Cross-Feature Analysis for Detecting Ad-Hoc Routing Anomalies

Yi-an Huang * Wei Fan † Wenke Lee * Philip S. Yu †

September 20, 2002

Abstract

With the proliferation of wireless devices, mobile ad hoc networking (MANET) has become a very exciting and important technology due to its characteristics of open medium and dynamic topology among others. However, MANETs are more vulnerable than wired networks. Existing security mechanisms developed for wired networks need be redesigned for MANET. In this paper, we discuss the problem of intrusion detection in MANET. The focus of our research is on techniques for automatically constructing anomaly detection models that are capable of detecting new (or unknown) attacks. We introduce a new data mining method that uses "cross-feature analysis" to capture the inter-feature correlation patterns in normal traffic. These patterns can be used as normal profiles to detect deviation (or anomalies) caused by attacks. We have implemented our method with two well known ad-hoc routing protocols, namely, *Dynamic Source Routing* (DSR) and *Ad-hoc On-Demand Distance Vector* (AODV), and have conducted extensive experiments using the ns-2 simulator. The results show that the anomaly detection models automatically computed using our data mining method can effectively detect the anomalies caused by representative intrusions.

Keywords: Cross-Feature Analysis, MANET, Anomaly Detection, Data Mining, Intrusion Detection, Security

* College of Computing, Georgia Institute of Technology, email addresses: {yian,wenke}@cc.gatech.edu
† T. J. Watson Research Center, IBM, email addresses: {weifan,psyu}@us.ibm.com

1 Introduction

In recent years, with the rapid proliferation of wireless devices, e.g., laptop computers with wireless network interfaces, PDAs and cellular phones, the potentials and importance of mobile ad hoc networking have become apparent. A mobile ad hoc network (MANET) is formed by a group of mobile wireless nodes often without the assistance of fixed or existing network infrastructure [Per00]. The nodes must cooperate by forwarding packets so that nodes beyond radio ranges can communicate with each other. Mobile ad hoc networking enables people to keep in touch even when the "default" network, such as the Internet, is not available (for example, due to terrorist attacks). There are a number of important MANET applications, e.g., battlefield operations, emergency rescues, mobile conferencing, home and community networking, and sensor dust [Per00].

With a striking similarity of the early days of Internet research, security issues in ad hoc networking are not being adequately investigated. MANET is much more vulnerable to attacks than a wired (traditional) network due to its limited physical security, volatile network topologies, power-constrained operations, intrinsic requirement of mutual trust among all nodes in underlying protocol design and lack of centralized monitoring and management point. There are recent research efforts, e.g., [MGLB00, HPJ02, PH02, PSW+01], in providing various schemes to secure the ad hoc routing protocols. Most of these are *prevention* techniques, i.e., authentication and encryption schemes. Experience in security research in the wired environments has taught us that we need to deploy defense-in-depth or layered security mechanisms because security is a process (or a chain) that is as secure as its weakest link. For example, confidential data transmitted via an encrypted link can still be stolen from the end systems because of break-ins that exploit "weak passwords" or system software bugs. Consequently, in addition to intrusion prevention, we need a second layer of security, i.e., *intrusion detection* and response.

An intrusion detection system (IDS) analyzes network or system activities captured in audit data and uses patterns of well known attacks or normal profiles to detect the presence of potential attacks. An alert on intrusion then triggers a response, which can be a call for investigation, prompt termination or blockage of the attack activities, or damage recoveries. There are two major categories of analysis: *misuse detection* and *anomaly detection*. Misuse detection uses the "signatures" of known attacks, i.e., the patterns of attack behavior or effects, to identify a matched activity as an attack instance. By definition, misuse detection is not effective against *new* attacks, i.e., those that do not have known signatures. Anomaly detection uses established normal profiles, i.e., the expected user or system behavior, to identify any unreasonable deviation from the normal profiles as the result of an attack. Anomaly detection can be effective

against new attacks because it does not assume prior knowledge of attack patterns.

Most intrusion detection systems (IDSs) today are designed for wired networks and systems[1]. Compared with wired networks where traffic monitoring is usually accomplished at switches, routers and gateways, MANETs do not have such traffic concentrating points where an IDS can collect audit data for the entire network. Therefore, at any time, the only available audit trace on any mobile node will be limited to its local information and communication activities taking place within the transmission range, thus intrusion detection algorithms must be modified to work with partial or localized information only. Furthermore, there may not be a clear separation between normalcy and anomaly in mobile environment so that IDS may find it increasingly difficult to distinguish false alarms from real intrusions. For example, a node that sends out false routing information could be the one that has been compromised, or merely the one that is temporarily out of sync due to volatile physical movement.

