RREQ reached D and that the reply was not corrupted on its way from D to S. Moreover, since the reply packet has been routed and successfully received over the reverse of the route it carries, the route information has not been compromised during the request propagation; i.e., before arriving at D. Thus, the connectivity information is authentic.

If the alternate form of reply with empty payload is returned, it is sufficient to validate the MAU, since the IP source-route provides the reversed route itself and implies that the reply arrived over this route. Moreover, if an intermediate node V having an SA with S provides a reply, the route suffix is accepted as authentic. That is, V is trusted to provide a correct $V \rightarrow D$ route, and the aforementioned checks are performed for the $S \rightarrow V$ route segment. If this is proven to be authentic, then the entire route is deemed authentic.

Intermediate Node Replies. The caching of overheard routes is a severe vulnerability, since false topology information can be easily disseminated throughout a large portion of the MANET. A malicious node can fabricate data packets or RREPs, which are, for example, cached by nodes operating in a promiscuous mode. When such routes are used or provided as replies, more unsuspecting nodes cache such invalid routes and may use them in the future.

In order to achieve the desired robustness, route caching is not encouraged in general and intermediate nodes are not required to provide route replies. However, route caching can improve the effectiveness of the route-discovery process. If an intermediate node V has an active route to D and an SA exists between S and V, then, a reply could be provided to S. This is the only case in which the RREQ does not actually reach D. This extension of the SRP functionality is enabled by the so-called Intermediate Node Reply Token (INRT). Two alternate designs are proposed.[1] Let K_G be a group key, i.e., a secret shared by the members of a group of nodes to which S belongs. At the same time, S and D, as every pair of the group nodes, have an established SA, as previously discussed. Then, INRT is merely the keyed hash of the RREQ message, calculated as before, but now the key is K_G, instead of $K_{S,D}$. Any group node, namely V, that has an active route to D validates the request based on INRT and generates the reply, as described earlier, using $K_{S,V}$. Alternately, instead of extending the header, S could simply use K_G for the MAU calculation, which is a solution only if D belonged to the group as well. A different method would be to calculate INRT as a digital signature; i.e., the hash of the RREQ encrypted with the private key of S. Then, any receiving node can validate the request and provide the reply. This mode is useful in case a node does not belong to a group but still is securely associated with nodes other than D.

Route maintenance. Route maintenance, though not directly related to route discovery, is an integral part of most MANET protocols. Topological changes have to be detected and the sources of the affected routes have to be notified, avoiding false or fabricated notifications. This task is facilitated by the fact that intermediate-node caching is disabled, but *route error* messages must be retained even if Secure Message Transmission (SMT) is used in conjunction with SRP. The SMT acknowledgments allow for enhanced detection of any type of transmission failure. However, this end-to-end approach does not allow distinguishing benign (due to topological changes) from malicious route failures.

Thus, route error messages generated by intermediate nodes are retained in SRP, in order to provide fast detection of path breaks. The route error packets are source-routed along the prefix of the route reported as broken; S compares the route traversed by the error message to the prefix of the corresponding route. In this way, it can verify that the provided route error feedback refers to the actual route and is not generated by a node not part of the route. The accuracy of the feedback, i.e., whether it reports an actual failure to forward a packet, cannot be verified.

For example, in Figure 1, if the route $\{S, 1, 4, D\}$ had been selected, M_2 could simply generate a route error reporting the $(4,D)$ link broken, even though the route was intact. In order to get the error message to S, M_2 has to source-route it to S, and it does so over $\{M_2, 4, 1, S\}$. Even though node 4 may not discard such a message, S will compare the source-route of the error message to the route reported as broken, or, specifically, the reverse segment reaching the broken link. The comparison fails, and the feedback is discarded, since S infers that an *outlying* node generated the route.

A malicious node lying on an $S \rightarrow D$ route can at most invalidate the route, mislead S by corrupting error messages generated by another node or masking a dropped packet as a link failure. Consequently, a malicious node can harm only the route to which it belongs, which is the extent of the damage if the node simply dropped or corrupted data packets. On the other hand, it is important that under normal conditions the responsiveness of the protocol remains high.

To summarize the SRP for use in the AIS development, the protocol is abstracted as the exchange of two messages, a RREQ and a RREP. The messages are transmitted over a public channel; i.e., a sequence of intermediate nodes that may cause impairment. The idealized form (i.e., the protocol with omission of the parts of the messages that do not contribute to the participants' beliefs) is depicted in Figure 2. $Q_{S,D}$ is the route request and H is the Message Authentication Code (MAU) function. The relevant fields of $Q_{S,D}$ are the sequence number Q_{seq}, and the source and destination node addresses. As for the RREP, denoted as $R_{S,D}$, the Q_{seq} field binds $R_{S,D}$ to the corresponding $Q_{S,D}$, and route is the actual route along which D returns the reply.

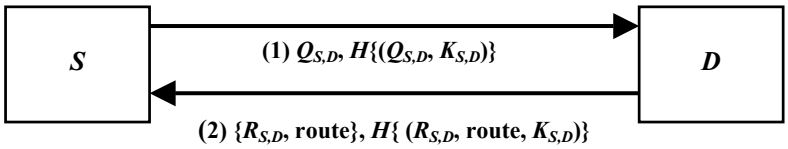

Figure 2. Idealized SRP viewed as an exchange of two messages, without unnecessary fields.

