Security and Intrusion Detection

This chapter deals with security and intrusion detection in MANET. Security is a key concern in MANET. In the designed scheme, two key metrics trust and confidence is evaluated before data transmission. Based upon these key parameters data is shared and transmitted along the host disjoint paths. Moreover to detect the intrusion, statistical data is collected and Markov Chain (MC) is constructed from this data. From the generated MC, classifier is constructed which is used to detect the intrusion. Rest of the chapter is organized as follows:

- Section 9.1 deals with the security and intrusion detection issues
- Section 9.2 gives problem formulation and assumptions
- Section 9.3 presents detection of malicious behavior of neighboring hosts
- Section 9.4 describes secure routing with data transmission
- Section 9.5 gives an intrusion detection system
- Section 9.6 deals with implementation
- Section 9.7 presents result and discussion
- Section 9.8 summarizes the chapter

9.1 Issues

In MANET host mobility changes quickly that causes frequent topological changes. This causes the need of security in MANET. The key issues for security in MANET are: Global trust development, bandwidth constraints, host selfishness not to forward packets or selective forwarding of packets, denial of service, and detection of intrusion if present in the network.
have achieved the goal of security in two ways. Firstly the routing path is made secure by identifying the malicious host present in the path. For this two key parameters are evaluated namely trust and confidence, so that hosts in an unsecure path are avoided from transmission. Secondly, the data is sent along the multi paths with each path have its own share. So enemy is not able to construct the message until it did not have full shares.

9.2 Problem Formulation and Assumptions
At the network layer, we assume that there are totally K host disjoint paths, path1, path2, ..., path k, available from source to destination. We use vector $p = \{p_1, p_2, ..., p_k\}$ to denote the security characteristics of the paths, where $p_i$ is the probability that path $i$ is compromised. Also $n = \{n_1, n_2, ..., n_k\}$ is used to denote a share allocation. These shares are allocated into the $k$ available paths, such that $n_i \geq 0$ and $\sum_{i=1}^{k} n_i = N$. We assume that if a host is compromised, all the shares traveling through that path are compromised. Therefore, we define that a path is compromised when any one or more of the hosts along the path is compromised. For each path, we consider that if it is compromised, all the shares allocated to it are compromised. As the paths chosen are host-disjoint, the probability that the message is compromised equals the probability that $n$ or more shares are compromised.

There are three different ways a host can engage in different malicious acts. Below we highlight these three different malicious behaviors:

- A host engaging in selfish behavior by not forwarding packets meant for other hosts, or selectively forwarding smaller packets while discarding the larger ones
- A host falsely accusing another host for not forwarding its packets, thus isolating the host from normal network operation.
- A host place itself in active route and then coming out to break the route, thus forcing more route request packets to be injected into the network. By repeating this malicious act, a large number of routing overhead is forcefully generated wasting valuable bandwidth and disrupting normal network operation.

9.3 Detection of Malicious behavior of Neighboring Hosts

Now we will evaluate the three possible malicious behaviors described above. Two key parameters namely trust and confidence of each host is selected. These parameters for all the three cases of selfishness are calculated as follows:

**Case 1:**
Let $N_n$ be the neighbors hosts list (each host maintains a list of its neighbors by receiving Hello messages), $R_n(i)$ be the total number of packets host $h$ has forwarded to host $i$ for further forwarding, $C_n(i)$ be the total number of packets that have been forwarded by $i$ and noticed by $n$. With the above defined parameters host $h$ calculates local trust value (LTV) about $i$ as follows:

$$LTV = f\left(\frac{C_n(i)}{R_n(i)}, C_n'(i)\right),$$

where $C_n'(i)$ is the confidence level of host $n$ on $i$.

This gives quite an accurate estimation about the trust of a host when weighted by the confidence level. But the trust computation does not take into account a host’s selective forwarding behavior, where it only forwards small packets while selectively discarding the larger ones. To reflect this kind of malicious behavior, we have computed the confidence level $C_n'(i)$ as given below:

$$C_n'(i) = \frac{\sum \left(\frac{C_n(i)}{R_n(i)}\right) \times PS}{\sum PS}$$

\[\text{.................................................................(9.1)}\]
Where \( PS \) is the packet size.