MANET is still under heavy development and not many MANET-specific attacks have emerged. Therefore, we have to develop intrusion detection approaches essentially based on anomaly detection methods due to its potential capability to detect novel attacks. Aware that routing protocols of MANET are specifically designed to fit in the ad-hoc environment while other protocols on lower (physical layer, link layer, etc.) or higher (transport layer, application layer, etc.) layers are not fundamentally different from those of wired networks or infrastructure based wireless networks, our current work focuses particularly on security issues of MANET routing protocols. Our anomaly detection approach is based on data mining technologies because we are interested in *automatically* constructing detection models using logs (or trails) of system and network activity data. Given a set of training examples of the form, $\mathcal{S} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_L, y_L)\}$, for some unknown function $y = f(\mathbf{x})$, the machine learning task is to compute a *classifier* that can correctly label as many examples as possible, whether the examples are already seen (in training set) or unseen. An ideal application in intrusion detection will be to gather sufficient "normal" and "abnormal" audit data for a user or a program, then apply a classification algorithm to learn a classifier that can label or predict new unseen audit data as belonging to the normal class or the abnormal class. There has been several successful data mining based approaches in intrusion detection [LSM99, GS99]. However, most of the approaches use statistical or probabilistic analysis, the resulting models is fairly difficult to be evaluated by human experts. Some data mining models require temporal sequence from data stream, which is domain specific and highly inefficient when a large feature set is involved. The MANET routing attack detection problem, for example, involves a large feature set which prevents us from applying these approaches directly.

[1] Please refer to [ACF+00] for a comprehensive survey of intrusion detection technologies.

We have developed a new approach based on cross-feature analysis which is suitable for the MANET detection problem. We observe that strong feature correlation exists in normal behavior patterns. And such correlation can be used to detect deviations caused by abnormal (or intrusive) activities. Consider a less dynamic network, a home network composed of wirelessly connected home appliances and wireless devices (such as PDAs) held by individuals. Since obviously most appliances do not move for most of the time, a large portion of the routing fabric should be stable for a long time. If we observe one day the count of *packets being dropped* increases dramatically without obvious change in the count of *route entries being changed*, it is more likely something unusual has happened. The relationship between the *packets being dropped* and *route entries being changed* can be captured by analyzing normal patterns and be used to detect (unseen) anomalies.

More formally, in the cross-feature analysis approach, we explore correlations between each feature and all other features, i.e., try to solve the classification problem $\{f_1, f_2, \ldots, f_{i-1}, f_{i+1}, \ldots, f_L\} \rightarrow f_i$ where $\{f_1, f_2, \ldots f_L\}$ is the feature set. The anomaly detection problem can then be transformed into a set of classification sub-problems, where each sub-problem chooses a different feature as class label. The outputs of each classifier are then combined to provide an anomaly detector. In our study, we use two of the routing protocols, DSR [JM96] and AODV [PR99] and collect trace logs of normal and abnormal data in the Network Simulator ns-2 [FeV00]. Our experiment results show that the anomaly detection models, trained on the normal traces using our data mining approaches, can identify different routing anomalies very effectively.

The rest of the paper is organized as follows. We briefly introduce several MANET routing protocols used in our experiments and analyze different threats against these protocols. We then present a cross-feature analysis framework for anomaly detection. After that, we present case studies on MANET routing attack detection problem together with discussion and findings from these experiments. The paper concludes with a summary and an outline of future work.

2 Routing in MANET

We study two popular MANET routing protocols implemented in ns-2 in our case study. These protocols are Ad-hoc On-Demand Distance Vector Routing (AODV) and Dynamic Source Routing (DSR). There are other MANET routing protocols such as ZRP [HP00], OLSR [CJL$^+$01], etc. We consider the above two protocols because they are already implemented in ns-2 and have been intensively studied in recent research because of their simplicity and competitive

performance under high load and mobility.

2.1 DSR

DSR uses source routing to deliver packets through MANET. That is, the sender of a data packet finds a source route (i.e., a full path from the sender to the receiver) and includes it in the packet header. The intermediate nodes use this information to determine whether they should accept a packet and where to forward it. The protocol operates on two mechanisms: route discovery and route maintenance. Route discovery is used when the packet sender has not yet known the correct path to the packet destination. It works by broadcasting a `ROUTE REQUEST` throughout the network in a controlled manner until it is answered by a `ROUTE REPLY` from either the destination itself or an intermediate node that knows a path to it. For better performance, the source and intermediate routes save the route information in cache for future use. Furthermore, intermediate nodes can also learn new routes by eavesdropping to other Route Discovery messages taken place in the neighborhood. Finally, route maintenance mechanism is used to notify source and potentially trigger new route discovery events when changes in the network topology invalidate a cached route.