4. IMMUNITY-BASED TECHNIQUES FOR INTRUSION DETECTION

4.1 Background

Security in computer networks, it has been argued, can be considered analogous to immunity in natural systems.[19] Threats and intrusions that compromise privacy, integrity, and availability may arise due to the malfunction at network interfaces and intrusive activities, both internal and external. The idea of using immunological principles in fixed computer network security has progressed since 1994.[20, 21, 22, 23] Forrest, et. al., at the University of New Mexico have worked toward a long-term goal of constructing an AIS for computers. In this approach, the problem of protecting such systems from harmful viruses is viewed as an instance of the more general problem of distinguishing self (authenticated users, uncorrupted data, etc.) from others (unauthorized users, viruses and other malicious agents). This method, called the *negative-selection algorithm*, was used to detect changes in protected data and program files. Kephart suggested another immunological approach using decoy programs for virus detection. In this approach, known viruses were detected by their computer-code signatures and unknown viruses by their unusual behavior within the computer system.[23]

In this paper, the AIS approach is applied to detect intrusive nodes that seek to disrupt MANET route discovery and maintenance. The detection features are formed as statistics of the parameters, $Q_{S,D}$, Q_{seq}, and $R_{S,D}$, that the SRP appends to the IP header in the underlying routing protocol and their error and discard rates in the route discovery and maintenance process.

In contrast with the AIS approach, a positive characterization (PC) method is first presented that uses only "self" data to construct a "normal" profile of MANET routing when SRP is in effect. It is applied here to perform anomaly detection of network intrusions, but is a general single-class approach that can be applied to diverse anomaly detection problems. A negative characterization (NC) method, on which the AIS approach is based, is subsequently presented that improves upon the efficiency of the PC method. The NC technique is motivated by published AIS concepts and attempts to extend Forrest's self-nonself, two-class approach to a multiple-class problem.[24, 25] Specifically, the nonself (abnormal) space is further partitioned into multiple subclasses to determine the level of abnormality.

Evaluation of the AIS approach uses simulated scenarios of intrusive and malicious activities in a MANET with the SRP in effect. Feature samples are processed from simulated outputs to generate the training and test data for both detection approaches.

4.2 Anomaly detection: definitions and terminology

The purpose of anomaly detection is to identify which states of a system are normal and which are abnormal. The states can be represented as a s et of features. The following definitions and terminology follow the discussion in [19].

Definition 1. *System State Space*: A state of the system is represented by a vector of features $\vec{x} = (x_1^i, \ldots, x_n^i) \in [0.0,1.0]^n$. The space of states is denoted by the set $S \subseteq [0.0, 1.0]^n$. It includes feature vectors corresponding to all possible states of the system. The features can represent current and past values of system variables. Actual values can be scaled or normalized to fall in the defined range [0.0, 0.1].

Definition 2. *Normal Subspace* (*Hard Characterization*): A set of feature vectors $Self \subseteq S$ represents the normal states of the system. The complement to the *Self* subspace is referred to as *Nonself* and defined as $Nonself = S - Self$. It is useful to define the self and nonself sets using the corresponding characteristic function $\chi_{self} : [0.0,1.0]^n \rightarrow \{0,1\}$

$$\chi_{self} = \begin{cases} 1, \text{if } \vec{x} \in Self \\ 0, \text{if } \vec{x} \in Nonself \end{cases}.$$

The terms self and nonself are motivated by the natural immune system. In general, there is no sharp distinction between normal and abnormal states. Instead, there is a degree or threshold of normal behavior and, likewise, of abnormal behavior. This implicit "soft" membership decision is formalized in the next definition.

Definition 3. *Normal Subspace* (*Soft Characterization*): The characteristic function of the normal (or abnormal) subspace is extended to take any value within the interval [0.0, 1.0] is given as $\mu_{self} : [0.0,1.0]^n \rightarrow [0.0,1.0]$. In this case,

the value represents the degree or level of normal behavior: a value of 1 indicates normal, a value of 0 indicates abnormal, and intermediate values represent elements with some degree of abnormality. This definition is basically a fuzzy-set specification, with μ_{self} a membership function. However, the fuzzy-set approach will not be developed here.

The soft characterization permits a more flexible distinction between normality and abnormality. In a real system, it may be necessary to decide when to issue an alert. In such situations, anomaly detection becomes again a binary decision problem. The soft characterization can be transformed to a hard one by introducing a threshold parameter θ

$$\mu_{self,\theta}(\vec{x}) = \begin{cases} 1, \text{ if } \mu_{self}(\vec{x}) > \theta \\ 0, \text{ if } \mu_{self}(\vec{x}) \le \theta \end{cases}.$$

Definition 4. *Anomaly Detection Problem*: Given a set of normal samples *Self'* $\subseteq$ *Self*, construct a close estimate of the normal space characteristic function χ_{self} (or μ_{self} in the non-hard case). This function should be able to determine whether the observed system state is anomalous, i.e., indicates a intrusive threat to the system.

In the next two subsections, the PC approach using the nearest-neighbor distance metric and a NC approach are developed, respectively.

4.3 Positive characterization approach

In the PC approach, normal samples are used to construct a characterization of the *Self* space. A model for the *Self* set is not assumed. A positive sample set is instead used as a representation of the *Self* space. The degree of abnormality of a test sample is computed as the distance from the sample to the nearest neighbor in the *Self* set. Toward this end, the definition of the characteristic function of the *Nonself* set is given as:

$$\mu_{nonself}(\vec{x}) = \mathscr{D}(\vec{x}, Self) = \min\{d(\vec{x}, \vec{s}) : \vec{s} \in Self\},$$

where $d(\vec{x}, \vec{s})$ is a Euclidean distance metric or any Minkowski metric. $\mathscr{D}(\vec{x}, Self)$ is the nearest-neighbor distance, i.e., the distance from $\vec{x}$ to the closest sample point in *Self*. Then, the closer sample $\vec{x}$ is to the *Self* set, the closer the value of $\mu_{nonself}(\vec{x})$ is to zero. The "hard" version of the characteristic function is given as:

$$\begin{aligned} \mu_{nonself,\theta}(\vec{x}) &= \begin{cases} 1, \text{ if } \mu_{self}(\vec{x}) > \theta \\ 0, \text{ if } \mu_{self}(\vec{x}) \le \theta \end{cases}, \\ &= \begin{cases} 1, \text{ if } \mathscr{D}(\vec{x}, Self) > \theta \\ 0, \text{if } \mathscr{D}(\vec{x}, Self) \le \theta \end{cases}. \end{aligned}$$

In a dynamic environment like a MANET, the values that characterize normal system behavior may vary within a certain range over a period of time, such as a time-out, a time-stamp on packet transmissions, or Q_{max}, the lifetime of an SA. The term $1-\theta$ represents the level of variability in *Self*, i.e., the maximum distance that a point can be from the set of *Self* samples to be classified as normal.