Host \( h \) computes its confidence level on \( i \) after sending a specified number of packets to \( i \). The computation is weighted by the packet size to reflect the selective forwarding behavior of a host. Each host updates its LTV and sends it to its neighbors. When a host \( h \) receives the LTV from host \( i \), it computes the overall trust value (OTV) of \( i \) as given below:

\[
OTV = \frac{\sum C_n'(i) \times C_i'(j) \times \frac{C_n(i)}{R_n(i)}}{\sum C_n'(i) \times C_i'(j)},
\]

where \( C_n'(i) \) is the confidence level of \( n \) on \( i \) and \( C_i'(j) \) is the confidence level of \( i \) on \( j \) and \( \frac{C_n(i)}{R_n(i)} \) is the forward ratio of host \( i \) on \( j \).

So based upon the above defines parameters for trust and confidence, each host calculates and evaluates its neighbors selfish behavior for not forwarding some packets or selectively forward smaller packets. This information is used by multi path secure routing agent to multicast to other hosts for construction of reliable route.

**Case 2:**

We foresee a threat where a host falsely accuses another host of not forwarding its packets, eventually to isolate that host an untrusting one. This malicious act is reflected in the trust computation, where every host is given a chance to defend it. We have modified equation (9.1) to reflect such a malicious act in the computation of the confidence level. The modified equation is shown below:

\[
C_n'(i) = \frac{\sum (\frac{C_n(i)}{R_n(i)}) \times PS}{\sum PS} \times \alpha,
\]

where \( \alpha \) is the accusation index of \( h \) by \( i \).

Where \( \alpha = \begin{cases} 1 & \text{for } \text{false accusation} \\ 0 & \text{for no false accusation} \end{cases} \), depending upon \( i \) falsely accuses \( n \) or not.
Host $h$ keeps a track of the packets it received from $i$. If $h$ finds out that $i$ is falsely accusing it for non-cooperation, it recomputed its confidence level on $i$ by taking into account the accusation index. It then broadcasts the new LTV with new $C_n'(i)$ thus resulting in computation of a new OTV, which is low enough to punish $i$. Thus, any sort of malicious behavior of $i$ by falsely accusing other hosts gets punished eventually.

**Case 3:**
If the malicious behavior of a host where it forces to change the network topology frequently (Trust against malicious topology change) is detected, the confidence level is changed in order to punish the malicious host. However, detection of such a behavior is not easy, as any such topology change can be viewed as a normal characteristic in MANET.

In this approach, each host maintains a table called a neighbor remove table, where it keeps track of any host moving out of the path. This information is broadcasted by MA from time to time so that each host has its own updated table of neighbors. It computes the mean, $\rho$ of the time difference of any particular host leaving the network. If $\rho$ is found lower than a threshold value $\lambda$, then the host is identified as malicious and the confidence level is computed as follows:

$$C_n'(i) = \frac{\sum(C_n(i) \times PS)}{\sum PS} \times m',$$

where $m'$ is the malicious index of host $i$. The value of $m' = \{0,1\}$, depending upon the value of $\rho \leq \lambda$ or not. The choice of the threshold value is selected based on the typical application for which MANET is deployed. A network that demands frequent topology change have a higher threshold to accommodate the normal network behavior. Finally, to combine all the malicious behavior discussed earlier and to reflect
those behaviors in trust computation, the confidence level of host $h$ on $i$ is computed as shown below:

$$C_n'(i) = \frac{\sum (C_n(i) \times PS) \times \alpha \times m'}{\sum PS}$$

The final overall trust value (OTV), when computed based on the local LTV, will reflect the different malicious behavior of a host as computed in the confidence level, and hence any malicious act gets detected and punished.

### 9.4 Secure Routing and Data Transmission

Once the identification and punishment of the malicious hosts occurs, then the next task is to send the data along the paths. There are three steps followed for data transmission: multi path secure routing, secret sharing, and share allocation so that adversary can not destroy the message sent in between. Each of these steps is executed by the respective agents. These agents not only communicate with each others but also choose and execute the corresponding policies defined in policy manager.