2.2 AODV

AODV, or Ad hoc On-demand Distance Vector, is another on-demand protocol since its route discovery process is also reactive on an *as needed* basis. It differs from DSR in that it does not utilize source routes and instead use a route table in all nodes with one entry per reachable destination instead. That is how the protocol has its name since it borrows ideas from distance vector based algorithms in wired networks and extends them to MANETs with a new on-demand feature added. A route table is used to record all reachable nodes in the network with following information, the next hop, the distance and a sequence number. The protocol looks up the route table to reply `ROUTE REQUEST`s and subsequently forward data packets to the proper next hop until the destination is reached. Readers interested in the difference between of these routing protocols can refer to [PRDM01], where an extensive study is presented about measuring and comparing routing performance and requirements between AODV and DSR.

2.3 Attacks

We classify attacks on MANET routing protocols into the following two categories, based on the underlying routing functionality.

- **Route logic compromise**: This type of attacks involves those where incorrect routing control messages are injected into the network to subvert or damage route fabrics. They can be further divided into external attacks and internal attacks. External attacks are generated from outside the network, while internal attacks are initiated by one or more of internal compromised nodes. A few typical routing attacks in ad hoc networks are demonstrated below.

  1. *Black hole*

     Black hole is a type of routing attack where a malicious node advertise itself as having the shortest path to all nodes in the environment. It can be used as a denial-of-service attack where it can drop the packets later. Or it can be followed by traffic monitoring and analysis to find activity patterns of each node. Sometimes it becomes the first step of a man-in-the-middle attack.

  2. *Update storm*

     The malicious node deliberately floods the whole network with meaningless route discovery messages or `ROUTE REPLY` messages. The purpose is to exhaust the network bandwidth and effectively paralyze the network.

- **Traffic distortion**: This type of attacks can snoop network traffic, manipulate or corrupt packet header or contents, block certain types of traffic or replay transmissions for some malicious purposes. Following are a few examples,

  1. *Packet dropping*

     Packet dropping does not actively change routing behavior as Black hole does. It simply drops data or route packets when it feels necessary. Based on the frequency and selectiveness, it has the following popular variations. A **random dropping** attack drops packets randomly. A **constant dropping** attack drops packets all the time. A **periodic dropping** drops packets periodically to escape from being suspected. A **selective dropping** attack drops packets based on its destination or some other characteristics. It can be introduced to specifically create a partitioning of the entire network [WVW00].

Attackers who do packet dropping can have different motivations. It can simply show selfishness expecting to save power, or intentionally prevent someone else from getting proper service. Some literature [BH02] claims that selfishness is human nature and should be stimulated (or rewarded) to forward packets instead of being punished for not to do so. In our current research, however, we regard both types of behavior as anomalies and treat them equally.

2. *Identity impersonation*

   Attackers can impersonate another user to achieve various malicious goals. In a networked environment, it is important to correctly attribute user behavior with proper user identity. Pointing to an innocent individual as the culprit can be even worse than not finding any identity responsible at all. In particular, IP and MAC (Medium Access Control) addresses can be used as identification purposes. Unfortunately, these identities are easy to be forged during the transmission of data packets on network or link layers if the underlying communication channel is not encrypted.

The advantage of the above attack classification system is that since the functionalities of a router in any routing protocols generally involve (a) establishing route path for future packet delivery, and (b) forwarding data packets based on established routes. Eventually, all attacks on routing protocols accomplish their goals by breaking at least one of the routing functionalities. Therefore, by collecting and analyzing proper routing and traffic related measures, we expect to detect possible attacks to the routing protocols. Note that the two types of attacks are not exclusive. It is not difficult to fabricate intrusions with combined attacks from both categories.

