Pak. J. Engg. & Appl. Sci. Vol. 7, Jul., 2010 (*p.* 71-94)

Energy Efficient Security in MANETs: A Comparison of Cryptographic and Artificial Immune Systems

Nauman Mazhar
National University of Sciences and Technology (NUST)
Sector H-12, Islamabad, Pakistan

Abstract

MANET is characterised by a set of mobile nodes in an inherently insecure environment, having limited battery capacities. Provisioning of energy efficient security in MANETs is, therefore, an open problem for which a number of solutions have been proposed. In this paper, we present an overview and comparison of the MANET security at routing layer by using the cryptographic and Artificial Immune System (AIS) approaches. The BeeAdHoc protocol, which is a Bio-inspired MANET routing protocol based on the foraging principles of honey bee colony, is taken as case study. We carry out an analysis of the three security frameworks that we have proposed earlier for securing BeeAdHoc protocol; one based on asymmetric key encryption, i.e BeeSec, and the other two using the AIS approach, i.e BeeAIS based on self non-self discrimination from adaptive immune system and BeeAIS-DC based on Dendritic Cell (DC) behavior from innate immune system. We extensively evaluate the performance of the three protocols through network simulations in ns-2 and compare with BeeAdHoc, the base protocol, as well as with state-of-the-art MANET routing protocols DSR and AODV. Our results clearly indicate that AIS based systems provide security at much lower cost to energy as compared with the cryptographic systems. Moreover, the use of dendritic cells and danger signals instead of the classical self non-self discrimination allows to detect the non-self antigens with greater accuracy. Based on the results of this investigation, we also propose a composite AIS model for BeeAdHoc security by combining the concepts from both the adaptive and the innate immune systems by modelling the attributes and behavior of the B-cells and DCs.

Key Words: MANET security; cryptography; Artificial Immune Systems (AIS); danger theory

1. Introduction

Recent past has seen widespread acceptance and adoption of Mobile Ad Hoc Networks (MANETs), which has made them an active area of research [1]. A MANET comprises a set of wireless mobile nodes that provide multi-hop connectivity without the need of a centralized controlling entity. All nodes within the transmission range of one another communicate directly using their radios and at the same time relay messages to their immediate neighbors if the destination is beyond their direct transmission range. Since nodes are willing to forward data for other nodes, each node acts as a router. The routers are free to move about randomly and as a result may collectively form any arbitrary topology. This self-configuring ability of MANET nodes to quickly organize into a network and start communicating over wireless links has found applications in diverse areas of civil and military use [2].

Since standardization is still underway, there are a number of proposals under consideration for MANET routing. The candidate routing algorithms can broadly be classified on the basis of the routing scheme followed; either next hop routing or the source routing. In next hop routing scheme, a routing table is consulted to determine the next hop node that can send the packet towards the destination node; each node decides upon the next hop on the basis of its own routing table. While in a source routing protocol, the packet being routed contains the complete route to the destination node within its packet header; the packet follows the specified hop-by-hop connected path. Examples of next hop routing protocols for MANETs include the Destination Sequence Distance Vector protocol, which is a pro-active routing algorithm [3], and Ad-Hoc On-demand Distance Vector protocol (AODV), which provides reactive next hop routing [4]. The source routing algorithms include the Dynamic Source Routing protocol (DSR)[5]. The DSR and AODV are regarded as the leading protocols for MANET routing.

MANETs offer a challenging environment for secure communication among the participating nodes. Wireless, as a broadcast medium is insecure. All nodes within the signal reception range can capture the packets and generate random responses. It is, therefore, trivial for a malicious node to design and mount a routing attack against any node in a MANET. The routing attacks are meant to interfere with the network's normal routing process and thus degrade the protocol routing performance [6].

The inherent insecurity of MANETs, therefore, necessitates development of an effective security solution to secure the MANET routing process. In addition to providing the required level of security, a MANET security solution also needs to consume minimum amount of energy owing to the MANET operation in wireless communication environment, especially in the constrained mobile ad-hoc scenario.

Need for Energy Efficient Security:

MANETs impose certain constraints on the communicating nodes, which broadly include the node mobility, limited available wireless bandwidth, usually limited processing power and short battery life. The node mobility results in a continuously changing network topology. Routing algorithms then need to frequently discover new routes and update the routing tables according to these changes. Limited wireless bandwidth requires the routing algorithms to reduce to the minimum the amount of control traffic generated to build and maintain routing tables and to discover new routes. Nodes should be able to route packets by transmitting the least possible control bytes into the network allowing the use of maximum available bandwidth for data transfer. The limited battery capacity also requires the routing algorithms to minimize the control traffic and to distribute packets on multiple paths. This allows the batteries of different nodes to deplete at equal rates, thus increasing the life time of the network [7] [8].

MANETs, therefore, impose a very fundamental constraint on the security solution being designed for this environment, i.e "*the security solution needs to be highly efficient in energy consumption*", and it translates to the following requirements:

- ***Minimum Control Overhead.*** Control overhead (bytes or packets) of the routing protocol needs to be minimized. Lower the control overhead, higher is the channel bandwidth utilization due to minimizing the transmission of redundant control bytes. Moreover, lower energy would be required to actually transmit and/or receive bits at the network interface card.

- ***Minimum Computational Complexity.*** An algorithm with lower computational load allows the mobile node to achieve longer battery life by consuming less amount of energy for its internal processing.

Energy efficient routing requires optimizing the packet routing process for lower energy consumption. However, none of the classical MANET routing algorithms have been designed keeping in view energy efficiency. The authors in [9] studied DSR and AODV for energy efficiency and report that protocol design parameters, such as lower delay and higher packet delivery ratio, do not achieve low energy consumption.

Security Approaches for MANETs

The popularity of MANETs has lead to an increasing interest in addressing their security issues. There are a number of proposals for secure MANETs that are based on standard cryptography and Artificial Immune Systems (AISs). The cryptographic systems include both the symmetric as well as asymmetric approaches. The AODV protocol has been secured by a public key system (using asymmetric keys) through Secure Ad-Hoc On-demand Distance Vector protocol (SAODV) [11]. While the DSR protocol has a secure version ARIADNE [10], which employs symmetric key cryptography to provide security to the source routing process. However, cryptography includes compute intensive mathematical operations [12], especially the asymmetric systems, and thus imposes heavy computational load on mobile nodes causing rapid battery depletion. Moreover, most of these security systems involve computing digital signatures and/or message hashes and transmitting them along with the data, resulting in transmission of additional control information, which reduces the effective protocol throughput.

Artificial Immune Systems have also been studied extensively for network anomaly detection [13] [14] [15]; authors of [16][17] provide a comprehensive review. An AIS security solution has been presented in [18] that can detect misbehavior in DSR protocol. This protocol as well as other earlier systems generally employed negative selection for discriminating self from non-self. Sometimes clonal proliferation was added to produce a quicker immune system secondary response. The systems, following the principles of classical immunology, had higher false positive rates and were not scalable due to issues such as the need for large detector population, creation of holes in detector space and requirement of learning the system normal behavior at startup. In addition, MANETs pose another problem for an AIS based on negative selection; MANET nodes are mobile, which causes the system self to change frequently during operation. Therefore, the system normal behavior (self) learnt at startup time no longer remains valid. Consequently, during the system operation even a legitimate but changed self causes generation of false alarms. What is needed for correct functioning of the system is an ability to update the system self dynamically. This ability is afforded by the "Danger Theory", which is a new immunological paradigm for AIS development. Danger theory is, therefore, especially suited to applications such as intrusion detection and has potential to provide a security solution for the MANET environment.

Bio-inspired Routing and AIS Security

In addition to the classical MANET routing algorithms, the focus of MANET research community has also been on the application of nature inspired engineering approach to solve the MANET routing problem. Consequently, a number of routing proposals have been presented by the nature inspired community, which include the AntHocNet [19][20], Termite [21][22][23] and BeeAdHoc [24][25][26]. These protocols discover multiple routes to destination nodes and then route data on these paths probabilistically to achieve balanced network traffic, which also has the effect of depleting the node batteries at the same rate.

The increasing use of mobile nodes, with their limited battery power availability, has resulted in making energy efficiency an important parameter for designing MANETs [8][27][28]. An example of a MANET protocol that has specifically been designed for energy efficiency is the BeeAdHoc. BeeAdHoc is a reactive protocol based on the principle of source routing. Its routing behavior is inspired from the foraging of a honey bee colony, discussed in [29][30]. BeeAdHoc has network performance comparable or better than that of DSDV, AODV and DSR protocols, but the protocol is considerably more energy efficient. The algorithm achieves lower energy consumption by generating less number of control packets and using multiple paths to route data packets. This makes BeeAdHoc an ideal candidate for implementing energy efficient security.

The provisioning of energy efficient security in MANETs is still an open issue. We, in this paper, provide an overview of MANET security by taking the nature inspired protocol, BeeAdHoc, as case study. We discuss both the cryptographic and the AIS security approaches. Major contributions to MANET security presented in this paper include:

- Study of the comparative effects of cryptography and AIS based security on MANET routing protocols, with emphasis on comparing the performance of self/non-self based AIS with the danger theory based AIS. We describe and analyze the security frameworks developed for BeeAdHoc; the cryptographic system BeeSec [31] and the AIS based systems, BeeAIS [32] and BeeAIS-DC [33]. BeeAIS implements the classical self/non-self approach to anomaly detection while BeeAIS-DC is based upon the danger theory.

- Performance comparison of our secure MANET protocols with the non-secure DSR and AODV; we determine the extent to which the security extensions degrade the BeeAdHoc performance. Compared to the study in [31][32][33], we introduce additional parameters for a broader comparison and ascertain scalability characteristics of our proposed security frameworks by considering dense networks of upto 150 nodes. Moreover, the BeeAIS performance parameters are now reported in mobility scenario.

- A composite model, CompAIS, for AIS based network anomaly detection is proposed that uses the concepts from innate immune system as well as adaptive immune system. We perform the activation of B-cells through DCs, followed by affinity maturation for a higher detection performance.

The organization of the paper is as follows. Section 2 gives an introduction to the architecture of BeeAdHoc protocol with a systematic analysis of its security vulnerabilities. In Section 3 we briefly describe the Biological Immune System. Section 4 covers the Artificial Immune Systems in sufficient detail to provide the background necessary for understanding the work presented in this paper. Then in Sections 5, 6 and 7 we give the implementation details of our asymmetic cryptography system (BeeSec), the negative selection based BeeAIS, and the danger theory based BeeAIS-DC, respectively. We have implemented the systems to secure BeeAdHoc protocol against routing attacks by malicious nodes. The addition of security should not lower the base protocol's network performance. Therefore, in Section 8, we carry out extensive network performance evaluation of our security frameworks and compare them with the base protocol, BeeAdHoc, as well as with other MANET protocols such as AODV and DSR. The results clearly demonstrate that the performance of security frameworks based on AIS is almost similar to that of BeeAdHoc and comparable to unsecured AODV and DSR. In order to improve the detection performance of AIS security framework, in Section 9, we propose a composite AIS, CompAIS, by combining the concepts from innate and adaptive immune systems. The CompAIS uses direct activation of B-cells by DCs along with affinity maturation to allow closer match with the suspected non-self antigens for more effective detection. In the end is the conclusion of the paper with proposed future work.

2. BeeAdHoc Protocol

In this section we introduce the nature inspired routing protocol, BeeAdHoc that has been designed for MANETs using the organizational principles of the honey bee colony discussed in [29][30]. BeeAdHoc is a reactive, source routing protocol that performs multipath, fault tolerant, robust and efficient routing. The most significant aspect of the protocol is its decentralized control that is essential for the effective operation of an adhoc mobile wireless network. Moreover, the protocol has specifically been designed for energy efficiency, which is another very important design consideration for MANETs.

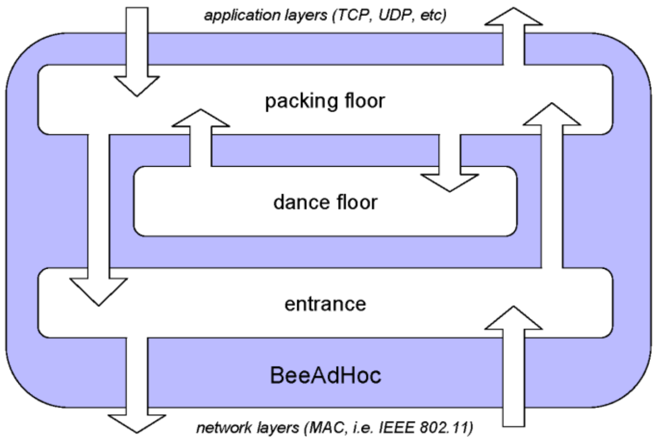

application layers (TCP, UDP, etc)
packing floor
dance floor
entrance
BeeAdHoc
network layers (MAC, i.e. IEEE 802.11)

Fig. 1: Architecture of BeeAdHoc

2.1 Agent Model

BeeAdHoc protocol comprises three different agents to model the packer bees, scout bees and foragers in a hive. The agents are briefly introduced here:

2.1.1 Packers*:* They are meant to store data packets received from the upper layer (transport). Packers in BeeAdHoc mimic the behavior of food storer bees in a hive. They are used to obtain routes for data packets in the form of foragers present in the hive. Data is then passed to these foragers for onward despatch to the destination. After that the relevant packer is removed from the system.

2.1.2 Scouts*:* The scouts are meant to discover new routes from the launching node to the destination node. BeeAdHoc has two scout types: forward scouts and backward scouts. The forward scout is a broadcast packet that is flooded across the network for new route discovery. The discovered route is a source route formed by accumulating the node addresses of visited nodes in scout header. The backward scout is a unicast packet that carries the discovered route back to the source node.

2.1.3 Foragers*:* A forager is the data carrier of BeeAdHoc protocol. Foragers on the source node receive data packets from the packers and carry them to their destination. A forager is a unicast source routed packet that inherits the route either from a scout, in case of a newly discovered route, or from another forager, in case data is being transported over an existing route. The foragers also collect network routing information from the visited nodes, which is then used to evaluate the quality of the traversed path. The quality of a route determines whether the path would be selected for packet routing. The probability of a particular path being selected, compared to other available routes, is increased considerably if the path has a higher quality metric.