#### 9.4.1 Multipath Secure Routing

Most of the designed protocols for multiple paths use source routing technique to control disjointness of the paths at source host. In such protocols, whenever source needs a path to a certain destination, it starts a route discovery process by broadcasting the route discovery messages throughout the network, the intermediate hosts that have a valid route to the destination reply by sending back route reply packet. Also cache is necessary for source host to store the routes previously found so that the host does not have to perform the costly route discovery for each individual packet. There are multiple paths to combat the frequent topological changes and link instability problems in MANET, since the use of multiple paths diminish the effect of possible link failures.
Let in the MANET, each host $h$ is associated with a security related parameter $k_i$. This security related parameter $k_i$ is the probability that host $h$ is compromised [110] and [186]. The identification of compromised host is calculated in Section 9.2. The probability that source to destination $(s, d)$ path $(s, h_1, h_2,..., h_f, d)$ is compromised is given by:

$$p = 1 - (1 - k_1)(1 - k_2)...(1 - k_d)$$

The cost of the link between host $h_i$ and $h_j$ is defined as

$$C = -\log \sqrt{(1 - k_i)(1 - k_j)}.$$

Then the cost from Source to Destination $(s, h_1, h_2,..., h_f, d)$ is

$$C' = -\log((1 - k_1)(1 - k_2)...(1 - k_d)) - \frac{1}{2}\log((1 - k_s)(1 - k_d))$$

is minimized

iff $$-\log((1 - k_1)(1 - k_2)...(1 - k_d))$$ is minimized

, i.e., $$\log((1 - k_1)(1 - k_2)...(1 - k_d))$$ is maximized, iff $p = 1 - (1 - k_1)(1 - k_2)...(1 - k_d)$ is minimized

Hence the path found by the conventional shortest path algorithm (Dijkstra Algorithm for shortest path) will be the most secure path and is chosen by corresponding MA to route the packets so as to increase the reliability of these packets.

**9.4.2 Secret Sharing**

The secret message is divided into $N$ pieces such that in order to get message, the adversary must compromise at least $N$ shares. With fewer than $N$ shares, the enemy cannot learn anything about the message and has no better chance to recover the secret. This gives the desirable security properties. We use secret sharing such that the generation of the message shares and the reconstruction of the messages are all linear operations over a finite field.
Suppose that the system secret is \( L \) which has to be transmitted to destination \( D \) from source \( S \). Divide it into \( N \) pieces, \( L_1, L_2, \ldots, L_N \), each is called a share. Each of \( N \) participants has one share of the secret. The generation of the secret shares guarantees that fewer than \( N \) participants cannot learn anything about the system secret \( L \). A secret sharing agent has two clones. First is used to generate and distributes the shares among the participants. The second is used to collect shares from the participants after a finite interval of time and recomputed the secret. The MA obtains the \( j \)th share by evaluating a polynomial of degree \((N-1)\) and \([110]\) and \([186]\).

\[
f(x) = (a_0 + a_1 x + a_2 x^2 + \ldots + a_{N-1} x^{N-1}) \mod p
\]

### 9.4.3 Share Allocation

The share allocation policy is selected with the objective of maximizing the message security. This is executed by the respective MA that chooses a host disjoint path for share allocation so that adversary can never get the message. Hence in designed scheme two key parameters trust and confidence is evaluated by MAs and based upon that a MA decides its route and update the topology accordingly. Secondly data is transmitted along the secure disjoint paths so that adversary could not be able to retrieve the transmitted data.

### 9.5 Intrusion Detection

This section presents IDA model and MC based anomaly detection algorithm. IDA serving as the line of defense for information systems, and is indispensable for MANETs with high security requirements. A local detection model aggregating abnormal behaviors is constructed to reflect recent activities in order to achieve low false positive ratio and high detection ratio.

#### 9.5.1 Problem Formulation and Assumptions

We model the network as an undirected graph \( G \). A graph \( G = (V, E) \) consists of a set of \( n \) hosts (vertices) and \( m \) host pairs. The set of hosts,
denoted by $V = \{1, 2, \ldots, n\}$, represents the network enabled mobile devices; the set of edges, denoted by $E$, represents the wireless communication links. Each link $(i, j)$ is bidirectional and connects host $i$ and $j$. Link $(i, j)$ is removed when the distance between host $i$ and $j$ is greater than the radio transmission range, while a new link is formed when their distance is less than or equal to the radio transmission range. The topology of $G$ is constantly changing.

9.5.2 Mobile Agent assisted Intrusion Detection System (MAIDS)

From the system aspect, we attach an IDA to each mobile host. These IDA run independently and monitor local activities to detect abnormal behaviors. We choose to implement an anomaly detection algorithm. It is difficult to obtain the complete trace of the attacks, which are often required in designing a misuse detection algorithm.