3 Cross-Feature Analysis

The basic idea of a **cross-feature analysis** framework is to explore the correlation between one feature and all the other features, i.e., try to solve the classification problem $\{f_1, f_2, \ldots, f_{i-1}, f_{i+1}, \ldots, f_L\} \rightarrow f_i$ where $\{f_1, f_2, \ldots, f_L\}$ is the feature vector. A basic assumption for anomaly detection is that normal and abnormal events should be able to separate from each other based on their corresponding feature vectors. In other words, given a feature vector, we can tell whether the related event is normal or not without ambiguity. This assumption is reasonable since otherwise the feature set is not sufficient and must be redefined. Under this assumption, we can name a feature vector related to a normal event a **normal vector**, for short. Similarly, we call a feature vector not related to any normal events an **abnormal vector**. We re-formulate the problem as follows. We assume here that all feature values are discrete.

A generalized extension will be discussed later in the paper. For all normal vectors, we choose one feature as the target to classify (which is called the **labeled feature**), and then compute a model using all normal vectors to predict the chosen target feature value based on remaining features. In other words, we train a classification model $\mathcal{C}_i$: $\{f_1, \ldots, f_{i-1}, f_{i+1}, \ldots, f_L\} \rightarrow \{f_i\}$. For normal events, the prediction by C_i is very likely to be the same as the true value of the feature; however, for anomalies, this prediction is likely to be different. The reason is that C_i is trained from normal data, and their feature distribution and pattern are assumed to be different from those of anomalies. This implies that when normal vectors are tested against $\mathcal{C}_i$, it has a higher probability for the true and predicted values of f_i to match. Such probability is significantly lower for abnormal vectors[2]. Therefore, by evaluating the degree of result matching, we are more likely to find difference between normal and abnormal patterns. We name the model defined above a **sub-model with respect to** f_i. Obviously, relying on one sub-model with respect to one labeled feature is insufficient as we haven't considered the correlation among other features yet. Therefore the model building process is repeated for every feature and up to L sub-models are trained. Once done, we have accomplished the first step of our cross-feature analysis approach, i.e. the training procedure, which is summarized in Algorithm 1.

Data : feature vectors of training data $f_1, \ldots, f_L$
Result : classifiers $\mathcal{C}_1, \ldots, \mathcal{C}_L$
begin
  $\forall i$, train $\mathcal{C}_i : \{f_1, \ldots, f_{i-1}, f_{i+1}, \ldots, f_L\} \rightarrow f_i$;
  return $\mathcal{C}_1, \ldots, \mathcal{C}_L$;
end

Algorithm 1: Cross-Feature Analysis: Training Procedure

To generalize the framework to continuous features or discrete features with an infinite value space (e.g., the integer set), we should keep in mind that they cannot be used directly as class labels since only discrete (nominal) values are accepted. We can either discretize it or use multiple linear regression. With multiple linear regression, we use log distance, $|log(\frac{C_i(x)}{f_i(x)})|$, to measure the difference of prediction from true value, where $C_i(x)$ is the predicted value from sub-model with respect to f_i.

Once all sub-models have been trained, we can analyze trace logs as follows. When an event is analyzed, we apply the feature vector to all sub-models, and count the number of models whose predictions matches the true value of the labeled feature. The count is then divided by L, so that the output, which is called the **average match count** throughout the paper, is normalized into the range of [0, 1]. We do not need all sub-models to be matched. In fact, what we need is a *decision threshold*. An event is classified as anomaly if and only if the *average match count* is below

[2] In a separate paper under preparation, we formally and rigorously explain and discuss the algorithm and prove the relationship between cross-feature analysis and optimal Bayesian reasoning. One important distinction is that cross-feature analysis is not statistical correlation analysis. Correlation analysis measures linear correlation between every pair of features. However, cross-feature analysis studies the predictability of one feature via multiple features.

the threshold. We find it hard to develop a solution to determine the decision threshold with perfect accuracy for a general anomaly detection problem from sub-models directly. In practice, a small value of false alarm rate is often allowed. We can determine the threshold by computing *average match count* values on all normal events, and using a lower bound of output values with certain confidence level (which is one minus false alarm rate). As a summary, Algorithm 2 lists the straw man version of the test procedure. For convenience, $f_i(x)$ denotes the value of feature f_i belonging to event x. $[\![\pi]\!]$ returns 1 if the predicate π is true.