The PC approach has an efficient implementation using spatial trees.[19] A KD-tree represents a set of *k*-dimensional points and is a generalization of the standard one-dimensional binary search tree. A KD-tree [26, 27, 28] has been used to implement the PC approach. The nodes of a KD-tree are divided into two classes: internal nodes that partition the space with a cut plane defined by a value in one of the *k* dimensions and external nodes, called leaves, that define volumes resulting in hyper rectangles where the points are stored. The KD-tree representation allows answering queries in an efficient manner. The amortized cost of a nearest-neighbor query is $O(\log N)$ [27], where N is the number of nodes. A library that implements the KD-tree structure, developed at the University of Maryland [29], was used in [19] to construct a PC approach using actual intrusion data from computer network attacks obtained from Lincoln Laboratory of the Massachusetts Institute of Technology.[30]

5. PERFORMANCE EVALUATION OF THE INTRUSION DETECTION ALGORITHMS

Unfortunately, extensive data does not exist from actual intrusion attacks and security breaches along with comparatively normal operational data from actual MANETs, as were available to Dasgupta and González.[19] Instead,

data sets used here are generated in simulations of the eight MANET intrusion scenarios described in [1], using the SRP modifying the underlying DSR protocol. Simulated scenarios were developed in the ns-2 environment, developed at University of California, Berkeley, with wireless extensions provided by the Monarch Project at Carnegie-Mellon University.[31, 32] The simulated data is post-processed into training and test samples from both normal and compromised route discovery and maintenance operations. Sample vectors are derived from parameter values extracted from the SRP header packets that have been added to the underlying protocol packets to protect the route processes against malicious nodes. While traffic data was not emphasized in the simulation scenarios, it could be considered in a combined data and routing security scheme, e.g., SMT with SRP. Parametric data from simulated scenarios without intrusion attacks are used to derive training samples, while data from scenarios with attack events are used to form test samples.

Processing the time-phased data for the parameters introduced into the IP header by SRP yields the features used in the intrusion detection algorithms: F_1, the rate of the number of discarded Q_{seq} to the number of Q_{seq} sent from the source S to destination D; F_2, the $Q_{S,D}$ error rate; F_3, the $Q_{S,D}$ discard rate; F_4, the number of discarded route feedbacks; F_5, the $R_{S,D}$ error rate; and F_6, the $R_{S,D}$ discard rate. These six features are derived from parametric data that is sampled during each simulation run; the values of each feature are normalized in value to lie in the range [0,1].

The set $\mathcal{F}$ of normalized features is generated from a series $\vec{r} = \{r_1, r_2, \ldots, r_n\}$ in an overlapping sliding window pattern

$$\vec{r}_w = \{(r_1,\ldots,r_w),(r_2,\ldots,r_{w+1}),\ldots,(r_{n-w+1},\ldots,r_n)\}$$

where w is the window size. In general, from a time series with n points, a set of $(n - w + 1)$-dimensional features can be generated. In some cases, it is appropriate to use more than one time series to generate the feature vectors; shorter feature vectors are placed sequentially to produce the final feature vector. For example, from the six time series F_1 - F_6 and a window size of 3, a set of 18-dimensional feature vectors is generated.

5.1 Evaluation of the PC approach

In the evaluation of the PC approach, the training set of sample vectors is used to construct a KD-tree to represent the *Self* set. The nearest-neighbor distance from each sample vector in the test set to the *Self* set determines the extent of the sample's "abnormality." In the set of experiments, sample vectors are constructed using only one feature (time series) at a time. Histograms of the training and test sample sets corresponding to each feature are generated from normalized feature values grouped and plotted by simulated time-of-occurrence. The plots allow visualization of correlations between feature values and intrusion events. Histograms of the values for each feature of the *Nonself* characteristic function $\mu_{nonself}(\vec{x})$, i.e., the distance from the test set to the training set, are created. Window sizes used to construct the features are 1 and 3, respectively. The Euclidean metric is always used in the *Nonself* characteristic function, while the distance metric $D_\infty(\vec{x},\vec{y}) = \max\left(|x_1 - x_2|,\ldots,|x_n - y_n|\right)$ is used selectively to compare the effect of the metric on the sensitivity of the feature to intrusion events.

Plots of the *Nonself* characteristic function indicate some local maxima that correspond to significant deviations from the normal. It can be verified that these peaks coincide with some of the eight ns-2 simulated scenarios of intrusion attacks by malicious nodes in the test data. Examination of the results leads to the following observations.

1) A single feature is not sufficient to detect the number of intrusion attacks. When used individually in $\mu_{nonself}(\vec{x})$ to detect deviations that correspond to attacks, none of the six parameters can detect all simulated attacks in the test data.

2) A larger window size increases the sensitivity of the *Nonself* characteristic function, seen in higher values of deviation from *Self*.

3) A larger window size allows for the detection of temporal patterns. For the time series T_1 and T_3, increasing the window size does not change the number of detected anomalies. However, for the time series T_2, when the window size is increased from 1 to 3, one additional deviation corresponding to a security attacks is detected. This deviation was not the result of a value of this feature, F_2, the $Q_{S,D}$ error rate, since it would be detected by window size 1. There was a temporal pattern that was not observed in the training data set and may be reported as an anomaly.

4) Changing the distance metric from Euclidean to D_∞ does not alter the number and type of deviations detected.

Therefore, in order to detect intrusion attacks embedded in the test data, more than one feature must be used in the sample vector. In the experiments, all six features were used to construct the sample vector and test the intrusion-detection ability of the PC approach. Two additional tests were performed in which the size of the sliding window was varied.