9.5.3 Architecture of IDA

An IDA consists of - the data collection module, the detection engine, LACE, GACE, and the intrusion response module. The data collection module is mainly responsible for collecting the security related data from various sources. The detection engine uses the data which is parsed, and filtered by the data collection module to perform intrusion detection locally. LACE will locally aggregate and correlate the detection results from different detection engines in an IDA. In an environment with high security requirements, it is necessary to have multiple detection engines, which enable the use of different detection techniques. They complement each other to improve the detection performance. The functionality of LACE is to combine the detection results of different local detection engines. The functionality of GACE depends on the type of mobile hosts. If the host is an intrazone host, GACE is mainly responsible for transmitting the locally generated alerts to the gateway hosts in the same zone; if the host is a
gateway host, GACE is to aggregate and correlate the detection results from the LACE of its own agent and the LACEs of the intrazone hosts in the same zone, and to cooperate with the GACEs of the gateway hosts with which it has physical connections. The intrusion response module is to handle the generated alarms.

**Figure 9.1:** Architecture of IDA

### 9.5.3.1 Data Collection Module

The functionality of the data collection module is to collect the security related data from various data sources and send it to detection engines. There may exist many data collection modules in an IDA. Each module is responsible for collecting data from a particular data source. There are mainly two different data sources: network packets and host audit. Because this research is focused on the routing attacks, the data source mainly consists of the routing activities, topological patterns and traffic changes, etc.
9.5.3.2 Detection Engine

Different detection techniques are deployed in different detection engines in order to improve the detection performance. Misuse-based detection techniques operate based on the known attack scenarios and system vulnerabilities. Their main disadvantage is that they are only effective in detecting known attacks. It is expected that many new different types of attacks may occur in MANETs, so anomaly based detection techniques play a crucial role in the MANET environment. Several types of anomaly detection techniques exist - string-based [187] [188], statistical-based [189] [190], and specification-based [191], etc. They differ in the format and the amount of available audit data as well as the modeling algorithms. An advantage of statistical-based anomaly detection techniques is their capability of explicitly representing and handling variations and noises involved in activities. Due to arbitrary mobility of hosts, MANETs are expected to have more dynamic activities. Therefore, we utilize a statistical based anomaly detection approach - MC based anomaly detection algorithm to meet the challenges.

9.5.3.3 LACE

Because different detection techniques are deployed in an IDA, it is necessary for the LACE to aggregate and correlate the different detection results before transmitting them to the GACE. The local IDA for a mobile host is capable of operating in a standalone mode and detect attacks against the host. Since MANETs are constrained by bandwidth, energy consumption, and process capability, it is desirable to correlate the alert information on the local hosts first, before transmitting every alert across the network. The correlation is very simple, for instance, based purely on the source address.
9.5.3.4 GACE
The functionality of the GACE depends on the host types- if the host is a gateway host, its GACE uses the detection results from the IDA of intrazone hosts in the same zone and neighboring gateway hosts. If the host is an intrazone host, the functionality of the GACE is to distribute the outputs of its LACE to all of the gateway hosts in the same zone.

9.5.3.5 Intrusion Response Module
The countermeasures taken by the intrusion response module are different due to the different intrusions, network services, applications and confidence in evidence. Possible countermeasures may include identifying the intruders, reinitiating the communication channels and excluding the compromised hosts from the networks.

9.5.4 Construction of MC
We have developed a MC based anomaly detection algorithm. It characterizes the normal behavior of the system and captures the characteristics of the temporal sequence of the system data by utilizing which states it moves between and with what probabilities. Assume \( X_1, X_2, X_3, \ldots, X_t \) is a time-series representation of a given event sequence, where \( X_t \) denotes an event occurring at time \( t \). We computed the transition probability matrix as follows:

\[
p_{ij} = \frac{N_{ij}}{N_{i}}, \quad \text{Where } N_{ij} \text{ is the number of observation pairs } X_i \text{ and } X_{i+1} \text{ with } X_t \text{ in state } i \text{ and } X_{i+1} \text{ in state } j \text{ and } N_{i} \text{ is the number of } X_t \text{ in state } i.
\]

We define the from state as \( X_{i}, X_{i+1}, \ldots, X_{i+w-1} \) and to state as \( X_{i+w} \). \( w \) is a parameter that characterizes MC.
Let \( \xi \) denote the set of symbols. A *trace* over \( \xi \) is a finite sequence of symbols. The set of finite traces over \( \xi \) is denoted as \( \xi^* \) and the set of traces of length \( w \) is denoted as \( \xi_w \). Given a trace \( \psi \) over \( \xi \), \(|\psi|\) denotes its length.