Data : classifiers $\mathcal{C}_1, \ldots, \mathcal{C}_L$, event $x = (f_1, \ldots, f_L)$, decision threshold θ
Result : either normal or anomaly
begin
  AvgMatchCount $\leftarrow \sum_i [\![C_i(x) = f_i(x)]\!]/L$;
  if *AvgMatchCount* $\geq \theta$ **then** return "normal";
  else return "anomaly";
end

Algorithm 2: Cross-Feature Analysis: Testing Procedure Using *Average Match Count*

One straightforward improvement to the straw man algorithm is to use probability instead of the 0-1 count. Most inductive learners (e.g., decision trees, induction rules, naive Bayes, etc.) can output a probability for every possible class. Assume that $p(f_i(x)|x)$ is the estimated probability for the true class of the labeled feature, we similarly define **average probability** as the average probabilities for the true classes over all classifiers. This approach can improve detection accuracy since a sub-model should be preferred where the labeled feature has stronger confidence to appear in normal data. The previous algorithm using match count can be treated as a special case of the probability based approach under the assumption that the predicted class is the only valid class and hence has a probability of 1.0, so the probability for the true class is either 1 (when the rule matches) or 0 (otherwise). Therefore, the optimization can be considered as a weighted version of Algorithm 2. The modified version is shown in Algorithm 3.

Data : classifiers $\mathcal{C}_1, \ldots, \mathcal{C}_L$, event $x = (f_1, \ldots, f_L)$, decision threshold θ
Result : either normal or anomaly
begin
  AvgProbability $\leftarrow \sum_i p(f_i(x)|x)/L$;
  if *AvgProbability* $\geq \theta$ **then** return "normal";
  else return "anomaly";
end

Algorithm 3: Cross-Feature Analysis: Testing Procedure Using *Average Probability*

We now discuss in detail how probability function can be calculated in a few popular classification algorithms. Decision tree learners (such as C4.5 [Qui93]) uses divide-conquer strategies to group examples with the same feature values until it reaches the leaves of the tree where it cannot distinguish the examples any further. Suppose that n is the total number of examples in a leaf node and n_i is the number of examples with class label ℓ_i in the same leaf.

$p(\ell_i|x) = \frac{n_i}{n}$ is the probability that x is an instance of class ℓ_i. We calculate probability in a similar way for decision rule classifiers, e.g. RIPPER [Coh95]. For naive Bayes classifiers, assume that a_j's are the attributes of x, $p(\ell_i)$ is the prior probability or frequency of class ℓ_i in the training data, and $p(a_j|\ell_i)$ is the prior probability to observe feature attribute value a_j given class label ℓ_i, then the score $n(\ell_i|x)$ for class label ℓ_i is: $n(\ell_i|x) = p(\ell_i) \prod_j p(a_j|\ell_i)$ and the probability is calculated on the basis of $n(\ell_i|x)$ as $p(\ell_i|x) = \frac{n(\ell_i|x)}{\sum_k n(\ell_k|x)}$.

An Illustrative Example We use a simplified example to demonstrate our framework. Consider an ad-hoc network organized with two nodes. Packets can only be sent from one end to the other if they are within each other's transmission range. We define the following three features. 1) Is the *other node reachable*? 2) Is there any *packet delivered during last 5 seconds*, and 3) is there any *packet cached for delivery during last 5 seconds*? For simplicity, we assume all features are binary valued, i.e., either **True** or **False**. All normal events are enumerated in Table 1. We then construct three sub-models with respect to each feature, shown in Table 2. The "Probability" columns here denote the probability associated with predicted classes. We use an illustrative classifier in this example that works as follows. If only one class is seen in all normal events where other non-labeled features have been assigned with a particular set of values, the single class is selected as the predicted class with the associated probability of 1.0. If both classes are seen, label *True* is always selected with the associated probability of 0.5. If none are seen (which means the combination of the other two feature values never appears in normal data), we select the label which appears more in other rules, with the associated probability of 0.5. To compute the probability for the true class, we use the probability associated with the predicted class if it matches, or one minus the associated probability if it does not. For example, in the situation when a route is viable, no data is cached and no data is therefore delivered is a normal case. We apply the corresponding feature vector, $\{True, False, False\}$, into all three sub-models and all match the predicted classes. But the first sub-model with respect to the feature "Reachable?" has a probability of 0.5 only, which is obvious since when no data is delivered, it does not matter whether the route is up or not. The average match count is then calculated as $(1 + 1 + 1)/3 = 1$, and the average probability is $(1 + 1 + 0.5)/3 = 0.83$. Suppose we use a threshold of 0.5, then both values tell that the event is normal, which is right. A complete list of the *average match counts* and *average probabilities* for all possible events (both normal and abnormal) is shown in Table 3. The results clearly show that given a threshold of 0.5, both Algorithm 2 and 3 work well to separate normal and abnormal events, while Algorithm 3 works better as it achieves perfect accuracy (Algorithm 2 has one false alarm with the input $\{False, False, False\}$).