1) *Window size 1:* Feature vector structure $\left[F_{1,j}, F_{2,j}, F_{3,j}, F_{4,j}, F_{5,j}, F_{6,j}\right]$,where $F_{i,j}$ is taken from the time series T_i.
2) *Window size 3:* Feature vector structure $\left[F_{1,n}, F_{1,n+1}, F_{1,n+2}, F_{2,n}, F_{2,n+1}, F_{2,n+2}, F_{3,n}, F_{3,n+1}, F_{3,n+2}, F_{4,n}, F_{4,n+1}, F_{4,n+2}, F_{5,n}, F_{5,n+1}, F_{5,n+2}, F_{6,n}, F_{6,n+1}, F_{6,n+2}\right]$ where $F_{i,j}$ is taken from the time series T_i.

The *Nonself* characteristic function, for feature vectors formed from the samples of all six time series, is evaluated over the test set. The results reveal several anomalies that correspond to the eight simulated intrusion attacks. Again, an increase in window size increases the sensitivity of the anomaly detection function. However, increased sensitivity raises the likelihood of more false positive detections. To measure the detection accuracy of these functions, they are transformed into their corresponding hard-decision versions, where decision outputs are either normal or abnormal. The decisions can be correlated to the occurrence of the simulated attacks to calculate how many anomalies in the test data caused by an attack were accurately detected.

From Definition 3, the hard version of $\mu_{nonself}(\vec{x})$ is generated by specifying a threshold θ. This threshold indicates the boundary in feature space between normal and abnormal and affects the accuracy of the detection system. A larger value of θ allows an increase in the region considered "normal" or *Self*, thereby increasing the number of false negatives. Conversely, a smaller value of θ restricts the "normal" region, increasing the number of detections, but also the number of false positives, i.e., false alarms. In order to understand the tradeoff between false alarm rate and detection rate, receiver operating characteristics (ROCs) diagrams are examined.[33] The $\mu_{nonself,\theta}(\vec{x})$ is tested with different values of θ, then detection and false alarm rates are calculated. These tests generate a set of points that forms the ROC diagram.

An examination of the ROC diagrams for the $\mu_{nonself}(\vec{x})$ functions reveals, in general, the behavior of the functions to be very similar, namely, high detection rates with small false alarm rates. The anomaly detection functions that use window size 3 show somewhat improved detection rates. This can be attributed to the increased sensitivity, produced by a longer window, to temporal patterns in the MANET data. It is conjectured that after a security breach in a MANET, the systemic reaction to the disturbance propagates throughout the network over some time period and only a characteristic function based on a larger window size, is able to detect it.

The PC approach can work adequately in the simulated test scenarios for the security-aware MANET. The primary limitation of this approach is the large memory requirement, since it is necessary to store all sample vectors that comprise the "normal" profile. The amount of data generated by MANET traffic dynamics can be very extensive, making implementation of the PC approach impractical.

5.2 Negative characterization approach

5.2.1 Background

The implementation difficulties of the PC approach motivate introduction of the negative characterization (NC) approach. As will be seen, the NC approach compresses the information of the normal profile without a significant loss of detection accuracy.

The main activity of the immune system is to distinguish between self, that is, all cells or molecules in the body, and nonself, that is, foreign cells. The nonself elements are further categorized in order to determine the specific response for protection and recovery from different diseases. Specifically, the self-nonself discrimination is achieved in part by T cells, which have receptors on their surface that can detect foreign proteins, i.e., antigens. During the generation of T cells, receptors are made by a pseudo-random, genetic-rearrangement process.[24, 34, 35] Then, the receptors undergo a censoring process, called negative selection, in the thymus, where T cells that react against self-proteins are destroyed, so only those that do not bind to self-proteins are allowed to leave the thymus. These matured T cells then circulate throughout the body to perform immunological functions to protect against foreign antigens. Rather than relying on a central controller, the immune system, however, uses a distributed detection and response mechanism for their survival

from foreign invaders. The nature of this detection and response is ideally suited to the distributed organization and control of a MANET.

Forrest, et. al., [25] developed a negative-selection algorithm based on the principles of self-nonself discrimination found in the immune system. The negative-selection algorithm is summarized by the following steps.

1. Define *Self* as a collection of strings of length *l* over a finite alphabet. This collection needs to be monitored. For example, *Self* may be a normal pattern of activity that is segmented into equal-sized substrings.
2. Generate a set *R* of detectors, each of which fails to match any string in *Self*.
3. Monitor *Self* for changes by continually matching the detectors in *R* against *Self*. If any detector ever matches, then a change has occurred, as the detectors are designed to not match any of the original strings in *Self*.

Many versions of the algorithm have been designed to generate efficient detectors. The algorithms typically use a binary representation and run in linear time with the size of *Self*, depending on the choice of matching rule.[22, 24, 25, 34, 35]

5.2.2 Structure of the NC approach

The applications of negative selection used for MANET security employ real-valued representation to characterize the self-nonself feature space and evolve a set of detectors that are able to cover closely the complementary subspace occupied by nonself samples. The structure of these detector rules is a sequence of conditional statements, namely,

$$\begin{aligned} R^1 &: \text{ If } \text{Cond}_1, \text{ then nonself} \\ &\vdots \quad \vdots \quad \vdots \\ R^m &: \text{ If } \text{Cond}_m, \text{ then nonself} \end{aligned}$$

where

$\text{Cond}_i \equiv F_1 \in [\text{low}_1^i, \text{high}_1^i]$ and $F_2 \in [\text{low}_2^i, \text{high}_2^i]$ and … and $F_n \in [\text{low}_n^i, \text{high}_n^i]$;
$(F_1, \ldots, F_n)$ is a feature vector; and
$[\text{low}_n^i, \text{high}_n^i]$ is the interval bounded by the lower and upper values for the feature F_i in the condition expression of the rule R^j.