Each *from state* in MC model is associated with a sequence of symbols, which is defined on \( \xi \cup \{\emptyset\} \) and its length is \( w \). Each tuple \( (s, s') \) is a *state pair* that represents the state transition from \( s \) to \( s' \). All *states* are stored in a hash table \( H \). The use of the hash table is to speed up the processing. Each *state* and *transition* is associated with a *counter*, which indicates how many times this *state* or *transition* has occurred. Construction of MC is presented in Figure 9.2

The initial *from state* of the MC model is associated with a symbol sequence of length \( w \) consisting of the first \( w \) symbols of \( \psi \). \( w \) is *window size*, which indicates the number of symbols associated with *from state*.

```plaintext
/* Construction of MC, Input: Statistical Data, Output: MC consists of hash tables */
1. Set *from state* to first \( w \) symbols of \( \psi \)
2. Shift \( \emptyset \) left by \( w \) positions
3. While (not reaching the end of \( \psi \))
4. Set *to state* to the first symbol of \( \psi \)
5. Shift \( \emptyset \) left by one position;
6. If (\( *from state \notin H \)); set *from state* \( \rightarrow H \) and counter of *from state* in \( H \) to 1
   Else
   Increase the counter of *from state* by 1
7. If (transition (\( *from state, to state \notin H \)); set (\( *from state, to state \)) \( \rightarrow H \) and increase the counter of (\( *from state, to state \)) in \( H \) to 1
   Else
   Increase the counter of (\( *from state, to state \)) by 1
8. Shift *from state* left by one position
9. Append *to state* to end of *from state*
```

**Figure 9.2**: Pseudo Code for Construction of MC

If the *transition* (\( *from state, to state \)) is not in hash table \( H \), it is inserted

For each trace \( \psi \), it sets *from state* and *to state*. If the *from state* is not in
hash table $H$, it is inserted into $H$ and associated with a counter 1. If the transition $(\text{from} \_ \text{state}, \text{to} \_ \text{state})$ is already inserted into hash table $H$, its associated counter is increased by 1. If the transition $(\text{from} \_ \text{state}, \text{to} \_ \text{state})$ is already inserted into hash table $H$, its associated counter is increased by 1. We can imagine $H$ is constructed using a window of size $w$ sliding through the trace $\psi$, each time by one position. After all the traces in the training set have been processed, hash tables are constructed. They store the possible normal states and their transitions, respectively. Each state and transition are associated with a positive integer. The probability of the transition $s, s'$ in $H$ is calculated as:

$$P(s, s') = \frac{N(s, s')}{N(s)}$$

where $N(s, s')$ is the counter associated with the transition $s, s'$ and $N(s)$ is the counter associated with the state $s$.

### 9.5.5 Constructing of Classifiers Using MC

Let $l_\zeta = \{\delta_1, \delta_2, \ldots, \delta_{m-w+1}\}$ is the sequence of symbols corresponding to $\zeta$, where $\delta_1 = \{s_1, s_2, \ldots, s_w\}$, $\delta_k = \{s_k, s_{k+1}, \ldots, s_{k+w-1}\}$. The algorithm using MC to construct the classifier is described in Figure 9.4.

The algorithm first initializes two real numbers $A$ and $B$ to 0. $\delta_i$, the from state, is obtained after repeatedly shift $\zeta$ left by one position from its beginning. If the transition $(\text{from} \_ \text{state}, \text{to} \_ \text{state})$ exists in $MC$, $A$ is increased by 1 and $B$ is increased by $F = 1 - P (\text{from} \_ \text{state}, \text{to} \_ \text{state})$. Therefore, $F$ sums up all the probabilities of the transitions from the from state that are not equal to the current $(\text{from} \_ \text{state}, \text{to} \_ \text{state})$. If the transition does not exist in $MC$, $A$ is increased by 1 and $B$ is increased by the
value \( Z \) (depending upon the locality of host). After each calculation, if \( B/A \) exceeds a preset threshold value \( r \), \( \zeta \) is considered containing malicious activities (return \textit{Anomalous}). \( A \) and \( B \) are also updated after each calculation to accommodate the \textit{locality frame} scheme. This requires the deletion of the oldest value, and the addition of newest value.