2.2 BeeAdHoc Architecture

In BeeAdHoc protocol, every node acts as a hive and implements the Entrance, Packing Floor and Dance Floor of a bee hive. Figure 1 shows the hive structure.

2.2.1 Packing Floor: In order to route data packets, BeeAdHoc operates at the network layer. Packing floor interfaces BeeAdHoc with the higher transport layer (TCP or UDP). Transport layer data received at the packing floor requires a route to the destination before it can be sent. The route may be provided by an existing forager from the dance floor; otherwise the packing floor initiates a new route discovery by broadcasting a forward scout.

2.2.2 Entrance: The entrance provides BeeAdHoc functionality between the Media Access Control and the Network layers. All scouts and foragers entering or leaving the hive pass through the entrance.

2.2.3 Dance Floor: The discovered routes are maintained at the dance floor in the form of foragers, along with the requisite routing information – quality of the path – for routing of data packets. Upon reaching the destination node, a backward scout or a forager advertises the route to other foragers at the dance floor by performing a “dance”. The “dance” is an abstraction of the quality metric of the path; higher the dance number, more number of clones would be generated for that forager, resulting in transporting more data packets over the advertised path. The concept is equivalent to the recruitment of a higher number of forager bees for a more profitable foraging site in nature [30].

2.3 The Operation of BeeAdHoc Protocol

When BeeAdHoc receives user data at the packing floor, it instantiates a packer and appends with it the received data. The dance floor is then searched for a forager with the required route to the destination. In case a route is available, the relevant forager gets the data packet and the packer dies. Otherwise, the packer waits for a returning forager that may be on its way back to the hive from the same destination. If no forager returns within the specified time, the packing floor initiates a new route discovery by broadcasting a forward scout. During scout flooding, a unique key based on the scout ID and its source ⟨*Scout ID, Scout Source Node*⟩ enables other nodes to distinguish a new scout from a copy of an already received scout; a possibility that exists with network packet flooding, in which case the copy is dropped. In order to reduce unnecessary traffic, BeeAdHoc implements controlled flooding, i.e– scouts are broadcast in an expanding ring search, with the time to live (TTL) field set initially to a small value and then incremented gradually for subsequent route discovery broadcasts. This allows to control scout flood radius, and hence minimizes the control overhead traffic.

While flooding across the network, the forward scout accumulates node IDs of visited nodes in its packet header, thus forming the source route. Upon reaching the destination node, the source route is reversed to create a backward scout, which is then sent back to the source node. The backward scout, upon arriving back at source, has the route to the destination which is advertised to other foragers at the dance floor by using the metaphor of “dance”. Consequently, the foragers select one of the routes and use it to transport data to the destination. Since BeeAdHoc allows multiple forward scouts to be received and processed by destination nodes, the protocol allows multiple paths to be discovered and subsequently be used to route the data packets. Once at the destination, a forager stays there until sent back to the source node. A reliable transport protocol at receiver, such as TCP, sends the data acknowledgement message to the sender [34]. BeeAdHoc makes use of TCP acknowledgements to send all foragers accumulated at the receiver back to the sender, where they are again ready to transport new data.

In addition to carrying data, the foragers also collect routing information from the network. BeeAdHoc utilizes different types of foragers – “delay” or “lifetime” foragers – based on the type of routing information collected. “Delay” foragers collect information related to path delay with the aim to preferably route data packets over the minimum delay path. On the other hand “lifetime” foragers monitor the remaining battery capacity of the nodes and try to route data packets to achieve maximum network life time through minimum battery consumption. The routing information collected by foragers determines the dance number of the traversed path, which represents the route quality. The route that has a higher quality of routing metric would have a higher dance number and causes more clone foragers to be generated, and therefore, would have a higher probability of being selected from the dance floor to transport data. We have provided here the brief description and operation of BeeAdHoc protocol that was considered necessary for understanding the work presented in this paper. More details are contained in [25].

2.4 Vulnerability Analysis of BeeAdHoc

MANETs are inherently vulnerable to malicious attacks [6]. In addition, if a routing protocol has vulnerabilities, a malicious node could launch a number of Byzantine [35] attacks to disturb the normal routing process of the protocol. The attack would be successful if a malicious node can either add/remove itself to/from a route or allow/prevent traffic through a particular node, while both cases were not possible under normal routing scenario. In [31], we have analyzed various shortcomings of the BeeAdHoc protocol that allow a malicious node in the network to fabricate various attacks against BeeAdHoc routing. The attacks are being described here for the sake of completeness of this paper.

2.4.1 Attacks Related to Scout: A forward scout, during the route discovery phase, is broadcast into the network. Resultantly, all nodes within close vicinity receive the transmission, even though they are not directly on the path to the destination. If the network has a malicious node, it is able to receive and either modify the route contained in scout header or insert a new and fake source route before re-broadcasting the scout back. Likewise, in the case of a backward scout, the malicious node can easily modify the route in the unicast packet. Moreover, it can spoof the scout source and forge a fake new scout or insert a fake scout ID. When these fake/modified scouts finally return to scout source, they cause establishment of forged routes at the source node.

2.4.2 Attacks Related to Forager Route: The foragers in BeeAdHoc are unicast packets with a pre-set source route. An intermediate node that acts maliciously can alter the source route stored in forager header to cause establishment of any desired route. Also, a forged forager can be generated having a spoofed source address or forged source route, or both, which would result in disrupting the normal routing process. A quicker attack may be launched by the malicious node by sending a forged forager with a desired route towards the source node. The attack effect would be realized faster as wait time for route discovery is reduced in this case.

2.4.3 Attacks Related to Forager Route Information: A malicious node may also launch an attack to illegally change routing parameters carried in forager headers, thus enhancing/degrading the path quality. This increases/ decreases the route dance number and consequently generates more/less replicas of the route. As a result, the probability of sending more data packets on a low quality route would increase, thus wasting the available network resources.

An assumption has been made here for the successful launch of above mentioned attacks; *the malicious node is able to ensure completeness of the forged/modified source* *route for all hops*. If the source route is not connected hop to hop, the forged/modified scout or forager will not reach the destination and the attack would be un-successful. This would, in effect, cause denial-of-service for the source node if foragers or scouts do not return to the source node and no data transfer takes place.

3. Biological Immune System (BIS)

BIS is a defence mechanism for the human body [36] that provides protection against infections by pathogens, including bacteria, virus and parasites. The immune system consists of a large number of cells, both acquired and immune, that interact with each other to detect and eliminate pathogens. Substances that can stimulate a specific response from the immune system are referred to as antigens. The antigens are protein compounds that exist both on the body cells as well as pathogens. They may be the self antigens when they belong to own body cells, or non-self antigens when they belong to pathogens. The immune system has mechanisms to detect and differentiate between the self and non-self antigens and initiate an appropriate immune response to eliminate the pathogens, i.e non-self antigens, from the body. The major components of the immune system include the Innate System and the Adaptive System.

Innate Immune System, also known as non-specific immunity, refers to the defence system against pathogens that an individual is born with. It protects the human body against some viruses, worms and bacteria by using built-in knowledge of these infections. It comprises specialized cells that ingest the pathogens and present fragments of their proteins to other immune system components. In this way, these cells work in coordination with the adaptive immune system to signal damage to self cells and cause activation of the adaptive immune response.

Adaptive Immune System is also called as the acquired or specific immunity. It has the ability to adapt, recognize and destroy previously unknown pathogens. The adaptive immune response is mediated mainly through lymphocytes – B-cells and T-cells. It learns to identify new pathogens and adapts itself to improve the non-self antigen matching performance. The system adaptability also enables to memorize newly encountered pathogens for a faster future detection.

3.1. Immune System Operation

All lymphocytes, B-cells and T-cells, have specific patterns on their surfaces, called antibodies, which are meant to detect pathogens. Antibodies are protein compounds with the ability to bind antigens. Binding of an antigen and an antibody implies their matching and the degree of match is termed as affinity [36].

B-cells are formed inside bone marrow such that one B-cell has only one specific type of antibody. Since a healthy human body creates approximately 1 billion new cells daily, the population of B cells is quite large. This results in the immune system having a large number of different antibodies capable of detecting almost all pathogens that exist in nature. Before the B-cells leave the bone marrow, they undergo negative selection; "*if a B-cell antibody matches a self antigen, the B-cell dies a natural death, with the surviving B-cells being tolerant to self*". However, B-cells in the body do not provide complete self tolerance. This is because during the negative selection of B-cells in bone marrow, all the body self antigens are not available. T-cells are the entities that have better self tolerance. T-cells are also born in bone marrow, as the B-cells, but then migrate to the thymus to maturate. In thymus, T-cells are presented with more complete set of body self antigens during negative selection, which makes the T-cells more self tolerant.

If antibody on a B-cell matches the antigen, and the antibodies of some T-cells also bind to the same antigen to provide costimulation of the B-cell [37][38], the detection of a non-self antigen is verified. B-cell then undergoes clonal proliferation and generates antibodies in large quantities to fight the invaders [39]. This first exposure to a non-self antigen is termed as the primary response. During clonal proliferation, some of the B-cell antibodies match the pathogen better than the original B-cells. Clonal selection then allows selecting the B-cells with antibodies having greater affinity with the non-self antigen, a phenomenon known as affinity maturation. The B-cells that match very closely with the pathogen are transformed to memory cells and upon future encounter with the same non-self antigen, produce a much faster secondary response [40].

The immune system has an additional control, the danger signal [41], that is used to activate the T-cells for co-stimulation of B-cells. After T-cell antibodies bind to an antigen, the T-cell gets activated only if a "danger" signal, generated either by self cells or the innate system, exists in the tissues. The "danger" signal indicates damage caused to self cells due to non-self antigens, e.g, when a cell dies of viral infection (necrocis). Cell debris produced when a cell dies its natural death (apoptosis) is different compared to when it is killed by a pathogen, and indicates the presence of "safe" signal in tissues. The "danger/safe" signal is sensed by the Dendritic cells (DCs) that form part of the innate immunity. The DCs sample the antigens, including both self as well as non-self antigens, from the body tissues and present them in thymus during maturation of T-cells. Depending on the type of signals (danger or safe) present in the tissues, DCs may have any one of the following three states:

Immature DCs. A DC when born is in an immature state. When exposed to the "danger" signals from necrotic cells, a DC transitions from immature to mature state; while "safe" signals generated from natural cell death cause an immature DC to turn into semimature DC. In both these states, DCs have the ability to go from the tissue to thymus for antigen presentation to T-cells.

Semi-Mature DCs. The DCs in semi-mature state present the sampled antigens in thymus to have a tolerogenic effect. T-cells whose antibodies match with any of the antigens from the DCs in semi-mature state get de-activated. Resultantly, these T-cell antibodies can no longer initiate any response from the adaptive immune system.

Mature DCs. A mature DC has an immunogenic effect when presented in thymus to T-cells. If the antibodies of a T-cell match an antigen that is offered by a DC in mature state, the T-cell gets activated. T-cell activation is required to elicit an adaptive immune response through co-stimulating the B-cells against non-self antigens.

4. Artificial Immune System (AIS)

An AIS can be defined as data manipulation, classification, and reasoning methodologies inspired by the BIS and based upon the same metaphors [36]. AISs can be applied to solve a number of real world problems. The basic requirement for a system to be characterized as an AIS is that the system should:

- model an immune system component, such as a cell, molecule, organ, etc.
- perform pattern matching.
- incorporate an immune principle; clonal/negative selection or immune network.

The AISs model different aspects of the BIS and are being applied to a diverse set of real-world applications. The authors in [42] provide a survey of the existing application areas for AISs and also suggest a set of features for such applications.

4.1 Immune Mechanisms and Theories

A number of computational procedures exist to model various immune mechanisms and theories, that are used as building blocks of an AIS. We will briefly describe two of these concepts here that we have applied in developing the AIS based security systems for MANETs. These include the following:

4.1.1 Self/Non-self Discrimination: The self/non-self discrimination employs negative selection algorithm from the Thymus Model [36][43] to recognize patterns that do not belong to a set of known patterns. Given a set of self-peptides, the algorithm tests the binding of T-cell receptors (TCRs) with the self-peptides. If a T-cell recognizes the self-peptide, the T-cell is discarded; Therefore, the resultant T-cells do not recognize the self-cells or molecules and can be used to differentiate self from non-self.

4.1.2 Danger Theory*:* The viewpoint of self/non-self discrimination holds broad acceptance in the field of immunology. It is believed that the adaptive immune response is initiated when some foreign or non-self antigens are identified to exist in the body. This belief is challenged by the Danger Theory [41]. The immunologists that support danger theory claim that recognition of pathogens is not enough to elicit a response from the adaptive immune system; an additional sensing of "danger" is needed before the body reacts to an infection that is caused by pathogens. Danger is indicated by the presence of tissue damage or cell debris caused by cell death during an infection. The presence of "danger" is determined by the DCs from innate immune system. Therefore, the proponents of Danger Theory, in effect, claim that it is the innate immune system that initiates and controls the adaptive immune response. When "danger" in not sensed in body tissues, adaptive immune response is suppressed.

4.2 Antigen-Antibody Representation and Interaction

Design of an AIS must provide mapping of immune system components as well as some quantitative description of antigen-antibody interactions [36]. Antigens (*Ag*) and antibodies (*Ab*) are represented as strings of attributes having some length *L*, in shape space *S*. The *Ag* and *Ab* can be considered as points in the L-dimensional shape-space. The strings may be composed of integers, real numbers, bits or symbols. The antigen-antibody interaction would be a function of the distance *D* between them, i.e. hamming, euclidean or manhattan distance. Their affinity or closeness with each other is then proportional to their distance with each other.

The authors in [36] have quantified antigen detection through defining a *recognition region*. Detection of an antigen would occur if antigen exists inside a region of volume V_ε around an antibody. The region is called the recognition region and ε is its radius, also termed as cross reactivity threshold. It implies that an antigen is detected only when $D \leq \varepsilon$, where the value of cross reactivity threshold ε lies in the range $(0 < \varepsilon < L)$.