The conditional expression of each rule defines a hypercube in the normalized feature space ($[0.0, 1.0]^n$). Then, the set of such rules tries to cover the *Nonself* space with hypercubes. For $n = 2$, the condition of a rule represents a rectangle.

The nonself characteristic function in the hard-decision form generated by a set of rules $R = \{R^1, \ldots, R^m\}$ is:

$$\chi_{nonself,R}(\vec{F}) = \begin{cases} 1, & \text{if } \exists R^j \in R \text{ such that } \vec{F} \in R^j \\ 0, & \text{otherwise.} \end{cases}$$

A genetic algorithm (GA) is used to generate rules for the covering of the *Nonself* space.[36] The set of rules defines the complement of the normal values of the feature vectors. A rule is considered to be effective, provided it does not cover normal samples and the volume it defines in feature space is large. These criteria direct the evolution of the GA.

As discussed earlier, soft-decision functions offer an effective characterization of the *Nonself* space. In this way, the *Nonself* space is further partitioned by different levels of deviation from normal or *Self*. With two features, the levels of abnormality define concentric regions around the *Self* domain in the plane.

Different levels of abnormality are distinguished using a variability parameter, ν, where ν represents the level of deviation permitted in the *Self* space. As ν increases, the *Self* space grows; conversely, a decrease in ν reduces the *Self* space. Parameter ν can be viewed as the radius of a hypersphere containing the feature samples corresponding to *Self*.

In *Nonself* space, a GA with different values of ν generates a set of rules that are able to completely cover this space. A set of such rules takes the form

$$
\begin{aligned}
R^1 &: \quad \text{If} \quad \text{Cond}_1, \text{ then Level 1} \\
&\vdots \qquad\quad \vdots \qquad\quad \vdots \\
R^i &: \quad \text{If} \quad \text{Cond}_i, \text{ then Level 1} \\
R^{i+1} &: \quad \text{If} \quad \text{Cond}_{i+1}, \text{ then Level 2} \\
&\vdots \qquad\quad \vdots \qquad\quad \vdots \\
R^j &: \quad \text{If} \quad \text{Cond}_j, \text{ then Level 2} \\
&\vdots \qquad\quad \vdots \qquad\quad \vdots
\end{aligned}
$$

The levels of deviation are ordered sequentially so that Level 1 contains Level 2, Level 2 contains Level 3, and so on. This ordering implies that a describing expression can be matched by more than one rule, but that the highest level matched will be assigned to the sample. A set of rules, *R*, as described above, generates a soft characteristic function for the *Nonself* space of the form

$$\mu_{nonself}(\vec{F}) = \max\left(\left\{l \,\middle|\, \exists R^j \in R, \vec{F} \in R^j \text{ and } l = \text{ level}(R^j)\right\} \cup \{0\}\right)$$

where $\text{level}(R^j)$ represents the deviation level reported by the rule R^j.

5.2.3 Genetic algorithms for detection rule generation

The goal of the GA is to determine so-called "good" detection rules that cover the *Nonself* space.[37, 38, 39] The "goodness" of a rule is determined by the number of normal samples that it covers, its volume in feature space, and its overlap with other rules. Rule selection is configured as a multi-objective, multi-modal optimization problem. That is, the objective is not one solution or rule, but a set of solutions, i.e., detection rules that jointly solve the problem of covering the *Nonself* region.

A niching technique is used with the GAs to generate different rules.[36] The input to the GA is a set of feature vectors $\mathcal{F} = \{\vec{F}^1, \ldots, \vec{F}^m\}$, that represent the normal behavior of the MANET. Each member $\vec{F}^j$ of $\mathcal{F}$ is an *n*-dimensional feature vector $\vec{F}^j = \left(F_1^j, \ldots, F_n^j\right)$. The algorithm for rule generation uses the following terms: *S*, the set of normal descriptors training data; ν, the level of variability; *maxRules*, the maximum number of detection rules in the solution set; *minFitness*, the minimum fitness level for a rule to be included in the solution set; *maxAttempts*, the maximum number of attempts to evolve a rule with a fitness ≥ *minFitness*. The following pseudo-code segment describes the rule generation algorithm.

```
initiate the ruleSet ← φ, the empty set
        numAttempts ← 0
1 if ( |ruleset| < maxRules) and (numAttempts < maxAttempt)
     then
     Run GA(S, ν)
     R ← best evolved rule
     if Fitness(R) > minFitness
        then
        ruleSet ← ruleSet ∪ {R}
        numAttempts ← 0
        return to 1
      end if
     numAttempts ← numAttempts + 1
  end if
  end
 return (ruleSet)
```

Figure 3. Rule generation algorithm

Each individual (chromosome) in the GA represents the conditional expression of a rule, since the consequence is identical for all the rules, namely, the sample belongs to the *Nonself* space. Levels of deviation in *Nonself* space are

represented by the parameter ν. Each element of the chromosome is represented by fixed binary representation of b bits. Decoding the chromosome is a mapping of the binary representation to the interval [0.0, 1.0].

Evaluation of the GA algorithm uses the following concepts.

1) *Fitness Evaluation:* Given a rule R with conditional expression, $F_1 \in [\text{low}_1, \text{high}_1]$ and … and $F_n \in [\text{low}_n, \text{high}_n]$, a feature vector $\vec{F}^j = \left(F_1^j, \ldots, F_n^j\right)$ satisfies the rule R if the hypersphere with center $\vec{F}^j$ and radius ν intercepts the hyper-rectangle defined by the points (low_1, …, low_n) and (high_1, …, high_n). Raw fitness of a rule is calculated using the following two variables:

a) the number of elements in the training set S that belongs to the subspace defined by the rule R,

$$num\_elements(R) = \left\{\vec{F}^i \in S \middle| \vec{F}^i \in R\right\}$$

b) the volume of hyper-rectangle represented by the rule R, $volume(R) = \prod_{i=1}^{n} (\text{high}_i - \text{low}_i)$.