<table>
<thead>
<tr>
<th>Input: Constructed MC and trace ( \zeta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output: Classifier containing Normal or Anomalous behavior</td>
</tr>
</tbody>
</table>

1. Initialize \( i = 1; A = 0; B = 0; \mu(\zeta) = 0, w = 1 \);
2. While \( (i < \text{lengthof } \zeta) \)
3. Set \( \text{from } \_ \text{state} \) to the sequence \( \delta[i] = (\zeta[i], \zeta[i+1], ..., \zeta[w+i-1]) \)
4. Set \( \text{to } \_ \text{state} \) to \( \zeta[w+i] \)
5. if (transition( \( \text{from } \_ \text{state} , \text{to } \_ \text{state} \)) is in MC)
   \[
   A = A + \sum_{s \in \text{Next}(\text{from } \_ \text{state})} P(\text{from } \_ \text{state}, s) = A + 1
   \]
   \[
   B = B + 1 - P(\text{from } \_ \text{state}, \text{to } \_ \text{state})
   \]
// Next( \( \text{from } \_ \text{state}, s \)) indicates all the \( \text{to } \_ \text{state} \) associated with current \( \text{from } \_ \text{state} \) */
Else
   ( \( \text{from } \_ \text{state} , \text{to } \_ \text{state} \)) is not the transition of MC
   \[
   A = A + 1
   \]
   \[
   B = B + 1 - P(\text{from } \_ \text{state}, \text{to } \_ \text{state})
   \]
6. Adjusting the value of \( A \) and \( B \) over the past locality frame
7. \( i++ \)
8. \( \mu(\zeta) = B / A \)
9. if (\( \mu(\zeta) \geq r \))
10. Return anomalous;

\textbf{Figure 9.3:} Pseudo Code for construction of Classifiers using MC

Different selections of \( A, B, \) and the alert threshold \( r \) will result in different detection models. Intuitively, the metric \( B/A \) measures how well the MC model predicts the trace \( \zeta \), e.g., a lower \( B/A \) indicates that the MC predicts the trace well.

For simplicity, we use
\[
A = A + \sum_{s \in \text{next}(\text{from}_\text{state})} P(\text{from}_\text{state}, s) = A + 1
\]

\[
B = B + 1 - P(\text{from}_\text{state}, \text{to}_\text{state})
\]

where \( \text{next}(\text{from}_\text{state}) \) denotes the set of \( \text{to}_\text{state} \) associated with this \( \text{from}_\text{state} \). In this definition, \( B \) sums up all the probabilities of the transitions from the \( \text{from}_\text{state} \) that are not equal to \( (\text{from}_\text{state}, \text{to}_\text{state}) \).

When analyzing the trace as in Figure 9.4, if the transition \( (\text{from}_\text{state}, \text{to}_\text{state}) \) has a low probability according to the constructed MC, then it might be an anomaly. Since \( B \) adds up all the probabilities of the transitions from the \( \text{from}_\text{state} \) that are not equal to \( (\text{from}_\text{state}, \text{to}_\text{state}) \), the small occurring probability of \( (\text{from}_\text{state}, \text{to}_\text{state}) \) will lead to relatively large \( B \).

### 9.6 Implementation

We have implemented the designed system according to parameters shown in Table 9.1[Appendix B]. The following metrics are chosen for intrusion detection:

- **Path optimality ratio** which is defined as

  \[
  \text{Path optimality ratio} = \frac{\text{hop count of the actual path}}{\text{hop count of the optimal path}}
  \]

- **Mean Time to the First Alarm (MTFA)** - This metric is defined over anomalous traces and measures how fast the classifier detects the attack. It is desirable that the IDA detect the attack as quickly as possible.

- **False Positive Ratio** - It is the ratio of percentage of decisions in which normal data become anomalous.
Detection Ratio - It is computed by dividing the total number of correct detections by the total number of victims in the anomalous data.

Communication Overhead - The communication overhead is computed as the number of transmission of local alerts in a given time period for single host.