4.3 Anomaly Detection Using AIS

Anomaly detection systems operate by creating a profile of the system, which describes the normal system behavior. If the system deviates from normal behavior, an "intrusion" or "anomaly" is said to occur [44][39]. Thymus Model is particularly suited to network anomaly detection [36]. Negative selection allows to learn normal behavior of the system. For a self set *S*, a potential repertoire *P* of immature detectors can be formed by randomly generating strings of length *L*. Considering similarity measures, if the affinity of any string in *P* with those in *S* is less than or equal to a given cross reactivity threshold ε, the string is eliminated. The remaining strings form an available repertoire *A* of detectors that represent system abnormal behavior and would match only with the non-self set. If such a match occurs, it indicates that an anomaly has occurred in the system.

The idea of AIS was explored in the general area of computer security in [43][45][46]. AIS were first implemented effectively for network anomaly detection in [37], when a primitive architecture, Light Weight Intrusion Detection System (LISYS), was proposed to detect TCP-SYN flood attacks. LISYS architecture provides a basic framework for AIS based anomaly detection; it however, does not implement the notion of thymus, which was proposed and implemented in [38]. Additional empirical work utilizing AIS for development of network anomaly detection systems was done in [47][48][49]. A system with real valued, fixed size detectors was also proposed in [50]. It was later improved with variable detector sizes [51], whose classification accuracy is similar to the fixed size detectors but has slightly less memory requirements and relatively higher computational complexity. The AISs have thus been used extensively as general pattern learning systems that are diverse, distributed, robust and adaptable.

The concepts of Danger Theory and dendritic cells have been useful in the design and development of AISs, with specific application in anomaly detection. With focus on detecting intrusions in the field of computer security, the authors in [52] and [53] show how to map the latest immunological concepts of danger theory to detection of intrusions. Their emphasis has been to identify and analyze danger signals of various types that are needed to activate the adaptive immune response. Danger Project [54] was launched with the aim of applying AIS to perform intrusion detection. The project has produced two algorithms; the Dendritic Cell Algorithm (DCA) [55] and Toll Like Receptor Algorithm (TLR) [56]. The DCA was presented in [57] as modelling the behaviour and functionality of DCs in immune system. The algorithm emulates the sampling of multiple antigens, processing of signals, expressing the costimulatory molecules, transitioning states and presenting antigens in a safe/dangerous context. The DCA is a classifier whose accuracy depends upon the assigned weights that determine the overall context of the tissue signals. The algorithm has been found suitable for detecting anomalies based on initial experiment results. The algorithm has also been presented formally in [58].

Authors in [59] report results of their additional experiments with DCA that involved machine learning and detecting port scaning attack. These results indicate that the DCA has the potential to be used for classification problems for static data and for detecting intrusions or anomalies in real time applications.

5. BeeSec : Cryptographic Security for BeeAdHoc

BeeSec [31] is a cryptographic security framework developed for BeeAdHoc that utilizes public-key cryptography. Section 2.4 described how a malicious node could subvert BeeAdHoc routing by spoofing and falsifying the scout and forager packet header fields. Therefore, BeeSec provides protection to the routing process by using digital signatures to secure scouts and foragers against fabrication and tampering attacks. BeeSec uses digital signatures to perform:

- • **Packet Authentication** to ensure that data carried in header fields of scouts and foragers – source address, destination address, packet ID, routing information – is from authorised nodes.
- • **Integrity Check** of the source route to ensure that a malicious node has not removed a valid node from the source route.

5.1 Scout Processing

The algorithm for processing scouts in BeeSec is shown in Fig. 2. A scout has certain header fields that are non-mutable. The values of non-mutable fields are not changed during scout propagation between source and destination nodes. BeeSec performs authentication of these fields by computing a digital signature or authenticator and inserting into the scout header. BeeSec utilizes two such authenticators: (1) Forward Scout Authenticator, $Auth_{FS}$, and (2) Backward Scout Authenticator, $Auth_{BS}$.

The forward scout authenticator, $Auth_{FS}$, is a digital signature computed by each $node_i$ that broadcasts a forward scout. It is computed on selected non-mutable fields in scout header by using the private key of node with the signature function.

$$Auth_{FS(i)} = Sign\ (H\ (IP_s, ID_{sct}, D_{sct}), KR_i) \qquad (1)$$

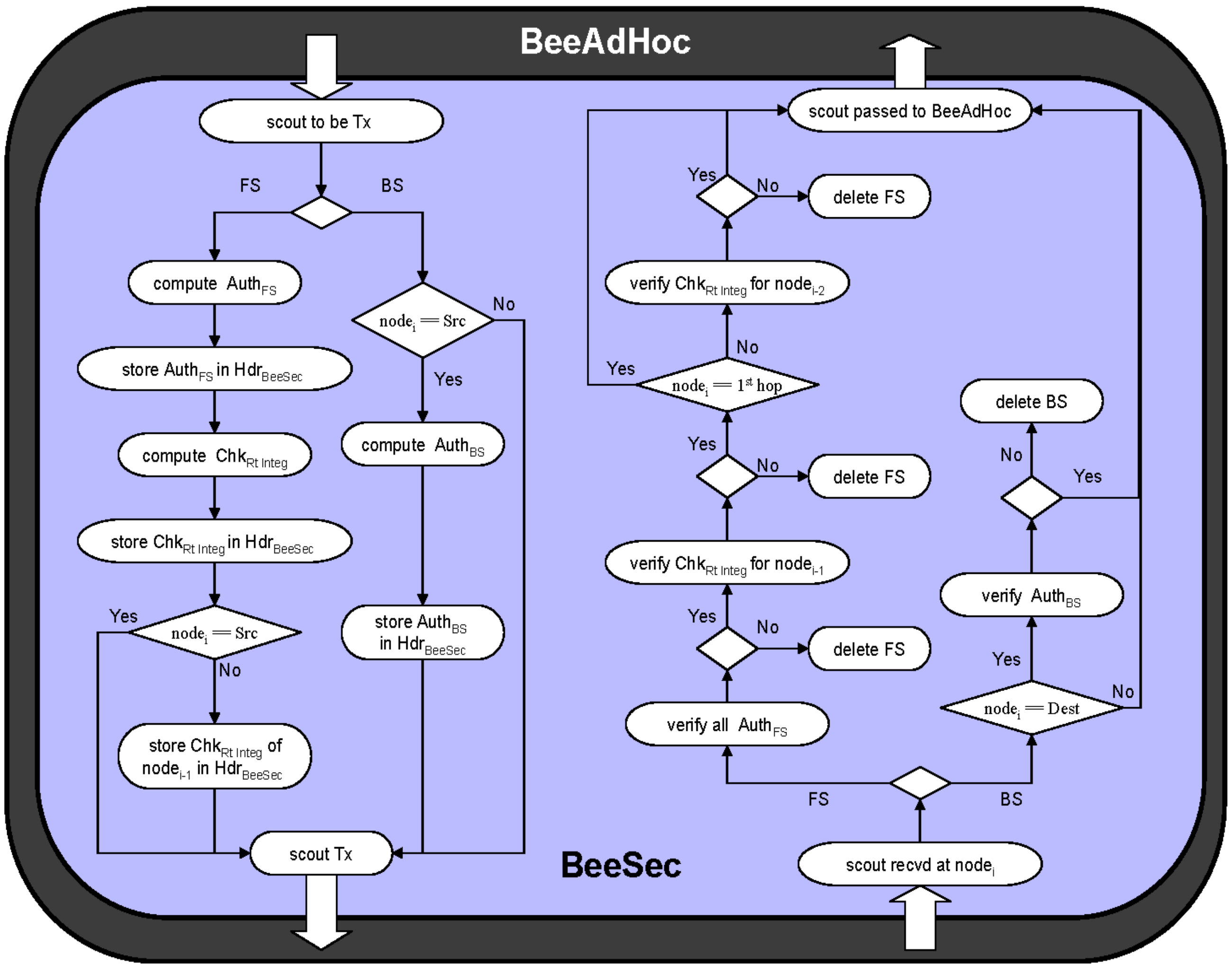


Fig. 2: BeeSec Scout Processing

Table I: List of BeeSec Symbols

Symbol	Description
IP_s, IP_d, R, R_i	source IP, destination IP, source route & source route till node i
FS_{sd}, BS_{sd}, F_{sd}	forward scout, backward scout and forager going from node s to d
Hdr_{BeeSec}	BeeSec protocol header
S_{sct}, ID_{sct}, D_{sct}	scout source, ID and destination
$Auth_{FS}$	forward scout header signature
$Chk_{RtInteg}$	scout source route signature
$Auth_{BS}$	backward scout header signature
$Auth_F$	forager header fields signature
$Auth_{RtInfo}$	forager route info signature
H(M)	hash of any message M
KU_s, KR_s	node public & private keys

Authenticator is placed inside the header of scout and the node broadcasts the scout. The next node, $node_{i+1}$, receiving the scout can retrieve the $Auth_{FS}$ values from the header and verify that the header field values have not been modified. The verification is done through public keys and signature verification function.

$$\text{Verify}\ (Auth_{FS(i)}\ ,\ H\ (IP_s,\ ID_{sct},\ D_{sct}),\ KU_i) \qquad (2)$$

The authenticator, $Auth_{BS}$, is used to authenticate the non-mutable fields of backward scouts. Upon reaching the destination, D_{sct}, a forward scout is sent back to the scout source, S_{sct}, in unicast mode. Before that, the destination, D_{sct}, adds the $Auth_{BS}$ to scout header.

$$Auth_{BS} = \text{Sign}\ (H\ (IP_s,\ ID_{sct},\ D_{sct},\ R),\ KR_s) \qquad (3)$$

Upon arrival back at source, the S_{sct} verifies the integrity of the backward scout; verification of digital signatures is done using public keys.

Additionally, the scout has the source route as mutable header field, whose value gets changed as nodes append their addresses to it. To ensure the integrity of this mutable field, the $node_i$ that broadcasts the scout adds the $Chk_{RtInteg}$ authenticator to the scout header.

$$Chk_{RtInteg} = Sign\ (H\ (R_i),\ KR_i) \qquad (4)$$

An intermediate $node_i$ receiving the scout checks to see that the route is indeed from $node_{i-1}$, as:

$$Verify\ (Chk_{RtInteg(i-1)}\ ,\ H\ (R_{i-1}),\ KU_{i-1}) \qquad (5)$$

However, above process cannot protect against route modification if $node_{i-1}$ had maliciously changed the route and also removed $Auth_{FS}$ for the corresponding node. To guard against this possibility, two $Chk_{RtInteg}$ authenticators are used in the header of scout. This allows a receiving $node_i$ to use individual $Chk_{RtInteg}$ from $node_{i-1}$ and $node_{i-2}$ to see if $node_{i-1}$ has modified the route maliciously.

$$Verify\ (Chk_{RtInteg(i-2)}\ ,H\ (R_{i-2}),\ KU_{i-2}) \qquad (6)$$

$Node_i$ drops the scout if the check fails. But if the verification holds, $node_i$ computes a new authenticator, $Chk_{RtInteg}$, places it in the scout header and broadcasts the scout. Therefore, predecessor of the predecessor is able to provide protection against possible source modification by a malicious node.

5.2 Forager Processing

BeeSec forager processing algorithm is shown in Fig.3. As in case of scouts, a forager also has fixed and mutable fields in its header. To protect the fixed value fields that include the IP_s, IP_d and R, the BeeSec utilizes the forager authenticator, $Auth_F$. When the source $node_s$ sends a forager, an $Auth_F$ authenticator is computed and placed in the header.

$$Auth_F = Sign\ (H\ (IP_s,\ IP_d,\ R),\ KR_s) \qquad (7)$$

The verification of $Auth_F$ is done when forager is received at the destination node, as:

$$Verify\ (Auth_F\ ,H\ (IP_s,\ IP_d,\ R),\ KU_s) \qquad (8)$$

The forager also has a mutable field comprising routing information, which is collected at each hop of the forager journey. Various parameters, such as packet delay or remaining battery life, are gathered by foragers to determine the quality of the forager route being traversed. To ensure that this information is not modified, $node_i$ sending the packet adds a route information authenticator, $Auth_{RtInfo}$, to the forager header.