The raw fitness function of the rule R is given by $raw\_fitness(R) = volume(R) - C \cdot num\_elements(R)$, where C is the coefficient of sensitivity. This coefficient specifies the penalty imposed on a rule if it covers normal samples, a so-called false-alarm penalty. The larger C is, the higher the penalty. Clearly, this function takes positive or negative real values.

2) *Sequential Niching Algorithm:* The GA is iterated multiple times to generate different rules in order to cover the entire *Nonself* region. In each GA iteration, new rules are generated that cover the remainder of the *Nonself* region. The raw fitness of each rule is modified according to the overlap with the previously selected rules. The following pseudo-code routine calculates the final fitness of the rule R.

```
fitness_R ← raw_fitness_R
for each R^i ∈ ruleSet do
    fitness_R ← raw_fitness_R – volume(R ∩ R^i)
end-for
```

where volume(X) calculates the volume of the set X.

6. COMPARATIVE PERFORMANCE RESULTS

In order to evaluate the intrusion detection performance of the NC approach, tests are performed using sets of simulated scenario data used earlier to evaluate the PC approach. The detection features F_1 - F_6 are again used. As a training set, the time series $S1$-$S6$ are used, while the time series $T1$-$T6$ are used as the test set, with window sizes of 1, 3 and 5, alternately.

The parameters for the GA follow. Population size was set at 100; number of generations 1000; mutation rate 0.2, crossover rate 0.8, and the coefficient of sensitivity 0.5. The latter setting represents moderate sensitivity.

The GA was performed with deviation ν set to 0.05, 0.1, 0.2, and 0.4, respectively. Next, the test sample vectors were classified according to the rules generated for each value of ν. This process was repeated 25 times and the results averaged over the runs. Table 1 shows the number of rules generated by the GA for each deviation level and window size, averaged over 25 runs. The table indicates dramatic changes in the number of rules, as the window size increases, i.e., as the dimension of the pattern space grows.

Table 1. Effect of deviation level and window size on number of detection rules

Level	Radius (Deviation)	Avg. Number of Rules (Window Size = 1)	Avg. Number of Rules (Window Size = 3)	Avg. Number of Rules (Window Size = 5)
1	0.05	1.3	23.4	103.4
2	0.1	1.4	28.1	157.8
3	0.2	1.2	34.3	225.4
4	0.4	1.4	45.0	317.9

Detection profiles for the eight simulated MANET intrusion scenarios are created by application of the GA-based rules of the AIS to the test set of samples. With a window size of 1, two of the eight intrusion attacks are detected. A window size of 3 detects four of the eight attacks. A window size of 5 improves the detection results to seven of eight.

While the NC approach of the AIS may lower memory usage and execution time over the PC approach, the PC approach is more accurate. To see this point, for a window size of 1, the positive approach must store 8,192 × 6 = 49,152 floating-point values, while the NC AIS approach need only store 12 × 12 = 144 floating-point values. This yields an approximate compression ratio of 342:1. For a window size of 3, the compression ratio is 8,192 × 18 = 147,456 to 128 × 36 = 4,608, or about 32:1. With a window size of 5, the compression ratio is reduced to approximately 12:1. As noted by Dasgupta and González, the tradeoff is between the compactness of the rule set and detection accuracy.[19] The MANET results confirm this observation. As the value of θ. varies from 0 to 1, computed detection rates for both approaches reveal that the PC method is consistently more accurate than NC AIS. Overall, the NC approach yields detection rates on the same order as the PC method. Table 2 summarizes the highest true positive rates, with a maximum false alarm rate of 1%, of the two approaches.

Table 2. Maximum true-positive detection rates for the detection techniques (maximum 1% false alarm rate)

Detection Technique (Metric)	**Rate for Window Size 1**	**Rate for Window Size 3**	**Rate for Window Size 5**
Positive Characterization (Euclidean)	89.1%	93.2%	98.9%
Positive Characterization (D_∞)	88.4%	91.4%	94.7%
Negative Characterization	80.3%	85.6%	90.8%

Entries in the matrix of Table 3 correspond to the number of test sample vectors in each class. Diagonal entries represent correct classification. The entries indicate that the NC AIS better approximates deviation for the PC method using the D_∞ distance metric. For all test samples, the differences between the deviation levels reported by the PC approach and by the NC were computed. A difference of 0 indicates that the reported levels are the same, a difference of 1 indicates that the results differ by one level, and so on. The results for the two distances and three window sizes are shown in Table 4; differences are expressed as a percentage of nonself sample vectors, i.e., distances > 0.05.

Table 3. Confusion matrix of output levels for PC, with Euclidean and D_∞ metrics, and NC AIS output levels

PC Output Level	**NC Output Level**				
Euclidean	Nil deviation [0.0,0.05]	Level 1 (0.05, 0.1]	Level 2 (0.1, 0.2]	Level 3 (0.2, 0.4]	Level 4 (0.4, …]
[0.0, 0.05]	8061	0	0	0	0
(0.05, 0.1]	5	8	2	0	0
(0.1, 0.2]	0	4.8	3.2	2	0
(0.2, 0.4]	0	28	7	30.4	0.6
(0.4, …]	0	2	7.8	15.3	14.9
D_∞					
[0.0, 0.05]	8060	0	0	0	0
(0.05, 0.1]	5.7	10.5	2.8	0	0
(0.1, 0.2]	0	27.4	6.2	2.4	0
(0.2, 0.4]	0	0	17.3	25.2	3.5
(0.4, …]	0	0	0	8	23

When different metrics are used for the PC approach, the comparative performance changes. Using the D_∞ metric in the PC approach improves the results of the comparison. Although only 48.2% of the outputs from the NC AIS agree with those of the PC algorithm, all of the NC AIS outputs are within zero or one level of the corresponding result reported by the PC. The reason for the improved proximity lies in the construction of the rules of the NC approach. In a Euclidean metric space of n dimensions, the set of points equidistant from a fixed point corresponds to a hypersphere, while, in the D_∞ metric space, this set of points corresponds to a hyperrectangle. Consequently, the rectangular rules of the NC AIS more closely approximate the structure of the D_∞ space, as evinced in the experimental results.