Table 1: Complete set of normal events in the 2-node network example

Reachable?	Delivered?	Cached?
True	True	True
True	False	False
False	False	True
False	False	False

Table 2: Sub-models in the 2-node network example

Delivered?	Cached?	Reachable?	Probability
True	True	True	1.0
False	False	True	0.5
False	True	False	1.0
True	False	True	0.5

(a) Sub-model with respect to 'Reachable?'

Reachable?	Cached?	Delivered?	Probability
True	True	True	1.0
True	False	False	1.0
False	True	False	1.0
False	False	False	1.0

(b) Sub-model with respect to 'Delivered?'

Reachable?	Delivered?	Cached?	Probability
True	True	True	1.0
True	False	False	1.0
False	False	True	0.5
False	True	True	0.5

(c) Sub-model with respect to 'Cached?'

4 Experimental Studies

In order to study how our data mining framework can be used to construct anomaly detection models for MANET routing, we have conducted the following experiments using the ns-2 simulator.

4.1 Experiment Setup

We use the Network Simulator ns-2 to run MANET simulations. ns-2 is a simulation project developed in ISI/USC with support from DARPA and NSF. It is one of the most widely used network simulators for wired and wireless networks.

Table 3: Complete set of both normal and abnormal events in the 2-node network example

Reachable?	Delivered?	Cached?	Class	Average match count	Average probability
True	True	True	Normal	1	1
True	False	False	Normal	1	0.83
False	False	True	Normal	1	0.83
False	False	False	Normal	0.33	0.67
True	True	False	Abnormal	0.33	0.17
True	False	True	Abnormal	0	0
False	True	True	Abnormal	0.33	0.17
False	True	False	Abnormal	0	0.33

Parameter Selection We apply the random way-point model in ns-2 to emulate node mobility patterns with a topology of 1000m by 1000m. We use both TCP and UDP/CBR (Constant Bit Rate) as underlying transport protocols, and different protocols are used separately in different experiments. The maximum number of connections is set to be 100, traffic rate is 0.25, the pause time between movements is 10s and the maximum movement speed is 20.0m/s. These settings are typical ad-hoc settings with adequate mobility and data load overhead, and are used in all our experiments. We use one running trace of normal data as training set. For evaluation purposes, we use several other traces with normal data only, and a few traces composed with black hole and packet dropping attacks, started at 2500s and 5000s respectively. All traces have a run time of 10000 seconds with route statistics logged every 5 seconds (see Table 4).

Feature Construction The features constructed in our experiments belong to two categories, non-traffic related and traffic related. All non-traffic related features are detailed in Table 4 and the meaning of each feature is further explained in the "Notes" column. These features capture the basic view of network topology and route fabric update frequency. In addition, we calculate absolute velocity from mobility traces directly in simulation.

All traffic related features are collected based the following considerations. Packets come from different layers and different sources. For example, it can be a TCP data packet delivered from the originator where the feature is collected. It can also be a route control message packet (for instance, a `ROUTE REQUEST` message, used in AODV and DSR), which is being forwarded at the observed node. We can then define the first two aspects of a traffic feature as, *packet type*, which can be data specific and route specific (including different route messages used in AODV and DSR), and *flow direction*, which can take one of the following values, `received` (observed at destinations), `sent` (observed at sources), `forwarded` (observed at intermediate routers) or `dropped` (observed at routers where no route is available for the packet). We do, however, exclude the combination that data packets can be forwarded or dropped since it would never appear in a real ns-2 trace log. Routing protocols in MANET usually encapsulate data packets by

Table 4: Feature Set I: Topology and route related features

Features	Notes
time	ignored in classification, only used for reference
absolute velocity	
route add count	routes newly added by route discovery
route removal count	stale routes being removed
route find count	routes found in cache and do not need to re-discovery
route notice count	routes noticed to cache eavesdropped from somewhere else
route repair count	broken routes currently under repair
total route change	
average route length	