$$Auth_{RtInfo(i)} = Sign\ (H\ (RouteInfo_i),\ KR_i) \qquad (9)$$

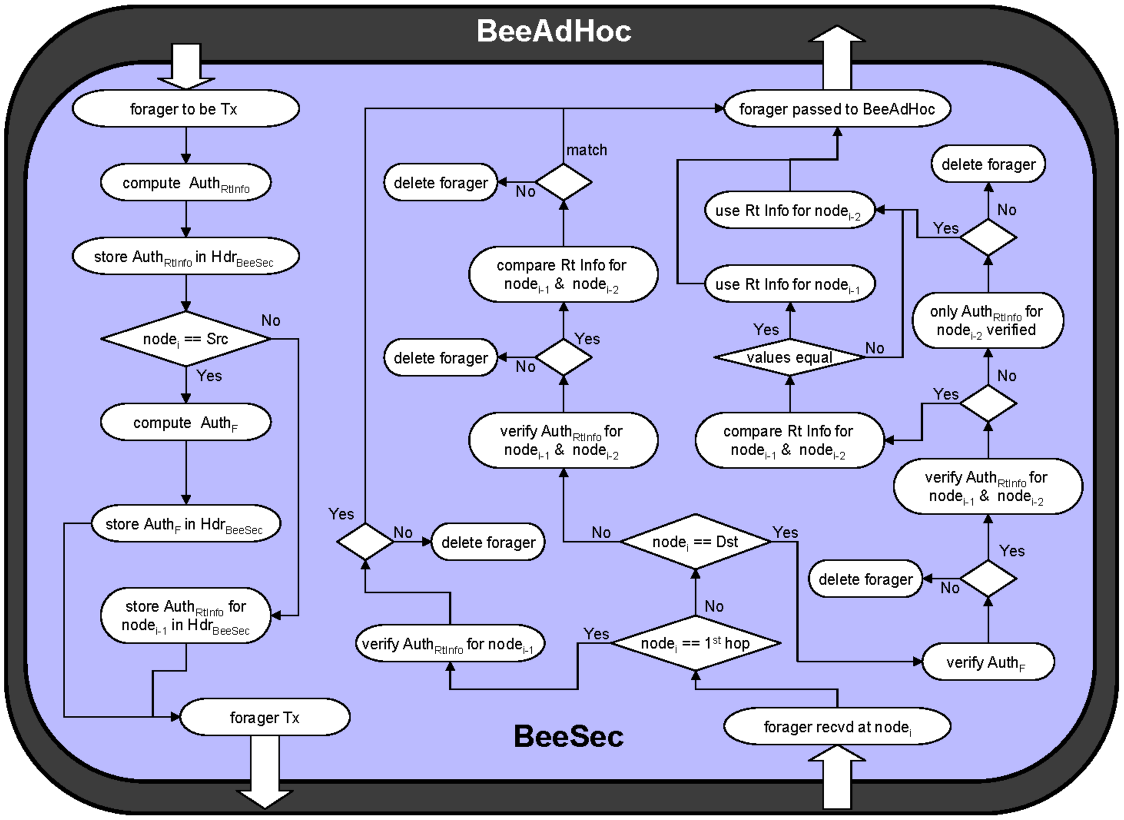

BeeAdHoc
forager to be Tx
compute $Auth_{RtInfo}$
store $Auth_{RtInfo}$ in Hdr_{BeeSec}
$node_i$ == Src
No
Yes
compute $Auth_F$
store $Auth_F$ in Hdr_{BeeSec}
store $Auth_{RtInfo}$ for $node_{i-1}$ in Hdr_{BeeSec}
forager Tx
Yes
No
delete forager
match
delete forager
compare Rt Info for $node_{i-1}$ & $node_{i-2}$
delete forager
verify $Auth_{RtInfo}$ for $node_{i-1}$ & $node_{i-2}$
verify $Auth_{RtInfo}$ for $node_{i-1}$
$node_i$ == Dst
$node_i$ == 1st hop
forager passed to BeeAdHoc
use Rt Info for $node_{i-2}$
use Rt Info for $node_{i-1}$
values equal
compare Rt Info for $node_{i-1}$ & $node_{i-2}$
delete forager
only $Auth_{RtInfo}$ for $node_{i-2}$ verified
verify $Auth_{RtInfo}$ for $node_{i-1}$ & $node_{i-2}$
delete forager
verify $Auth_F$
forager recvd at $node_i$
BeeSec

Fig. 3: BeeSec Forager Processing

However, a possibility exists that node$_i$ is a malicious node that has added a fake route along with its authenticator. Therefore, two Auth$_{RtInfo}$ authenticators are used in this case also to protect the route information against modification by a sending node; the Auth$_{RtInfo}$ from the node$_{i-1}$ is also included in the forager. The presence of two authenticators enables the next hop node$_{i+1}$ to verify that node$_i$ did not modify the route information coming from previous node$_{i-1.}$ These authenticator checks allow the destination node to use either the route information coming from node$_{i-1}$ or from node$_{i-2}$ to determine the correct dance number; if the route information has been modified by node$_{i-1}$, the value from node$_{i-2}$ is used, which would be much closer to the true route information value. However, an assumption is made that there are no two malicious nodes in immediate succession on a path.

5.3 Analysis of BeeSec Algorithm

The use of digital signature authentication enables the BeeSec to secure BeeAdHoc protocol against routing attacks by a malicious node. We have shown in [31] that BeeSec can counter five different types of fabrication/tampering attacks:

- **Attack-1:** Forward scout forging.
- **Attack-2:** Backward scout forging.
- **Attack-3:** Returning a forward scout as backward scout with modified route.
- **Attack-4:** Forging a spoofed forager.
- **Attack-5:** Modifying a forager's route information.

The BeeSec, therefore, has excellent security characteristics; in fact the cryptographic security system is able to correctly identify all malicious packets, thus achieving 100% detection. However, a limitation of such a system is its excessive control overhead and relatively higher energy consumption, compared with the BeeAdHoc, DSR and AODV protocols. This is due to the need to carry large size digital signatures in scout and forager headers for authentication. The overhead also results in BeeSec having lower throughput and higher packet delay [31].

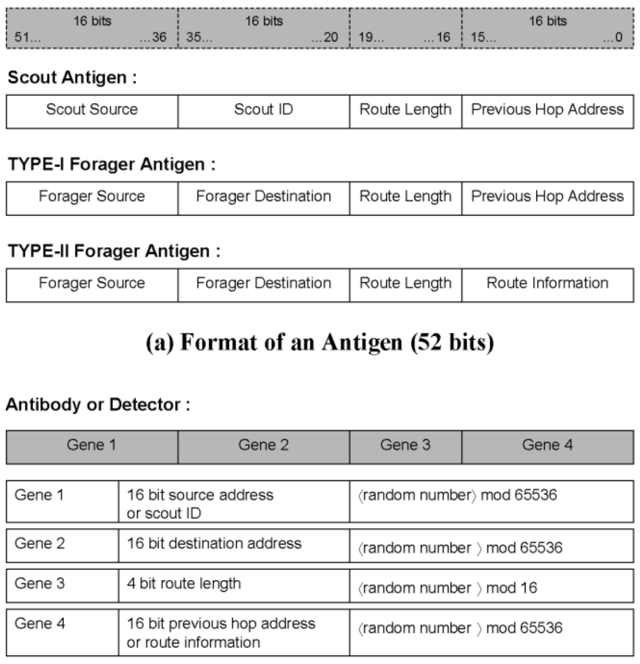
6. BeeAIS : Self/Non-Self Based AIS Security for BeeAdHoc Protocol

BeeAIS [32] is an Artificial Immune System security model for the BeeAdHoc routing protocol. BeeAIS employs negative selection for detecting anomalies. Using negative selection, BeeAIS is able to make a profile of the system behavior under normal routing and then monitor the system for instances of abnormal behavior patterns.

6.1 Mapping Concepts from Nature to Networks

6.1.1 Antigens and Antigen Formats: In BeeAIS, antigens are extracted from the incoming network traffic; an antigen comprises a set of packet header data that is expected to change under abnormal protocol operation. BeeAIS models antigens of three different types: a scout antigen to detect anomalies in scouts, and forager antigens of two types, Type-I and Type-II, for detecting modification in forager route information or source route. The antigen formats are shown in Fig. 4(a). Antigens are designed as bitstrings of length 52 bits, with each antigen comprising of four different genes. The four genes in each antigen have respective lengths of 16, 16, 4 and 16 bits.

6.1.2 Antibodies or Detectors: Antibodies or detectors are also selected as bitstrings of 52 bits length, and have genes of the same type as that of antigens. A detector/antibody is created by first generating all four gene values as random numbers, which are then scaled within the pre-determined range as shown in Figure 4(b). The final detector/antibody bitstring is created by concatenating all the four individual genes.



16 bits	16 bits		16 bits
51… …36	35… …20	19… …16	15… …0

Scout Antigen :

Scout Source	Scout ID	Route Length	Previous Hop Address

TYPE-I Forager Antigen :

Forager Source	Forager Destination	Route Length	Previous Hop Address

TYPE-II Forager Antigen :

Forager Source	Forager Destination	Route Length	Route Information

(a) Format of an Antigen (52 bits)

Antibody or Detector :

Gene 1	Gene 2	Gene 3	Gene 4

Gene 1	16 bit source address or scout ID	⟨random number⟩ mod 65536
Gene 2	16 bit destination address	⟨random number ⟩ mod 65536
Gene 3	4 bit route length	⟨random number ⟩ mod 16
Gene 4	16 bit previous hop address or route information	⟨random number ⟩ mod 65536

(b) Arrangement of Genes in an Antibody (52 bits)

Fig. 4: BeeAIS - Format of Antigens and Antibodies

6.1.3 Matching Function: BeeAIS evaluates the antigen–antibody interaction as a measure of their distance in the hamming shape space. Detection of an antigen is said to occur if $D \leq \varepsilon$, where D is the antigen–antibody hamming distance and ε is threshold of cross reactivity between an antigen and an antibody. In our system, the threshold is empirically selected to lie within the range ($0 < \varepsilon < L$), where $L = 52$.

6.2 BeeAIS Operation

The operation of BeeAIS comprises mainly the Learning Phase and the Protection Phase.

6.2.1 The Learning Phase*:* In this phase (Fig. 5), BeeAIS defines the system self by learning the system behavior in normal routing conditions. While BeeAIS profiles the normal behavior, it is assumed that there are no non-self antigens present in the system. In learning phase, each network node monitors the ongoing traffic to collect data required for forming the self antigens. On receiving a scout, the node forms a scout antigen, whereas on receiving a forager, it forms two forager antigens, Type-I and Type-II. A node may receive the same self antigen multiple times, therefore, it matches the new formed antigen with all antigens that have been gathered and drops the copies. When learning phase ends, BeeAIS generates a set of detectors (or antibodies) through negative selection of the collected self set. Detector are generated randomly and then matched with the self antigens to keep only the detectors that do not match. Such a detector set is representative of the system non-self. During detector generation, three detectors sets are formed at a node; a detector set for scouts and two detector sets (Type-I/Type-II) for forager. The sets of detectors created at different nodes differ substantially as nodes experience unique traffic sets during their learning phase.

When a node forms a Type-II forager self antigen, it also computes two parameters from the received forager data; the number of hops (Hops_{Fgr}) and the journey time ($\text{JourneyTime}_{\text{learnt}}$) taken by the forager to reach the current node. Both parameters are computed from the start of forager at source node. Each node maintains a list of the computed values {Hops_{Fgr}, $\text{JourneyTime}_{\text{learnt}}$}. Journey time is computed as:

$$\text{Journey Time} = T_{\text{curr}} - T_{\text{start}} \qquad (10)$$

On receiving a new forager, the $\text{JourneyTime}_{\text{learnt}}$ is determined as moving average of the computed forager journey times for the latest five received foragers.

6.2.2 The Protection Phase*:* In protection phase stored detector sets are used to match with the current behavior pattern of the system being monitored. Nodes gather

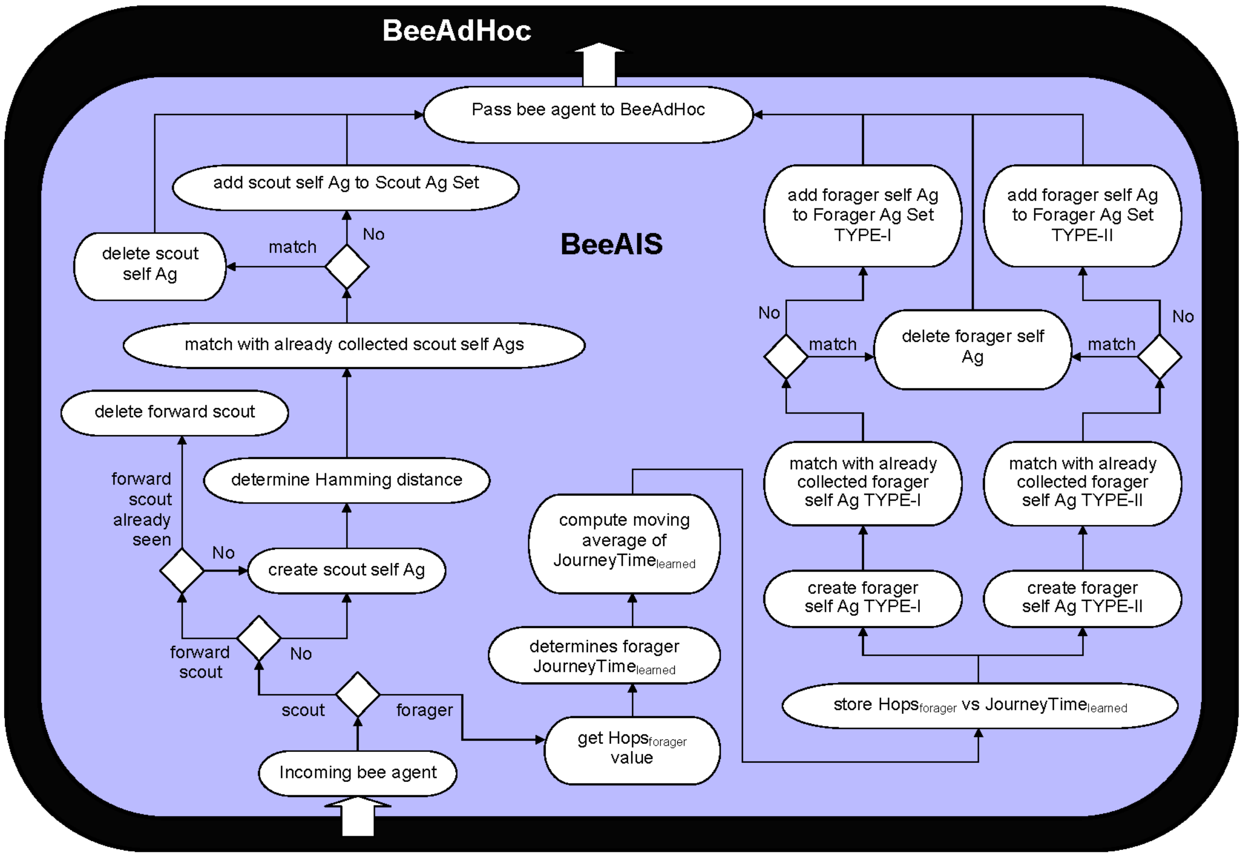

BeeAdHoc
Pass bee agent to BeeAdHoc
add scout self Ag to Scout Ag Set
BeeAIS
delete scout self Ag
match
No
match with already collected scout self Ags
delete forward scout
determine Hamming distance
forward scout already seen
No
create scout self Ag
forward scout
No
scout
forager
Incoming bee agent
compute moving average of JourneyTime_learned
determines forager JourneyTime_learned
get Hops_forager value
add forager self Ag to Forager Ag Set TYPE-I
add forager self Ag to Forager Ag Set TYPE-II
No
match
delete forager self Ag
match
No
match with already collected forager self Ag TYPE-I
match with already collected forager self Ag TYPE-II
create forager self Ag TYPE-I
create forager self Ag TYPE-II
store Hops_forager vs JourneyTime_learned

Fig. 5: Learning Phase of BeeAIS

antigens from the incoming traffic and measure their affinity with detector sets. If there is a match, it shows that a non-self antigen is present or an anomaly has occurred. Protection phase forager algorithm (Fig. 6) shows that both types of antigens (Type-I & Type-II) are matched with their respective detector sets. In BeeAIS, antigen Type-I are matched first to drop the matching foragers. After that antigen Type-II are matched, which detects route information modification.