Table 4. Difference between PC and NC Output Levels for Test Data

Delta between PC and NC Output Levels	Euclidean Distance	D_∞ Distance
0	18.7%	48.2%
1	30.8%	51.8%
2	46.3%	0.0%
3	4.2%.	0.0%
4	0.0%	0.0%

7. CONCLUSIONS

The proposed NC method of the AIS yields a good approximation to the level of deviation achieved by the PC approach to intrusion detection within a secure MANET operation. In order to assess the quality of this approximation, a comparison of the NC output levels with the distance results of the PC approach, using the feature set derived from header information of a security-aware protocol for MANETs, is performed. These results are summarized in the form of a confusion matrix in Table 3. For each sample $\vec{F}$ in the test set, the characteristic function $\mu_{\text{nonself}}(\vec{F})$ generated by the NC AIS is evaluated to find the level of deviation; this level is then compared to the distance computed by the PC algorithm. Each row and column of the confusion matrix corresponds to a range or level of deviation, respectively. The ranges are shown in square brackets in the matrix. An ideal output of the NC AIS algorithm generates only values on the diagonal. The natural immune system has the property that system intruders, considered the nonself, are readily detected with few false positives. Detection is performed in two stages. In the first stage, the nonself is identified as new to the invaded system; in the second stage, the immune system then preserves a long-term memory of this identified pattern. By emulating the actions of the immune system, a one-class pattern recognition algorithm can be implemented in the part of the feature space complementary to the set of self features. Design refinements to the algorithm can exchange the number of false positives and negative positives.

An immuno-computing technique, introduced for intrusion detection in computer networks, has been applied to the problem of intrusion attacks on MANET routing.[19] The technique is used to evolve one-class pattern detectors in the feature space complementary to normal MANET route discovery and maintenance processes to identify any changes in normal behavior as observed in the patterns of the security-aware protocol features. The NC technique is used to distinguish and identify the malicious activities of intrusive nodes by monitoring key security protocol parameters in simulated MANET scenarios. The detection efficacy of the NC AIS is then compared with a PC approach. A lack of data from actual MANET security attacks forces the use of simulated intrusion scenarios, employed by other researchers in their evaluation of new security-aware routing protocols.

Some preliminary observations can be made together with suggestions for further research.

When the NC and PC detection schemes are compared, the PC appears to have more detection accuracy at the expense of more memory and execution time. Early results show that the NC AIS detection scheme is practical against intrusion attacks on MANET routing. The NC approach can be seen to detect seven of eight simulated security attacks for a detection rate of 87.5% at a maximum false alarm rate of 1%, while using only 8.25% of the memory required by the PC approach.

The best detection results for the NC AIS approach were attained for window size of 5. Increasing the window size may make the AIS algorithm more sensitive to the dynamic abnormal behavior of intrusive nodes in the ever-changing topology of a MANET. As mobility of the nodes increases, it is conjectured that intrusion detection is improved by increasing the window size of the feature vectors. However, increasing the window size causes dramatic growth in the number of rules, and a corresponding decrease in the NC AIS compression advantage. Additional experimentation is necessary to refine the technique, using an expanded feature set, various window sizes, and more data from simulated MANET scenarios in which node mobility can vary, the underlying reactive routing protocol is other than DSR, and both data and routing messages face malicious attacks.[40] Security-aware protocols for MANETs, other than SRP, can also be used to derive new detection features.

The problem of intrusion detection is essentially one-class pattern recognition of the non-target samples. In addition to the NC AIS approach using GA-rule selection, other neuro-computational approaches to generate covering rules for the nonself space are subjects for investigation. Within the AIS scheme, different covering regions of the nonself space, based on hyperspheres or other uniform volumes and on non-stationary distance metrics, are to be explored in the future.

ACKNOWLEDGMENTS

The author wishes to thank FIT for its support and to recognize the Internet Engineering Task Force (IETF) MANET Working Group in establishing and continuing the development of secure routing protocols for ad hoc networks.

REFERENCES

1. P. Papadimitratos and Z. J. Haas, "Secure routing for mobile ad hoc networks," *Proc. SCS Commun. Networks and Distr. Modeling and Simul. Conf. (CNDS 2002)*, San Antonio, TX, pp. 1–13, Jan. 2002.
2. S. Marti, T.J. Giuli, K. Lai, and M. Baker, "Mitigating routing misbehavior in mobile ad hoc networks," *6th Annual ACM/IEEE Int. Conf. Mob. Computing and Networking (MobiCom)*, Boston, MA, pp. 255 – 265, Aug. 2000.
3. L. Buttyan and J.P. Hubaux, "Enforcing service availability in mobile ad hoc WANs," *Proc. 1st IEEE/ACM Workshop Mob. Ad Hoc Networking and Computing (MobiHoc)*, Boston, MA, Aug. 2000.
4. S. Yi, P. Naldurg and R. Kravets, "Security-aware ad-hoc routing for wireless networks," *Tech. Rpt. UIUCDCS-R-2001-2241,* Dept. of Comp. Sci., Univ. of Illinois, Urbana-Champaign, IL, Aug. 2001.
5. P. Papadimitratos, "Secure routing: methods for protecting routing infrastructures – a survey," work in progress.
6. D B. Johnson, D. A. Maltz and Y-C. Hu, "The dynamic source routing protocol for mobile ad hoc networks (DSR)," *Internet Draft, MANET Working Group*, *IETF*, Feb. 2002.
7. Z. J. Haas, M. Perlman and P. Samar, "The interzone routing protocol (IERP) for ad hoc networks," *IETF MANET Working Group*, draft-ietf-manet-zone-ierp-01.txt, Jun 1, 2001.
8. Z. J. Haas and M. Perlman, "The performance of query control schemes of the zone routing protocol," *IEEE/ACM Trans. Networking*, **9**, pp. 427-438, Aug. 2001.
9. C. K. Toh, "Associativity-based routing for ad-hoc mobile networks, " *Wireless Pers. Commun. J.*, **4**, pp. 103-139, Mar. 1997.
10. R. Zucceratto and C. Adams, "Using elliptic curve Diffie-Hellman in the SPKM GSS-API," *Internet Draft*, *IETF*, Aug. 1999.
11. W. Diffie and M.E. Hellman, "New directions in cryptography," *IEEE Trans. Info.Theory*, **22**, pp. 644--654, Nov. 1976.
12. N. Asokan and P. Ginzboorg, "Key agreement in ad hoc networks," *Comp. Commun*., **23**, pp. 1627-1637, Nov. 1, 2000.
13. L. Lamport, R. Shostak and M. Pease, "The Byzantine generals problem," *ACM Trans.v Program. Lang.*, **4**, pp. 382-401, Jul. 1982.
14. IEEE Std. 802.11, "Wireless LAN media access control (MAC) and physical layer (PHY) specifications," http://standards.ieee.org/getieee802/, 1999.
15. H. Krawczyk, M. Bellare and R. Canetti, "HMAC: keyed-hashing for message authentication," *IETF RFC 2104*, Feb. 1997.
16. A. Menezes, P. van Oorschot and S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, Boca Raton, FL, Oct. 1996
17. "Secure hash standard," *FIPS Pub. 180-1*, Comp. Sys. Lab., NIST, Gaithersburg, MD, Apr. 1995.
18. R. Rivest, "The MD5 message-digest algorithm," *IETF RFC 1321*, Apr. 1992.
19. D. Dagupta and F. González. "An immunity-based technique to characterize intrusions in computer networks," *IEEE Trans. Evol. Comput.*, **6**, pp. 281-291, Jun. 2002.
20. D. Dasgupta, ed., *Artificial Immune Systems and Their Applications*, Springer-Verlag, New York, NY, 1999.