Suppose for a given host, at time $t_1$, there are $N_1$ routing entries, the routing entry set is $S_1$, and the sum of hops of all routing entries is $H_1$ and at time $t_2$, there are $N_2$ routing entries, the routing entry set is $S_2$, and the sum of hops of all routing entries is $H_2$. We define $PCR$ and $PCH$ as following:

Percentage of the Change in Route entries (PCR) - It is calculated as $\frac{|S_2 - S_1| + |S_1 - S_2|}{|S_1|}$. $|S|$ indicates the number of elements. $(S_2 - S_1)$ means the newly increased routing entries during the time interval $t_2 - t_1$, and $(S_1 - S_2)$ means the deleted routing entries during $t_2 - t_1$. They together represent the changes of routing entries in $t_2 - t_1$.

Percentage of the Change in number of Hops (PCH) - It is calculated as $\frac{(H_2 - H_1)}{H_1}$. $H_2 - H_1$ indicates the changes of the sum of hops of all routing entries during the time interval $t_2 - t_1$.

9.7 Result and Discussion

In both 50 and 100 hosts networks, designed scheme have better path optimality than AODV. This is because MAs are used for route discovery process which attempt to make a longer jump in each hop of rebroadcast in the process of route discovery, which naturally leads to shorter end-to-end route in term of the number of hops. Figures 9.4 and 9.5 present path optimality ratio for 50,100 hosts using the designed scheme and AODV. Results obtained show that when the network becomes more stable, then path optimality ratio of designed scheme is very close to 1.0, i.e., designed scheme achieves very good end-to-end performance. We have also observed from Figures that the path optimality ratio for designed scheme drops as the
pause time increases, while the path optimality of AODV increases a little bit instead. Because as the pause-time increases, the degree of mobility decreases and the network topology is more stationary. So once a route is discovered between source and destination, the route will be used for quite a long time because no route breakage is likely to occur. This is the reason why designed scheme performs better than AODV in MANET.

**Figure 9.4:** Path Optimality Ratio for 50 hosts
Figures 9.6 and 9.7 present average end to end delay for 50, 100 hosts using designed scheme and AODV. In both 50 and 100 hosts network, designed scheme has lower end-to-end delay than AODV. MA in designed scheme discovers routes that are shorter in term of hop count, and longer-lived in term of link lifetime. Since the path has smaller number of hops, the packets will face less queuing delay, comparing to AODV. Since the path is longer-lived, fewer route breakages occur and thus data packets face less buffering delay. So designed scheme achieves better end-to-end delay than AODV.
Figure 9.6: Average Delay for 50 hosts

Figure 9.7: Average Delay for 100 hosts
**False Positive Ratio**

Comparing Figures 9.8, 9.9, 9.10, 9.11, we have observed the slight increase of the false positive ratio, which corresponds to window size 7. This is due to the fact that with the increase of the window size, the MC model characterizes that the normal behavior of routing cache changes more accurately because more history information is included. This results in a larger probability to generate false alerts, i.e., detector with larger window size is more sensitive to unexpected abnormal changes.

From the Figures 9.8, 9.10 we have observed that the classifier constructed using the feature *PCR* results in a larger false positive ratio compared to the classifier constructed using the feature *PCH*. This demonstrates that *PCH* is better than *PCR* in terms of false positive ratios. Because each entry of the DSR routing cache contains a full path to destination, *PCH* considers not only the change of the number of routes, but also the change in length of each routing entry. Therefore, *PCH* contains more information compared to *PCR*. Similar behavior is seen for Figures 9.9 and 9.11.
Figure 9.8: False Positive Ratio constructed on PCH for window size = 6

Figure 9.9: False Positive Ratio constructed on PCH for window size = 7
Figure 9.10: False Positive Ratio constructed on PCR for window size = 6

Figure 9.11: False Positive Ratio constructed on PCR for window size = 7
Mean Time to First Alarm (MTFA)

With the increase of the alarm threshold $r$, MTFA increases. Because with a larger alarm threshold, the detector needs a longer malicious trace in the current locality of frame to make the alarm signal that exceeds the alarm threshold. This results a larger MTFA.

As it is clear from Figures 9.12 and 9.13, there is increase in MTFA because when $w$ increases, the MC model characterizes the behavior more accurately. A transition which is valid in MC model with a small window size becomes invalid when $w$ becomes larger. Therefore, it needs a longer history for the alert signal to exceed $r$.