TABLE II: List of symbols used in BeeAIS

Symbol	Description
T_{curr}, T_{start}	current time & forager start time at the source node
$Hops_{Fgr}$	number of hops covered by a forager till the current node
$JourneyTime_{learnt}$	taken by a forager to reach from source to the current node
$RouteInfo_{Fgr}$	expected correct value of route information in forager header

BeeAIS then corrects the route information. Tampered value is replaced with the $RouteInfo_{Fgr}$ value, which is obtained from the $JourneyTime_{learnt}$ value that is learnt by foragers during the learning phase.

$$RouteInfo_{Fgr} = T_{curr} - JourneyTime_{learnt} \qquad (11)$$

The node obtains the value of JourneyTimelearnt from the list {HopsFgr, JourneyTimelearnt} that is built during the node's learning phase.

6.3 Analysis of BeeAIS Algorithm

We have tested the security functionality of self/non-self based BeeAIS in [32], where we have shown that BeeAIS can perform detection of non-self antigens in four of the five attacks that are launched against the BeeSec; the attacks are described in Section 5.3 above. However, in the case of Attack-5, i.e, "*returning a forward scout as backward scout with modified route*", the attack is not detected by BeeAIS. This is because the attack is launched right at the start of the simulation when a node running BeeAIS is in its learning phase. The node, therefore, learns the attack behavior also as normal behavior, and is unable to recognize the attack as an anomaly during the protection phase.

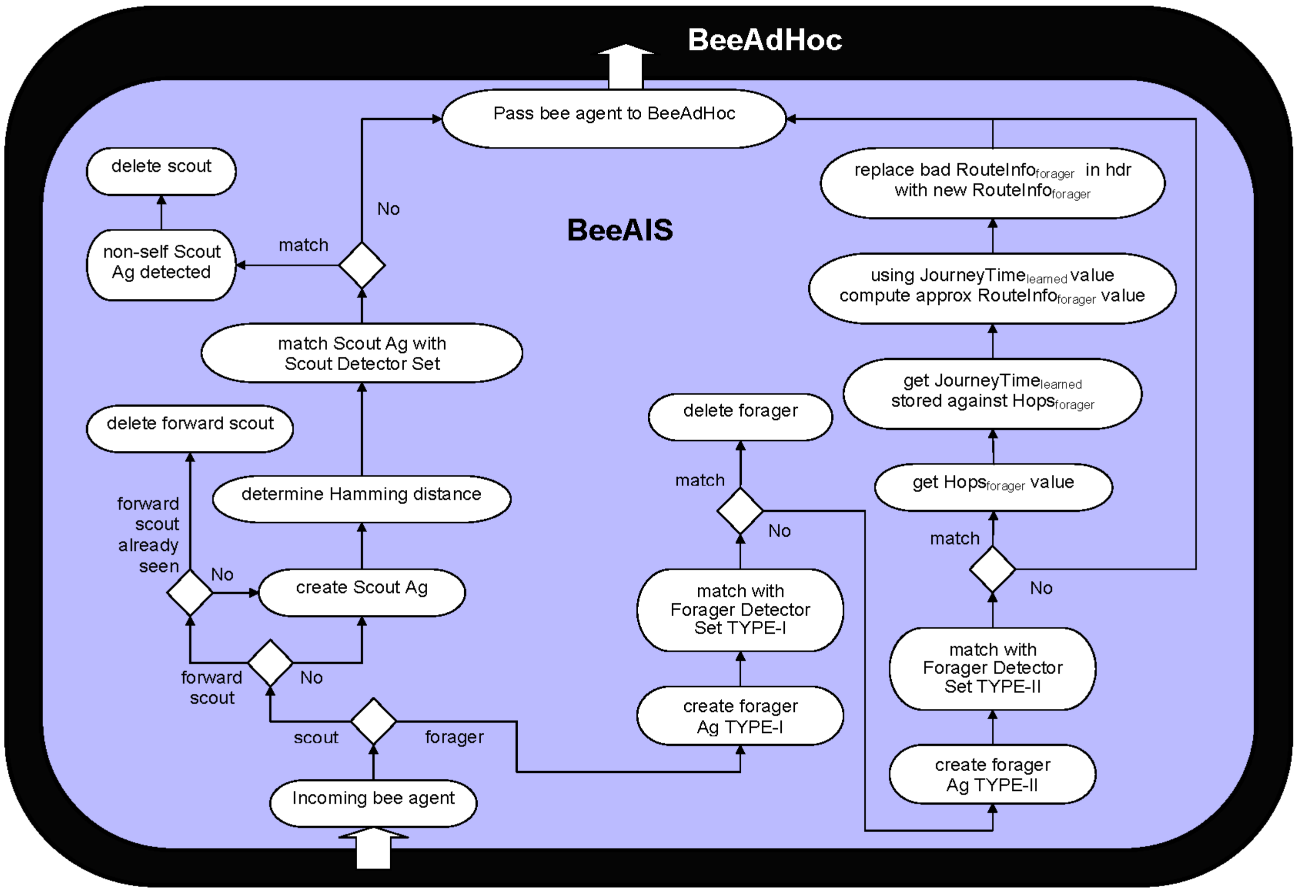

BeeAdHoc
BeeAIS
Pass bee agent to BeeAdHoc
delete scout
non-self Scout Ag detected
match
No
match Scout Ag with Scout Detector Set
delete forward scout
determine Hamming distance
forward scout already seen
No
create Scout Ag
forward scout
No
scout
forager
Incoming bee agent
delete forager
match
No
match with Forager Detector Set TYPE-I
create forager Ag TYPE-I
replace bad RouteInfo_forager in hdr with new RouteInfo_forager
using JourneyTime_learned value compute approx RouteInfo_forager value
get JourneyTime_learned stored against Hops_forager
get Hops_forager value
match
No
match with Forager Detector Set TYPE-II
create forager Ag TYPE-II

Fig. 6: Protection Phase of BeeAIS

The network performance of BeeAIS has been evaluated in [32] using a static grid of 49 nodes. The performance parameters include the packet delivery ratio, throughput, packet delay, control bytes, transmission efficiency and energy consumed per data byte; detailed description of these parameters is contained in [32]. The results of our network simulations show that BeeAIS has almost similar performance compated to BeeAdHoc, which implies that AIS based security does not impose any significant control or processing overhead on the base routing protocol. However, when we tested BeeAIS under mobility scenario in [33], the average throughput was found to be much less than that of BeeAdHoc. This is attributed to the learning phase of the BeeAIS protocol; algorithm learns the system normal behavior only once during the initial 50 seconds of its operational life. Consequently, during the protection phase, the system declares all newly observed behavior as anomalous even if the new behavior represents legitimate system activity. The investigation of BeeAIS lower average throughput revealed that the protocol drops a large number of scouts; the system falsely identifies a large number of newly seen scout self antigens as non-self scout antigens, which gives a high false alarm rate (FAR) of upto 67.4% for MANET of relatively small size [33]. As a result, BeeAIS does not discover new routes and application running on top drops the data packets. This reduces the protocol average throughput. We, therefore, conclude that AIS based on self non-self discrimination is not suitable to secure a dynamically changing environment such as MANETs with frequent changes in system self and non-self.

7. BeeAIS-DC : DC Inspired AIS Security for BeeAdHoc Protocol

In order to allow the AIS security framework to learn and adapt to frequent changes in system self and non-self as a result of node mobility in MANETs,

We developed a danger theory inspired system, BeeAIS-DC, for BeeAdHoc that was reported in [33]. BeeAIS-DC implements the function/behaviour of DCs to sense the occurrence of “danger” in the system, which in Biological Immune System is indicative of tissue damage caused by pathogenic infection. Likewise in MANETs, “danger” would indicate disruption of normal routing behavior caused by malicious attacks. The sensing of “danger” enables activation of the system against non-self antigens and system tolerization for the newly changed self.

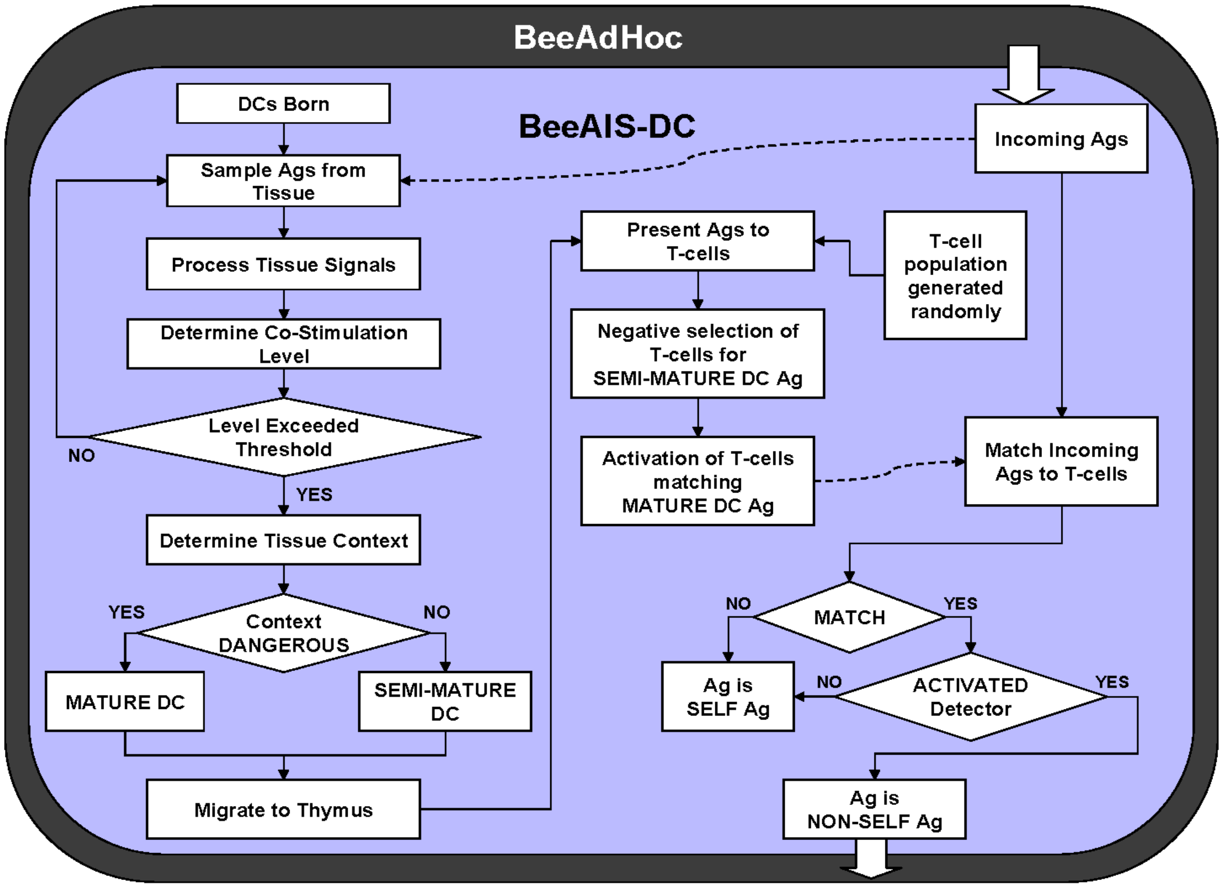


Fig. 7: BeeAIS-DC algorithm

BeeAIS-DC operation implements sampling of the scout antigens by dendritic cells from the body tissues. Sampling is done for both self and non-self antigens. After that, depending on whether a danger signal is present, the DCs adopt the differentiation pathways leading to semi-mature or mature states, and present the sampled scout antigens for activation of T-cells (detectors) in thymus. In case danger signal is not present, the changed normal behavior of the system can be presented as new self, instead of mis-interpretation as non-self. Description of the system in more detail is covered in the following sub-sections.

7.1 Antigens

BeeAIS-DC has adopted the antigen format used in [32]. A node creates an antigen when it receives a scout (forward or backward). The antigen is formed by extraction of relevant field values from the scout header and their concatenation to generate the antigen bitstring. The extracted scout header fields include the *scout source, destination*, *length of route* and *node ID of previous hop*. BeeAIS-DC antigens are binary strings in hamming shape space; each string is 52 bits long. The antigen, therefore, comprises 4 genes with each gene having 16, 16, 4 and 16 bits length. Gene values are determined from specific scout header field.

TABLE III: BeeAIS-DC symbols/parameters

Symbols / Parameters	Description
$Count_{FS}$, $Count_{BS}$	number of scouts received (forward & backward)
T_{curr}, UDINT	current simulation time, and periodic time interval to check for presence of danger signal to update dynamic sets of detectors
THRESH-RCVD-FS THRESH-RCVD-BS	upper limits of average forward and backward scouts received at a node to trigger declaration of node context as dangerous
CO-STIMUL-SCT	threshold to allow DC state to transition to MATURE, before presenting the non-self antigens in thymus to update detectors
NUM-DETS-SCT	maximum quantity of detectors available in the system to match with the antigens (scouts) being received at a node

7.2 Formation of DCs

A node froms a DC when it first sees a scout. At their birth, the node initializes the DCs with a number of attributes that are needed to implement their required functionality. The attributes includethe following:

7.2.1 DC Antigen: The antigen sampled from the received scout is appended to the dendritic cell. This emulates the tissue antigen that would later be presented in thymus during T-cells maturity.

7.2.2 DC Life: The BeeAIS-DC algorithm needs to compute the most recent system state viz-a-viz the self and the non-self. The DCs have, therefore, been assigned a short life after which they die a natural death. The objective is to ensure that thymus is always exposed to the latest system state.