21. D. Dasgupta and H. Brian, "Mobile security agents for network traffic analysis," *Proc. DARPA Inform. Surviv. Conf. and Expo. II (DISCEX-11*), Anaheim, CA, Jun. 12-14, 2001.
22. S. A. Hofmeyr and S. Forrest. "Architecture for an artificial immune system," *Evol. Comput*., **8**, pp. 443–473, 2000.
23. J. O. Kephart, "A biologically inspired immune system for computers," *Artificial Life IV, Proc. 4th Int. Workshop Synth. Simul. Living Sys.*, Cambridge, MA, pp. 130–139, Jul. 1994.
24. P. D'haeseleer, S. Forrest and P. Helman, "An immunological approach to change detection: algorithms, analysis, and implications," *Proc. 1996 IEEE Symp. Comp. Security and Privacy*, Oakland, CA, pp. 110–119, May 1996.
25. S. Forrest, A. Perelson, L. Allen and R. Cherukuri, "Self-nonself discrimination in a computer," *Proc. 1994 IEEE Symp. on Resrch in Security and Privacy*, Los Alamitos, CA, May 1994.
26. J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, **18**, pp. 509–517, Sept. 1975.
27. J. L. Bentley, " K-d trees for semidynamic point sets," *Proc. 6th Ann. ACM Symp. Comput. Geom*., pp. 187–197, 1990.
28. J. H. Friedman, J. L. Bentley and R. A. Finkel, "An algorithm for finding best matches in logarithmic expected time," *ACM Trans. Math. Softw*., **3**, pp. 209–226, Sept. 1977.
29. D. Mount and S. Arya, "Ann: a library for approximate nearest neighbor searching," *2nd Annual Fall CGC Workshop on Comput. Geom*., http://www.cs.umd.edu/ mount/~ANN, 1997.
30. M. Zissman,. "DARPA intrusion detection evaluation," MIT Lincoln Laboratory, http://www.ll.mit.edu/IST/ ideval/index.html, 1999.
31. "Network simulator version 2, ns-2," http://www-mash.cs.berkeley.edu/ns/.
32. "CMU Monarch Project, wireless and mobility extensions to ns-2," http://www.monarch.cs.cmu.edu/cmu-ns/html.
33. F. Provost, T. Fawcett and R. Kohavi, "The case against accuracy estimation for comparing induction algorithms" *Proc. 15th Int. Conf. Mach. Learning*, Madison, WI, pp.445-453, Jul.1998.
34. D. Dasgupta and S. Forrest, "Novelty detection in time series data using ideas from immunology," *Proc. Int. Conf. on Intelligent Sys*., pp. 82-87, Jun. 1996.
35. A. Somayaji, S. Hofmeyr, and S. Forrest, "Principles of a computer immune system," *Proc. 2nd New Security Paradigms Workshop*, pp. 75–82, Sept. 1997.
36. D. Beasley, D. R. Bull and R. R. Martin, "A sequential niche technique for multimodal function optimization," *Evol. Comput*., **1**, pp. 101–125, 1993.
37. D. Dagupta and F. González, "Evolving complex fuzzy classifier rules using a linear genetic representation," *Proc. Int. Conf. Genet. and Evol. Comput. (GECCO),* Morgan Kaufman, San Fran., CA, July 2001
38. D. Dagupta and F. González, "An intelligent decision support system for intrusion detection and response," *Info. Assurance in Comp. Networks*, *Lecture Notes in Comp.Sci.*, *Proc. Int. Wrkshp Math. Meth., Models, and Architect. Comp. Netwks Security (MMM-ACNS),* St. Petersburg, Rus., Springer-Verlag, New York, NY, pp. 1–14, May 2001.
39. D. Dasgupta and Z. Michalewicz, ed., *Evolutionary Algorithms in Engineering Applications*, Springer-Verlag, New York, NY, 1997.
40. C.E. Perkins, E.M. Royer, and S.R. Das, "Ad hoc on-demand distance vector routing," *Internet Draft*, *MANET Working Group*, *IETF*, draft-ietf-manet-aodv-12.txt, Nov. 2002.