Table 5: Feature Set II: Traffic related feature dimensions

Dimension	Values
Packet type	data, route (all), `ROUTE REQUEST`, `ROUTE REPLY`, `ROUTE ERROR` and `HELLO` messages
Flow direction	received, sent, forwarded and dropped
Sampling periods	5, 60 and 900 seconds
Statistics measures	count and standard deviation of inter-packet intervals

adding particular headers with routing information at the source node and unpack them at the destination. Therefore all activities (including forwarding and dropping) during the transmission process only involve "route" packets. Also, we need to evaluate both short-term and long-term traffic patterns. In our experiments, we sample data in three predetermined *sampling periods*, 5 seconds, 1 minute and 15 minutes. Finally, for each traffic pattern, we choose two typical *statistics measures* widely used in literature, namely, the packet count and the standard deviation of inter-packet intervals. Overall, a traffic feature can be defined as a vector $<$ packet type, flow direction, sampling periods, statistics measures $>$. All dimensions and allowed values for each dimension are defined in Table 5. For instance, the feature to compute the standard deviation of inter-packet intervals of received `ROUTE REQUEST` packets every 5 seconds can be encoded as $< 2, 0, 0, 1 >$. Overall, we have $(6 \times 4 - 2) \times 3 \times 2 = 132$ traffic features, where 6, 4, 3, 2 are the number of packet types, flow directions, sampling periods and statistics measures, respectively.

For all continuous features or discrete features with infinite value space, we discretize them using a frequency-bucket scheme. We divide the value space of a continuous feature into a fixed number of continuous ranges (buckets), so that the frequencies of occurrences of feature values dropped in all buckets are equal. Then a continuous feature can be replaced by the index of its corresponding bucket. This approach guarantees that the chances of appearance of all possible labels (after discretization) in a feature are approximately the same. A pre-filtering process using a small random subset of normal vectors is necessary to retrieve the frequency distribution of all continuous features. In our experiments, we choose the bucket number to be 5.

Intrusion Simulation We choose to implement the following intrusion scripts to study the problem of anomaly detection in MANETs.

- Black hole attack

  It is implemented by the following means. For DSR, a compromised host H_c broadcasts bogus `ROUTE REQUEST` messages with selected source and destination and a fake sequence number with maximum allowed value. The source route field in the fabricated `ROUTE REQUEST` specifies a one-hop route from the source to H_c as if the host were the immediate neighbor who forwarded the source's first `REQUEST`. Those messages are not harmful themselves except for the overhead due to flooding. Instead, the attack is really accomplished by the side effect that a neighbor overhearing such an bogus `REQUEST` message would take the source route recorded in the message, reverse it, mistakenly assume the reversed source route could be a better route to the source and override existing valid route(s) to the source by the faked route. If the attacker generates multiple bogus `REQUEST` messages where each message specifies a different source and all sources (except for the attacker itself) are covered, then all traffic flows, no matter where their destinations could be, will be forwarded to the compromised node. This is very similar in concept to an astronomical object, i.e., a region in space in which the pull of gravity is so strong that nothing can escape, as a corollary to Einstein's general theory of relativity.

  For AODV, we choose same destination as well as the source in each bogus message, which is allowed in this protocol. Similarly, the attack script fabricates a source sequence number to be the maximal allowed value, and claims that the compromised host is the next hop from the source node.

- Selective packet dropping attack

  Under this attack, packets are dropped based on its destination on the compromised host. The destination is specified as a parameter to the attack script.

For each type of intrusion, we don't actually turn on the intrusion behavior all the time as otherwise it could become an obvious target to be detected. Instead, we introduce a simple on-off model where intrusion sessions are inserted periodically. For the sake of simplicity, we assume the duration of each intrusion session and the gap between two adjacent intrusion sessions are same. The value of duration is specified as a script parameter. We summarize the implemented intrusions in Table 6.

Table 6: Simulated MANET intrusions

Attack Script	Attack Description	Script Parameters
Black hole	Generate bogus shortest route to all nodes and absorb all traffic nearby	duration
Selective Packet Dropping	Drop packets to specific destination	duration, destination