From the Figures 9.14 and 9.15, we have observed that MTFA of PCR has a larger value compared to that of PCH.

![Figure 9.12: MTFA constructed based on PCH when window size is set to 6](image)

Figure 9.13: MTFA constructed based on PCH when window size is set to 7

Figure 9.14: MTFA constructed based on PCR when window size is set to 6
**Figure 9.15:** MTFA constructed based on PCR when window size is set to 7

**Detection Ratio**

Comparing Figures 9.16 and 9.17, when $w$ increases, the detection ratio corresponding to the same mobility also increases slightly. However, there exist some exception points. The slight increase in detection ratio is due to the fact that when $w$ increases, the MC model characterize the normal behavior more accurately and thus detect more abnormal changes of the routing caches. This gives an increase in detection ratio. In the detection model, we notice that an abnormal transition which is not a valid transition with a small window size is not a valid transition with larger window size in the MC model. However, an abnormal transition that is not a valid transition with a larger $w$ may be a valid transition with a smaller $w$ in MC model. The slight decrease of the detection ratio is due to the fact that when $w$ increases, the alert signal also increases. This makes it more difficult to detect the attackers.
Comparing Figures 9.16 and 9.18 the classifier constructed using $PCH$ results in larger detection ratio compared to the classifier using $PCR$. This demonstrates that $PCH$ is better than $PCR$ in terms of detection ratio. The reason is similar when we come to the issue of false positive ratios. Normal profile constructed using $PCH$ contains more information of DSR routing caches, and is thus more accurate to characterize routing activities. The same is true if we compare Figures 9.17 and 9.19.

![Graph showing detection ratio vs. pause time](image)

**Figure 9.16:** Detection Ratio constructed on PCH for window size = 6
**Figure 9.17:** Detection Ratio constructed on PCH for window size = 7

**Figure 9.18:** Detection Ratio constructed based on PCR for window size = 6
Figure 9.19: Detection Ratio constructed based on PCR when window size is set to 7

Communication Overhead

The extra communication overhead introduced by MAIDS is caused by propagating the local alerts of intra zone hosts. We have measured the communication overhead as the number of transmission of local alerts in a given time period for one host. As shown in Figure 9.20, we can see that, when there are attacks in the network, the communication overhead is higher because of the increased number of generated local alerts. We can also see that, when there are no attacks in the network, the communication overhead decreases with the decrease of mobility. This is because when the mobility is low, local MAAs have better performance in terms of false positive ratios, thus reduces the number of alerts locally propagated in the zone.
Figure 9.20: Communication Overhead

9.8 Summary

In this chapter, we have presented MA based secure routing and intrusion detection for infrastructure less MANET. A trust based routing mechanism for MANET is also presented. The key challenges in the implementation of secure systems are reliable data delivery, trust establishment, topology changes, etc. The implemented system improves the reliability of packet forwarding over multi-hop routes in the presence of potentially malicious hosts. We have evaluated the system based on the two key metric for trust establishment namely trust and confidence.

Algorithm is evaluated based on these metrics (trust and confidence). If any host is found faulty about its behavior, information is passed to other hosts about its selfishness and the corresponding host is discarded from the route discovery process.
The metrics which are used in measuring the performance are the path optimality ratio, and end to end delay.

The path optimality ratio drops as the pause time increases, while the path optimality of AODV increases a little bit instead. Designed scheme has lower end-to-end delay than AODV. MAs can discover routes that are shorter in term of hop count by executing the algorithm for shortest path, and longer lived in term of link lifetime.

For detection of an intrusion in MANETs, we have used MAIDS. MAIDS consisting of following modules for intrusion detection: data collection module, detection engine, LACE, GACE, Intrusion Response Module. Whole network is divided into cluster heads and gateways hosts. IDA runs on gateway host and cooperates with other IDA. Each host runs IDA locally and equally cooperates with other IDAs running on other hosts.

In addition to the traditional accuracy (false positive ratio and detection ratio) measurements, MTFA and the communication overhead are also considered. We also collect the statistical features of interest, PCR and PCH, from the routing cache of mobile hosts, which reflect the mobility of the network. We compared the performance of MAIDS with respect to these above defined parameters and found that MAIDS is able to detect 60 % more intrusion than earlier designed systems and also it generates low communication overhead.

In the next chapter, secure and fault tolerant data dissemination over open networks is presented.