7.2.3 DC State: When a DC is instantiated, it is an immature DC. After sampling the antigens, and when safe signals are present, the DC transitions to semi-mature state. In case of exposure to danger signals, the DC transforms into a mature DC, and moves towards thymus for presentating the antigens. The DCs, therefore, may exist in the system in mature, semi-mature or immature states.

Multiple scouts may be broadcast into the network to discover one route. As a node receives these multiple copies, it needs to discard the scouts already received and processed. The check is made by comparing the scout header field values ⟨*source*, *destination*, *source route*⟩ with those already stored in the DCs. A scout DC is assigned life at birth as per Eq. 12. On receiving a copy of the scout again, its life is re-initialized as per the same equation. Also, the node increments the count $Count_{FS}$ or $Count_{BS}$ parameters used for forward or backward scouts. These values are required by BeeAIS-DC for later determination of the occurrence of danger signals (Section VII-C).

$$\text{DC Life} = T_{curr} + \text{UDINT} \tag{12}$$

7.3 Computation of Danger Signal

The BIS requires sensing of danger signals prior to activation of DCs to allow their transition to mature or semi-mature states and subsequent migration to thymus for presenting the antigens. The BeeAIS-DC emulates the process by determining the presence of danger signals and updating the detectors sets at a fixed periodic *update interval* (UDINT). During the operation of protocol, scout DCs maintain a count of forward as well as backward scouts that arrive in the current UDINT interval. On expiry of every UDINT interval, an average of forward and backward scouts is computed for each path stored in the DCs. Whenever the average exceeds the nominated values of the thresholds for forward/backward scouts, THRESH-RCVD-FS and THRESH-RCVD-BS, the node raises the co-stimulation value related to the path. The danger signal is finally turned “HIGH” if co-stimulation value for the DC exceeds the scout co-stimulation level, CO-STIMUL-SCT. Consequently, the DC context turns dangerous and its state is changed to mature. The DC then moves over to thymus to expose the T-cells to the sampled antigen assumed as non-self.

The BeeAIS-DC mechanism of assigning the DC life and checking of various parametrs and thresholds ensures that danger signal only turns high when during the stipulated life of the scout DC, non-self antigens arrive continuously, are identified as suspicious and the suspicious UDINT periods exceed the threshold. The scheme ensures sufficient level of co-stimulation prior to final detection of danger signal, which helps in reducing the system false positive rate. Also, the danger signal once raised, remains as such for the following one more than CO-STIMUL-SCT intervals of UDINT. The scheme helps to reduce false negative rate when the system is unable to detect the attack in contiguous update intervals.

7.4 Updation of Detector Set

BeeAIS-DC performs antigen sampling by DCs followed by computation of danger signals with the aim to determine the state of DCs as either semi-mature or mature. The DC state is then used to generate a tolerogenic or immunogenic response; self tolerization is achieved through semi-mature DCs and the activation of immune response against non-self is achieved through mature DCs.

Initially, as the system starts, the node generates a set of random scout detectors. These detectors then undergo the negative selection process while being matched with antigens that are presented in thymus by semi-mature DCs. Detectors matching with an antigen are eliminated. This produces an antibody or detector set that is tolerant to self. The resulting detectors are then able to match only with non-self antigens and the system is thus tolerized to the self. In order to make up for the scout detectors falling below the NUM-DETS-SCTS level, the node generates new detectors, which are added to the scout detector set after subjecting to negative selection again. The process, therefore, results in a fixed set of detectors at each node that is capable of recognizing only the non-self antigens.

The antigens that are presented in the thymus by mature DCs, which have a “dangerous” context, are used to elicit an immunogenic response from the system against the non-self antigens. In BIS, mature DCs are used to activate the T-cells matching closely with the non-self antigens sampled by DCs. BeeAIS-DC emulates the T-cell activation through matching of detectors set with those antigens that are presented in thymus in mature state. Upon matching, the T-cell detector is transformed from the *naive* to an *activated* detector. Consequently, the activated detectors can now be used to initiate the system immune response by matching with any of the incoming antigens that are then said to be detected as non-self antigens.

7.5 Re-initializing or Eliminating DCs

The scout DCs are required to undergo re-initialization or elimination every UDINT interval of time. When a UDINT interval is over, DCs whose life has completed during the last interval are to die a natural death and are thus eliminated. The remaining DCs have their fields ($Count_{FS}$, $Count_{BS}$) re-initialized for fresh antigen sampling and data gathering, in order to sense the occurrence of new danger signals in the subsequent UDINT interval. The surviving mature and semi-mature DCs also have their states transformed to immature.

7.6 Matching Detectors with Antigens

During operation, the BeeAIS-DC needs to initiate an immunogenic response against the non-self antigens. Each node, therefore, upon receiving a network packet (scout bee) performs its classification as self or non-self depending on whether the extracted antigen matches with the scout detector set. Matching is done only for the activated scout detectors. In case of match, it is assumed that detection of non-self antigen is complete, and antigen is then dropped as a response.

7.7 Analysis of BeeAIS-DC Algorithm

The modelling of T-cells, DCs and the use of danger signal enables the BeeAIS-DC to implement a dynamic detector set; a detector set that keeps evolving with the system dynamics. The detector set is subjected to two simultaneous actions after periodic intervals of time; (1) tolerization to the antigens presumed to be self, and (2) activation against the antigens presumed to be non-self. This enables the detector set to perform an accurate detection of the non-self antigens by estimating the latest state of the system at any given time. Moreover, it does not require the system to undergo learning in the intitial phase of the system. BeeAIS-DC is, therefore, able to differentiate between the system self and non-self quite early in the its operational life. The investigation of BeeAIS-DC security characteristics through launching of routing attacks in [33] demonstrates the effectiveness of the system in providing security against these attacks.

However, BeeAIS-DC has a limitation that it prevents only the scout related attacks. Our experiments with the forager attacks, such as forging/spoofing foragers and modifying the foragers route information, were not successful, i.e, BeeAIS-DC was unable to detect these attacks. This is because the system uses hamming distance as the measure to determine affinity of detectors with non-self antigens, and the forager non-self antigens generated by the two forager attacks lie very close to the forager self antigens in hamming shape space. BeeAIS-DC is, therefore, unable to distinguish the forager non-self antigens from self antigens. The detection could be made possible if the antigen – detector affinity, with respect to the suspected non-self antigen, is increased through a process such as affinity maturation, as in Biological Immune System. This is possible with the use of B-cells, instead of the T-cells. Therefore, to improve the system attack detection performance, we propose to employ the activation of B-cells by DCs combined with affinity maturation. In this way, we would combine the concepts from both the self/non-self discrimination and danger theory to form a Composite AIS for effective misbehavior detection in the dynamic MANET environment. The proposed system details are covered in Section 9.

8. Network Performance

The fundamental design issue for a network security protocol is to achieve the desired security functionality. Equally important is to have minimal impact on network performance; the security overhead may not significantly degrade the system performance parameters. The network performance of our proposed security systems BeeSec, BeeAIS and BeeAIS-DC is evaluated and compared with the base protocol BeeAdHoc, as well as with the other classical MANET protocols, DSR and AODV. The ns-2 network simulator is used to carry out performance simulations. For simulations, the BeeAdHoc code from [25] is taken, while the code for our earlier security frameworks is from [32], [31] and [33]. The DSR/AODV code is from standard ns-2distribution. The simulation results reported in this paper add onto those reported earlier in [32], [31] and [33] in that:

- Simulation results reported earlier were for networks of upto 60 nodes, while the current results are for dense networks of upto 150 nodes where we ascertain the scalability characteristics of our proposed security frameworks.
- Additional performance parameters have been introduced for a broader comparison.
- BeeAIS network performance parameters are now reported in mobility scenario.

The BeeSec simulations for a network of more than 60 node could not be performed in ns-2 because of the extensive resource requirement of asymmetric cryptography.

8.1 Network Simulations

The version 2.29 of network simulator ns-2 is used for these simulations. Network topology comprises a node operation area of 2400 by 480 sq meters. The initial node positions are random. After that the node movement is in accordance with *random-waypoint* model. The pause time for a node ranges between 1 and 20 seconds. All the nodes act as sources and destinations, generating data at constant bit rate of 30 packets/second. The size of packets is 512 bytes. Six node scenarios are selected, with network size ranging between 10 to 150 nodes, except for BeeSec simulations. There are five randomly seeded simulations for each of the six node scenarios with each simulation run lasting for 1000 seconds.

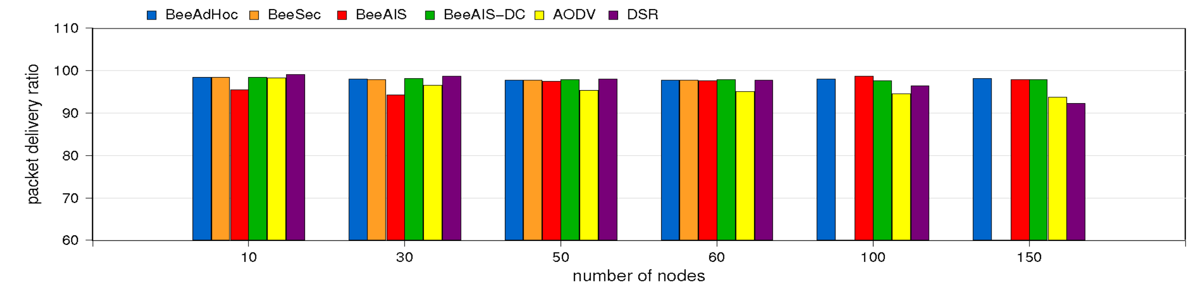
Results that are reported are an average of the values collected from 5 independent runs for each network scenario. The protocols performance is measured using the following parameters:

- **Packet Delivery Ratio:** Ratio between data packets that are delivered successfully to all destinations and the total data packets generated for those destinations.
- **Throughput:** Number of bits of data that finally arrive at the destination application and node, per unit of time.
- **Latency:** Average time difference between the packet generation at source node and its delivery to the destination. The wait period for reactive route discovery at source node is also included, in case the route to destination is not available.
- **Average Hops:** Average number of hops taken by the data packets for all traversed paths.
- **Transmission Efficiency:** Bytes of useful data delivered by the protocol to destination application per unit of control byte transmitted.

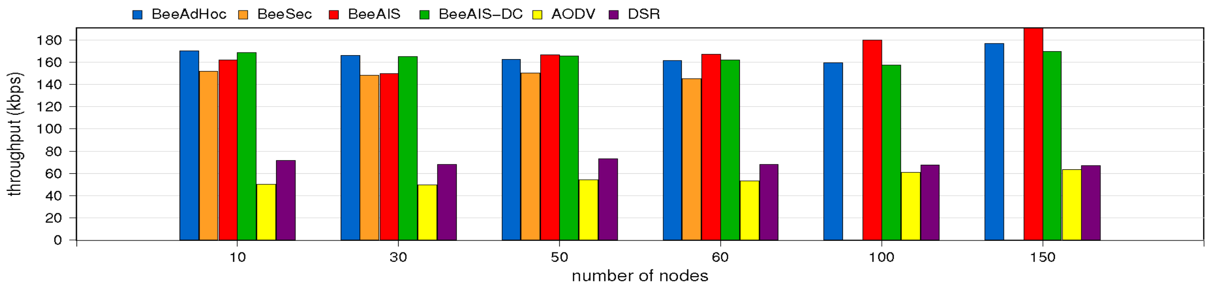
8.2 Network Performance of Protocol

In Fig. 8 we show results obtained from these simulations to compare the network performance of protocols. We can see that the performance parameters of BeeAIS and BeeAIS-DC are almost the same as BeeAdHoc. This implies that the AIS security extensions have minimal degradation effect on the base protocol, thus providing a security solution with minimal overhead. The cryptographic system, BeeSec, has lower throughput than BeeAdHoc, for different node scenarios, Fig. 8(b). This is attributed to the high control overhead of digital signature bytes that are carried inside the BeeSec packet headers for authentication. Resultantly, as seen in Fig. 8(e), the BeeSec efficiency to transmit data compared to BeeAdHoc , BeeAIS and BeeAIS-DC is the least.

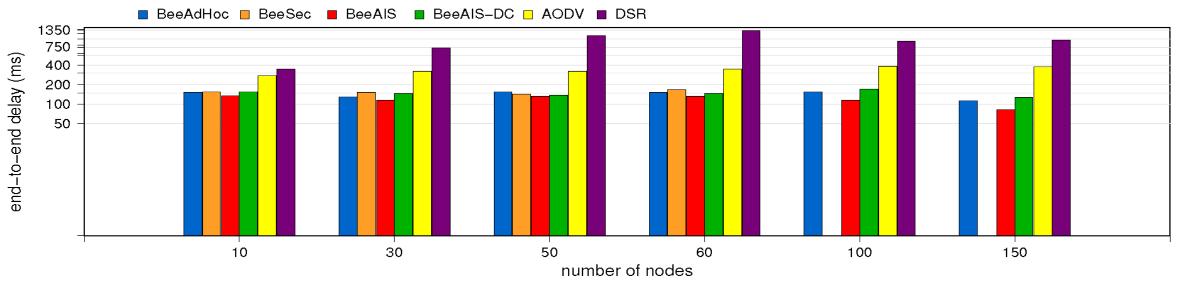
The self non-self based system, BeeAIS, has relatively lower packet delivery ratio and throughput for small networks, Fig. 8(a) and 8(b). BeeAIS delivers upto 4% less



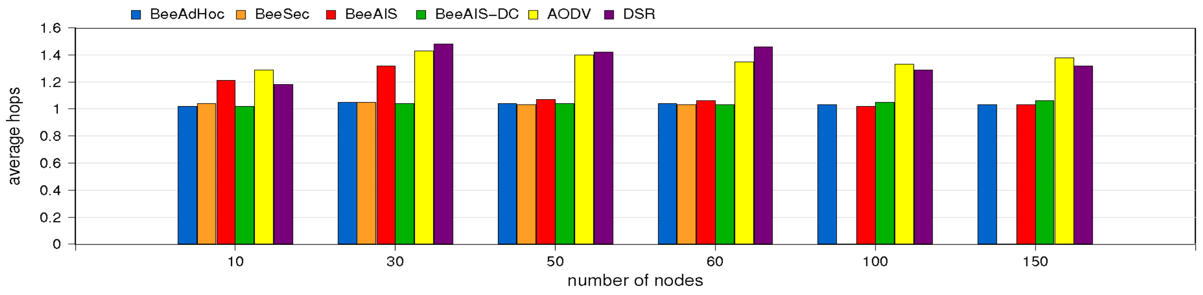
(a) Packet delivery ratio



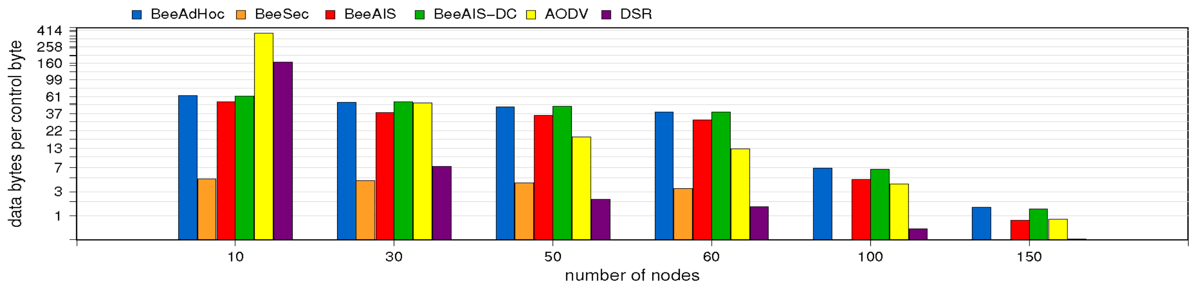
(b) Throughput



(c) Latency



(d) Average hops



(e) Transmission efficiency

Fig. 8: Comparison of network performance of BeeAdHoc, BeeSec, BeeAIS, BeeAIS-DC, AODV & DSR

packets due to its limitation of not adapting to a changing self under node mobility conditions [33]. This also slightly reduces its throughput for small networks of 10 to 30 nodes by upto 8%, Fig. 8(b). The effect, however, is greatly pronounced when we compare the average throughputs for the protocols. The average throughput of the protocol is defined as, "*the total bits of data delivered to the destination during the complete simulation*". Table V shows that the average throughput of BeeAIS is the least among the AIS security frameworks, from approximately 60% to 200% less than BeeAdHoc. This is because the BeeAIS erroneously drops some benign packets (scouts) as malicious, which it determines to lie outside the system normal behavior that is seen during the protocol learning, but which actually constitutes legitimate and newly changed system behavior.

BeeAIS FAR was studied in [33] and showed that the protocol drops as much as 67% of legitimate scouts as malicious for network of small size. This prevents new route discovery, which is needed frequently under node mobility. The data packets generated by the application are thus dropped due to unavailability of route, which adversely effects the protocol throughput. Table IV gives the average data packets generated by the application layer for BeeAdHoc, BeeSec, BeeAIS and BeeAIS-DC protocols and packets that the routing protocol drops due to route not being available. We find that maximum number of data packets are dropped by BeeAIS, from about 7% to 145% more than the other protocols, for different node scenarios. The transport layer at the sending node considers packet drop as an indication of network congestion. The sending TCP, therefore, initiates slow start [60], reducing the number of data packets handed down to the network layer for routing. BeeAIS receives from 28% to 67% less application data for routing, compared to the other protocols. This reduces the BeeAIS average throughput. In Fig. 8(b), the throughput for BeeAIS appears to increase for larger networks of 100 and 150 nodes. This occurs because the BeeAIS is asked to deliver from about 32% to 55% lesser packets than the other protocols, and the BeeAIS is able to deliver them efficiently. However, due to dropping of more packets, the average throughput for BeeAIS still remains low, Table V.

Since the BeeAIS-DC adapts its detector set as self changes, the legitimate packets are not identified as malicious. Resultantly, the protocol average throughput value is same as BeeAdHoc and higher than BeeAIS, Table V. The average throughput for BeeAIS-Comp is also higher than BeeAIS (37.6% to 66.6%). Even the BeeSec protocol, despite its cryptographic overhead of computation and communication, is able to achieve upto 61.7% average throughput than BeeAIS, the system based on self non-self discrimination.

Table IV: The comparison of data packets dropped by protocols versus those generated by applications

Nodes	*Protocol*	*Average data pkts*		*Pkt drop per 1000 generated*
		Pkts generated	*pkts dropped*	
10	***BeeAdHoc***	100864.20	228.20	**2.26**
	BeeSec	88403.60	210.00	2.38
	BeeAIS	33930.60	188.40	**5.55**
	BeeAIS-DC	100329.00	233.20	2.32
30	***BeeAdHoc***	138241.80	469.20	**3.39**
	BeeSec	122158.60	432.80	3.54
	BeeAIS	63935.00	405.60	**6.34**
	BeeAIS-DC	136899.00	468.20	3.42
50	***BeeAdHoc***	156178.00	632.80	**4.05**
	BeeSec	139014.60	575.00	4.14
	BeeAIS	86204.40	477.60	**5.54**
	BeeAIS-DC	157733.20	611.60	3.88
60	***BeeAdHoc***	153409.00	595.80	**3.88**
	BeeSec	126745.40	568.80	4.49
	BeeAIS	91599.60	454.80	**4.97**
	BeeAIS-DC	158559.60	580.60	3.66
100	***BeeAdHoc***	142609.40	764.20	**5.36**
	BeeAIS	92588.20	627.20	**6.77**
	BeeAIS-DC	135643.00	860.60	6.34
150	***BeeAdHoc***	93286.00	982.20	**10.53**
	BeeAIS	44751.00	907.00	**20.27**
	BeeAIS-DC	85861.60	991.80	11.55

Table V: Average network throughput (kbps)

Protocol	**Number of Nodes**					
	10	**30**	**50**	**60**	**100**	**150**
BeeAdHoc	421.8	576.3	649.6	637.9	592.4	385.0
BeeSec	369.5	508.8	578.0	526.6	-	-
BeeAIS	141.5	265.7	357.7	380.2	384.7	182.1
BeeAIS-DC	419.6	570.8	656.6	660.0	562.0	353.5
DSR	464.5	484.5	418.0	358.7	187.5	33.2
AODV	420.8	522.0	553.0	559.0	524.0	309.6

Moreover, it is shown by Figures 8(a) to 8(e) that the network performance of AIS based systems, especially the throughput, latency and average hops, are better than those for AODV and DSR. This result is consistent with our earlier work for BeeAIS [32] and BeeAIS-DC [33]. With these additional performance results for large networks of upto 150 nodes, it can be deduced that the AIS security systems definitely perform better or the same as the leading routing protocols for MANETs, AODV and DSR. This has important implications as it suggests that the combination of nature inspired routing protocol and AIS protection paradigm can be employed to provide much needed efficient security for routing in the MANET domain.

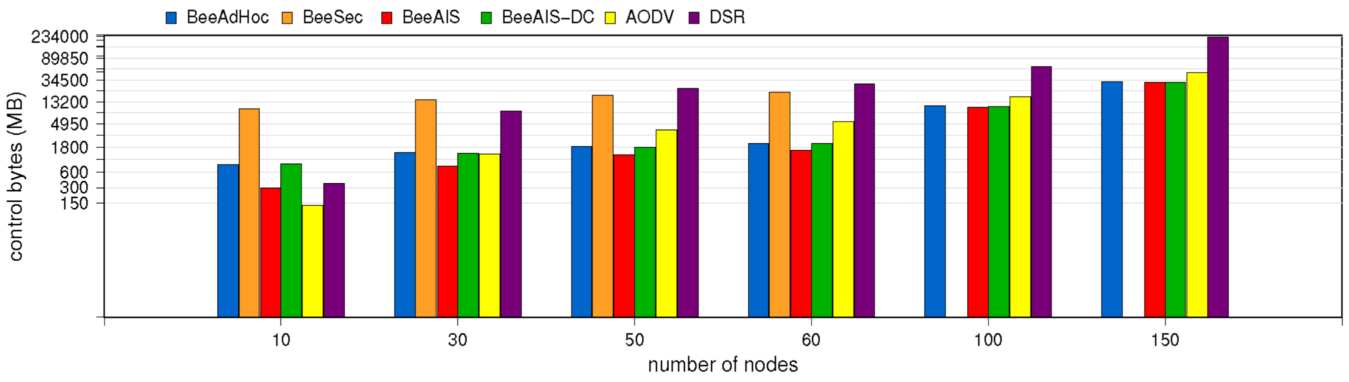
8.3 Control/Energy Overhead

Mobile nodes have serious limitations in bandwidth available battery power. These nodes operate in wireless medium for transmission/reception where data rates are limited, and operate on batteries, which have limited capacity. The inclusion of a security layer, however, constitutes additional load on the already constrained node/network resources. This mandates designing a security solution that is highly efficient with respect to minimizing the control overhead and power consumption during all of its operations.

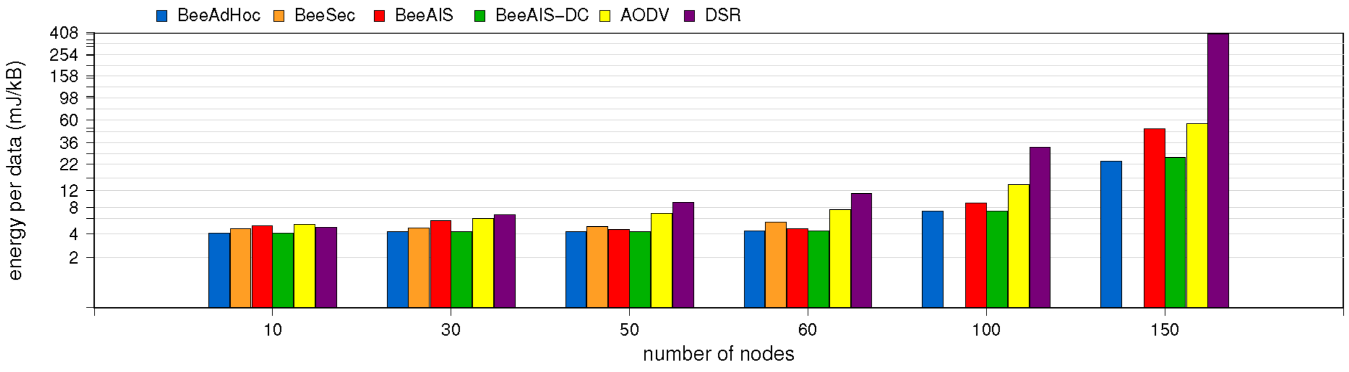
We measured the energy consumption and control overhead for BeeAdHoc protocol during packet transmission/reception operations and compared it with the security frameworks, BeeSec, BeeAIS and BeeAIS-DC, as well as with DSR and AODV protocols. For these comparisons, we have used the following metrics:

- **Energy consumption.** The energy consumed (Joules/kbits) in transporting control and data packets from source to destination node, per 1 kilo bit of useful data. The model presented in [61] is used to determine the energy spent during packet transmission (point-to-point & broadcast).
- **Control overhead.** The total number of control bytes (kbytes) transmitted, both in control packet and data packet headers.

Fig. 9(a) shows the control overhead for the compared protocols. BeeAIS-DC protocol has almost the same control overhead as that of BeeAdHoc, while that of BeeAIS is relatively lower. However, it is due to the BeeAIS transporting relatively lower number of data packets, which



(a) Total control bytes



(b) Energy consumption

Fig. 9: Protocol overhead in the form of control bytes & energy consumed

reduces the aggregate source route bytes carried in data packet headers. The BeeAIS-DC is also more efficient in control traffic than cryptography based BeeSec (89% to 91% lesser overhead). The BeeSec consumes more energy during transmission/reception of bits as compared to the security systems based on AIS, Fig. 9(b); the energy consumption is 10% to 21% higher compared to BeeAIS-DC. We can, therefore, easily come to the conclusion that a security system designed using the AIS primitives would have lower control and energy costs than a security system that utilizes cryptography, especially a public key system.

We can also see that for larger networks the AIS based security frameworks consume lesser energy in transporting user data as compared to the unsecured classical protocols for MANETs. The AODV protocol energy consumption is 160% higher and that of DSR is 1791% more than the BeeAIS-DC. AIS security solutions also have lower overhead in control traffic than that of DSR and AODV. BeeAIS-DC has from 36% to 61% less control overhead than AODV network of 50 or more nodes, and from 84% to 93% less control overhead than DSR network of 30 or more nodes. This clearly indicates the ability of the AIS to provide protection to MANET routing with lower impact on control overhead and energy efficiency of MANETs.

9. A Composite AIS Framework (CompAIS) for Anomaly Detection in MANETs

As discussed in Section 6 and Section 7, both the self/non-self based BeeAIS and the dendritic cells inspired BeeAIS-DC cannot effectively secure the dynamic MANET environment; BeeAIS is constrained by its fixed detector set, which is non-adaptable to systems whose self or non-self keeps changing frequently, while BeeAIS-DC employs T-cells that lack the ability to undergo affinity maturation to enable generation of a response that is more specific to a particular non-self antigen. Our experience with BeeAIS-DC indicates that AIS based anomaly detection can be made more effective in MANETs if, instead of the T-cells, we employ the DC activation of B-cells. We, therefore, propose to use a composite AIS system designed for BeeAdHoc protocol, called as CompAIS, that combines self/non-self discrimination with a DC based system to achieve the following objectives:

- Adaptively learn, both the system non-self and the self, through processing of signals for determining the appropriate safe/dangerous context by DCs.
- Dynamically update the B-cells detector set through negative selection based on the sampled antigens presented in “safe” context.
- Activate B-cells matching with the antigens presented in “dangerous” context, followed by affinity maturation to achieve higher detector affinity with the suspected non-self antigen.
- Segregate non-self antigens through pattern matching with the activated B-cell detectors.

The proposed system implementation of CompAIS through the activation of B-cells is shown in Fig. 10. The activated B-cells undergo clonal proliferation/hypermutation to achieve greater affinity with the non-self antigens that lie close to each other in the hamming shape space. Such an approach would not only improve the system detection performance but also enable a quicker secondary response from adaptive immune system through use of the memory detectors.

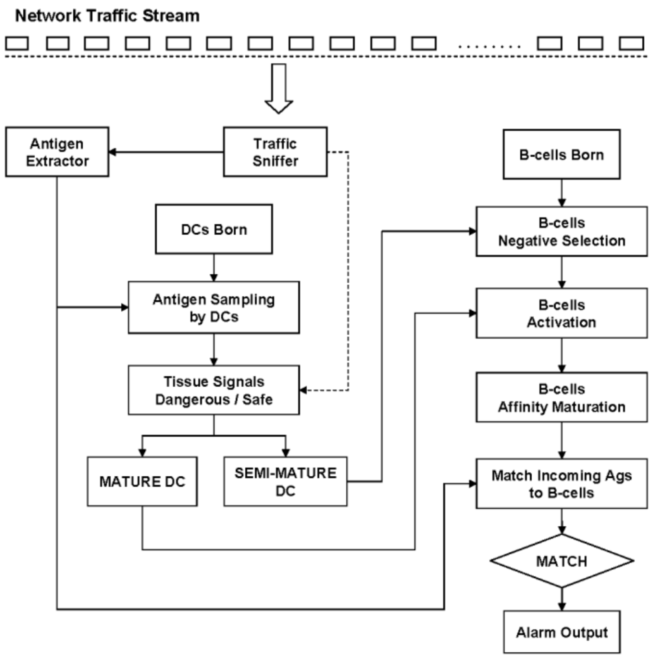


Fig. 10: A composite AIS security framework

10. Conclusion and Future Work

The inherently insecure MANETs require designing and implementing effective security mechanisms to allow secure, mobile computing environment. Security needs to be provided at the minimum possible cost to energy, which is a limited resource for a mobile node. The research presented in this paper is a cardinal step towards provisioning of energy efficient security in MANET environment, for which we have selected BeeAdHoc protocol that routes data based on honey bee hehavior. BeeAdHoc has primarily been designed for energy efficient routing. It has similar/better network performance compared to DSR, AODV but consumes significantly less energy, which clearly makes BeeAdHoc a preferred candidate for implementing energy efficient security.

In this paper, we carry out a review and comparison of two distinct approaches to MANET security; the cryptographic systems and the Artificial Immune Systems. We analyze and compare three security frameworks that we have developed for BeeAdHoc protocol; a cryptographic system BeeSec [31] and two AIS based systems, BeeAIS [32] and BeeAIS-DC [33]. We perform additional experiments to ascertain the network performance and scalability of these systems to larger networks. Our extensive simulations in ns-2 indicate that BeeSec has significantly higher control overhead due to the need to carry large size digital signatures in packet headers. The energy consumption (per kilo bit of data delivered to destination) by BeeSec is also higher compared with BeeAdHoc protocol and the AIS based security systems.

In the context of AIS based security, anomaly detection has been performed using two AIS approaches; (1) the classical self/nonself discrimination through negative selection algorithm, and (2) the more recent danger theory approach using dendritic cell algorithm. The self/non-self based BeeAIS does not suffer from high control overhead and excessive energy consumption like BeeSec; however, the protocol has poor average throughput. This is due to the continuously changing MANET topology, where the system self as well as the non-self keep evolving with system operation – the previously learnt benign routes may become malicious or the previously identified malicious routes may become benign. This makes it very difficult for BeeAIS to effectively segregate the new legitimate self from the non-self; a limitation attributed to the fixed detector set of BeeAIS, which is unable to adapt to a changing environment. As a result, the protocol drops legitimate scouts, which prevents new route discovery and reduces throughput. We have also observed that the danger theory based BeeAIS-DC has the ability to learn the changing system self and adapt its detector database to identify the new self as legitimate system activity. The BeeAIS-DC routing performance, is therefore, almost the same as the base protocol, BeeAdHoc. However, BeeAIS-DC is unable to detect the forager related routing attacks, where the non-self and self forager antigens lie very close to each other in the hamming shape space.

We, therefore, conclude that although the cryptographic security systems have the ability to secure the inherently insecure MANET routing, their high control overhead and energy consumption makes them less suitable for MANETs compared with the AIS based security systems. The use of AIS for security is beneficial from two aspects; (1) the Biological Immune System is successful in providing effective protection to human body in a distributed environment with self organization ability, which are the characteristics required for MANET security, and (2) the AIS does not involve transmitting additional control bytes or employs complex cryptographic functions. AIS based security, therefore, has significantly smaller control and computational load, and much lower energy consumption for packet processing and transmission, thus enabling prolonged battery life for mobile nodes.

To improve the limited detection capability of BeeAIS-DC, we have proposed a composite AIS model, CompAIS, for anomaly detection in MANETs by combining the functionality of adaptive and innate immune systems. We link the algorithms from self/non-self discrimination and danger theory through the activation of B-cells by DCs. This model uses innate system for presenting the antigen and its context to the adaptive system, which then identifies the non-self antigens. The use of B-cells allows to incorporate affinity maturation of B-cell antibodies to have a closer match with the non-self antigen. This would not only improve the system detection performance but also enable a quicker secondary response.

References

[1] Royer, E. and Toh, C. K.; *IEEE Personal Communications*, (April 1999), 46–55.

[2] Toh, C. K.; *Ad Hoc Mobile Wireless Networks: Protocols and Systems*, Prentice Hall, (2001).

[3] Perkins, C., and Bhagwat, P.; Proc. of ACM SIGCOMM'94 Conference on Communications Architectures, Protocols and Applications, (1994), 234–244.

[4] Perkins, C. and Royer, E.; Proc. of Second IEEE Workshop on Mobile Computing Systems and Applications, (February 1999), 90–100.

[5] David, B. J. and David, A. M.; *Mobile Computing*, Kluwer Academic Publishers, (1996), 153–181.

[6] Zhou, L. and Haas, Z. J.; *IEEE Network Magazine*, 13(6)(1999), 24–30.

[7] Suresh, S., Mike, W. and Raghavendra, C. S.; Proc. of Fourth ACM/IEEE Conference on Mobile Computing and Networking, (1998), 181–190.

[8] Christine, E. J., Krishna, M. S., Prathima A. and Jyh-Cheng C.; *Wireless Networks*, 7(4) (2001), 343–358.

[9] Laura, M. F.; *Mobile Networks and Applications*, 6(3) (2001), 239–249.

[10] Yih-Chun, H., Adrian, P. and David, B. J.; *Wireless Networks*, 11(1-2)(2005), 21–38.

[11] Zapata, M. G.; Internet-Draft, draft-guerrero-manet-saodv-05.txt, February, 2005.

[12] Stallings, W.; *Cryptography and Network Security - Principles and Practices*, Pearson Educ. Inc., 2003.

[13] Dasgupta, D.; *Artificial Immune Systems and Their Applications*, Springer-Verlag, Berlin, (1998), 3–21.

[14] Hofmeyr, S.; *An immunological model of distributed detection and its application to computer security*, PhD Thesis, University of New Mexico, 1999.

[15] Kim, J. and Bentley, P.; Proc. of EUFIT'99, 1999.

[16] Aickelin, U., Greensmith, J. and Twycross, J.; Proc. of 3rd International Conference on Artificial Immune Systems, LNCS 3239, Springer-Verlag, (2004), 316–329.

[17] Kim, J. and Bentley, P., Aickelin, U., Greensmith, J., Tedesco, G. and Twycross, J.; *Natural Computing*, 6(4) (2007), 413–466.

[18] Sarafijanovic, S. and Le Boudec, J. Y.; *IEEE Transactions on Neural Networks*, 16(5), Sep 2005.

[19] Caro, G. Di, Ducatelle, F. and Gambardella, L. M.; Proc. of Parallel Problem Solving from Nature, LNCS 3242, Springer-Verlag Berlin Heidelberg, (2004), 461–470.

[20] Caro, G. Di, Ducatelle, F. and Gambardella, L. M.; *European Transactions on Telecommunications*, 16(2)(2005), 443–455.

[21] Roth, M. and Wicker, S.; Proc. of Second Mediterranean Workshop on Ad-Hoc Networks, 2003.

[22] Roth, M. and Wicker, S.; In Proc. Stigmergic Optimization, Studies in Computational Intelligence, Springer Berlin Heidelberg, 31(2006), 155–184.

[23] Roth, M. and Wicker, S.; In Proc. of IEEE GLOBE-COM, Dec 2003.

[24] Wedde, H. F. and Farooq, M.; Proc. of IEEE Swarm Intelligence Symposium, (2005), 341–348.

[25] Wedde, H. F. and Farooq, M., Pannenbaecker, T., Vogel, B., Mueller, C., Meth, J. and Jeruschkat R.; Proc. of ACM Genetic and Evolutionary Computation Conference, (June 2005), 153–160.

[26] Wedde, H. F. and Farooq, M., Timm, C., Fischer, J., Kowalski, M., Langhans, M., Range, N., Schletter, C., Tarak, R., Tchatcheu, M., Volmering, F., Werner, S. and Wang, K.; Technical Report PG-460, LS III, School of Computer Science, University of Dortmund, 2005.

[27] Li, J., Blake, C., Douglas, S. J. De Couto, Hu I. L., and Morris, R.; Proc. of 7th ACM International Conference on Mobile Computing and Networking, July 2001.

[28] Rodoplu, V. and Meng, T. H.; *IEEE Journal on Selected Areas in Communications*, 17(8)(August 1999), 1333-1344.

[29] Seeley, T. D.; *The Wisdom of the Hive*, Harvard University Press, London, 1995.

[30] Von Frisch, K.; *The Dance Language and Orientation of Bees*, Harvard University Press, Cambridge, 1967.

[31] Mazhar, N. and Farooq, M.; Proc. 9th Annual Conference on Genetic and Evolutionary Computation, (July 2007), 102–109.

[32] Mazhar, N. and Farooq, M.; Proc. 6th International Conference on Artificial Immune Systems, LNCS 4628, Springer-Verlag Berlin Heidelberg, (Aug 2007), 370–381.

[33] Mazhar, N. and Farooq, M.; Proc. 10th Annual Conference on Genetic and Evolutionary Computation, July 2008.

[34] Stallings, W.; *Data & Computer Communications*, Pearson Educ. Inc., 2000.

[35] Perlman, R.; *Network layer protocols with byzantine robustness*, PhD Thesis, Deptt of Elec. Engg. and Computer Science, MIT, 1998.

[36] De Castro L. N. and Timmis, J.; *Artificial Immune Systems: A New Computational Intelligence Approach*, Springer-Verlag, UK, Sep 2002.

[37] Hofmeyr, S. and Forrest, S.; *Evolutionary Computation Journal*, (2000), 443–473.

[38] Ranang, M. T.; *An artificial immune system approach to preserving security in computer networks*, MS Thesis, Norwegian University of Science and Technology, June 2002.

[39] De Castro, L. N.; Proc. of the ICANNGA, April 2001.

[40] Marrack, P. and Kappler, J.; *Scientific American*, 263(3)(Sep 1993), 48–55.

[41] Matzinger, P.; *Annual Reviews in Immunology*, 12(1994), 991–1045.

[42] Hart, E. and Timmis, J.; Proc. of 4th International Conference on Artificial Immune Systems, LNCS 3627, (2005), 483–497.

[43] Forrest, S., Perelson, A., Allen, L. and Cherukuri, R.; Proc. of IEEE Symposium on Security and Privacy, IEEE Computer Society, (1994), 202.

[44] Debar, H., Dacier, M. and Wepsi, A.; *Computer Networks*, (2000), 361–378.

[45] Forrest, S., Hofmeyr, S. and Somayaji, A.; Proc. of IEEE Symposium on Research in Security and Privacy, IEEE Computer Society Press, 1996.

[46] Forrest, S., Hofmeyr, S. and Somayaji, A.; *Computer immunology*, Communications of the ACM, (1997), 88–96.

[47] Kim, J. and Bently, P. J.; *IEEE Transactions on Evolutionary Computing*, 2001.

[48] Kim, J. and Bently, P. J.; Proc. of Congress on Evolutionary Computation (CEC), (2001), 1244–1252.

[49] Kim, J. and Bently, P. J.; Proc. of Genetic and Evolutionary Computation Conference, (2002), 1330–1337.

[50] Gonzalez, F., Dasgupta, D. and Nino, L. F.; Proc. of 2nd International Conference on Artificial Immune Systems, LNCS 2728. Springer-Verlag, 2003.

[51] Ji, Z. and Dasgupta, D.; Proc. of the ACM Genetic and Evolutionary Computation Conference, USA, 2004.

[52] Aickelin, U., Bentley, P., Cayzer, S., Kim, J. and McLeod, J.; Proc. of 2nd International Conference on Artificial Immune Systems (ICARIS'03), LNCS 2728, Springer-Verlag, (2003), 147–155.

[53] Aickelin, and Cayzer, S.; Proc. of 1st International Conference on Artificial Immune Systems, University of Kent, Canterbury Printing Unit, (2002), 141–148.

[54] Danger Project. http://www.dangertheory.com.

[55] Greensmith, J.; *The dendritic cell algorithm*, PhD Thesis, University of Nottingham, Oct 2007.

[56] Twycross, J.; *Integrated innate and adaptive artificial immune systems applied to process anomaly detection*, PhD Thesis, University of Nottingham, Jan 2007.

[57] Greensmith, J., Aickelin, U. and Cayzer, S.; Proc. of 4th International Conference on Artificial Immune Systems, LNCS 3627, Springer-Verlag, (2005), 153–167.

[58] Greensmith, J., Aickelin, U. and Twycross, J.; Proc. of 5th International Conference on Artificial Immune Systems, LNCS 4163, (2006), 404–417.

[59] Greensmith, J., Twycross, J., and Aickelin, U.; Proc. of Congress on Evolutionary Computation (CEC), (2006), 664–671.

[60] Peterson, L. L. and Davie, B. S.; *Computer Networks: A Systems Approach*, Morgan Kaufmann, 1996.

[61] Feeney, L. M. and Nilsson, M.; Proc. of IEEE INFOCOM, 2001.