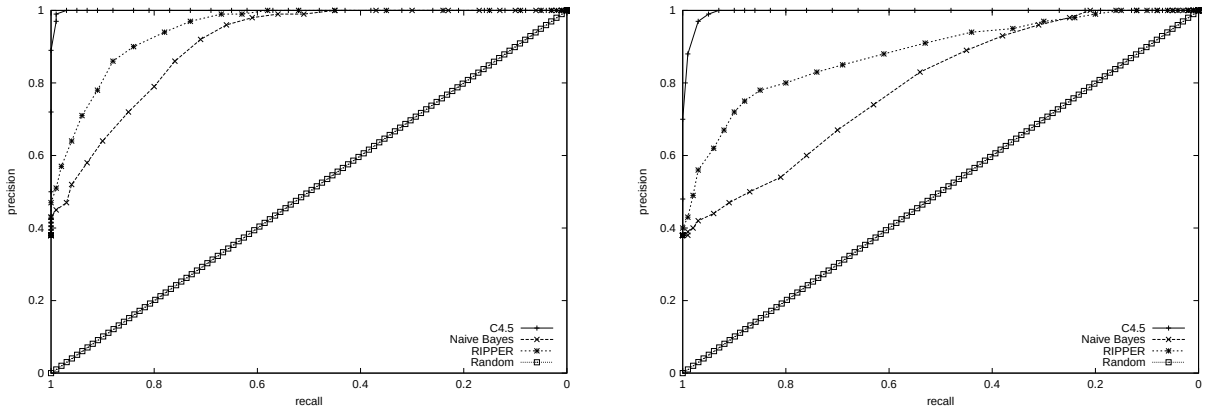
4.2 Experimental Results

With a combination of AODV vs. DSR and TCP vs. UDP/CBR, we use all four scenarios in our experiments. We show results on all scenarios to compare the performance of our approach when different routing and transport protocols are used. Also, we choose several classifiers using different classification algorithms for evaluation purposes. These classifiers are C4.5 [Qui93], a decision tree classifier, RIPPER [Coh95], a rule based classifier, and NBC[3], a naive Bayes classifier. Note that all results discussed in the paper are collected on one node only, for brevity. Similar results and performance have been verified on other nodes of the simulated network throughout our experiments.

Although we suggested earlier that a decision threshold can be computed by calculating the lower bound of output values from normal events, we here show alternative results and performance when different values of threshold are used by using a recall-precision curve and explain how an optimal threshold values can be achieved empirically.

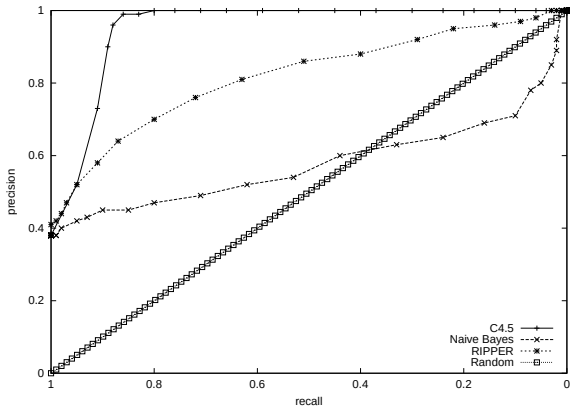
Recall is a measure of the fraction of known positive examples that are correctly classified. Precision is a measure of the fraction of the classified positive examples that are truly positive. If I denotes intrusions and A denotes alarms (i.e., classified as positive), then recall rate is $p(A|I)$ and the precision rate is $p(I|A)$. The better (higher) a recall rate is, the more abnormal instances are captured. The better (higher) a precision rate is, the more chance that alarms are really intrusions. It is hard to achieve perfect results for both measures at the same time in practice. Some trade-off has to be made based on other criteria. In a recall-precision curve, the x-axis is the recall rate and the y-axis is the precision rate. In our experiments, we obtain various operation points in the recall-precision space by varying decision thresholds. A larger (smaller) threshold implies more (less) examples are classified as positive, then the recall rate may go up (down) since more (less) anomalies have chance to be classified correctly. On the other hand, the precision rate may go down (up) since more (less) normal events are also classified as alarms. The 45-degree diagonal of the recall-precision curve is the result of "random guess", where anomalies and normal connections have equal probability to be classified as anomalies. The closer the curve follows the left border and then the top border, the more accurate a proposed method is. In practice, recall and precision carry different costs. The choice of decision threshold is to minimize the total loss.

[3]The source code is available at http://fuzzy.cs.uni-magdeburg.de/∼borgelt/software.html.

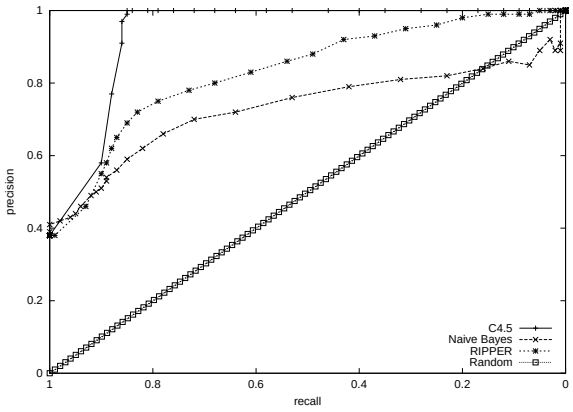


(a) AODV/UDP

(b) AODV/TCP



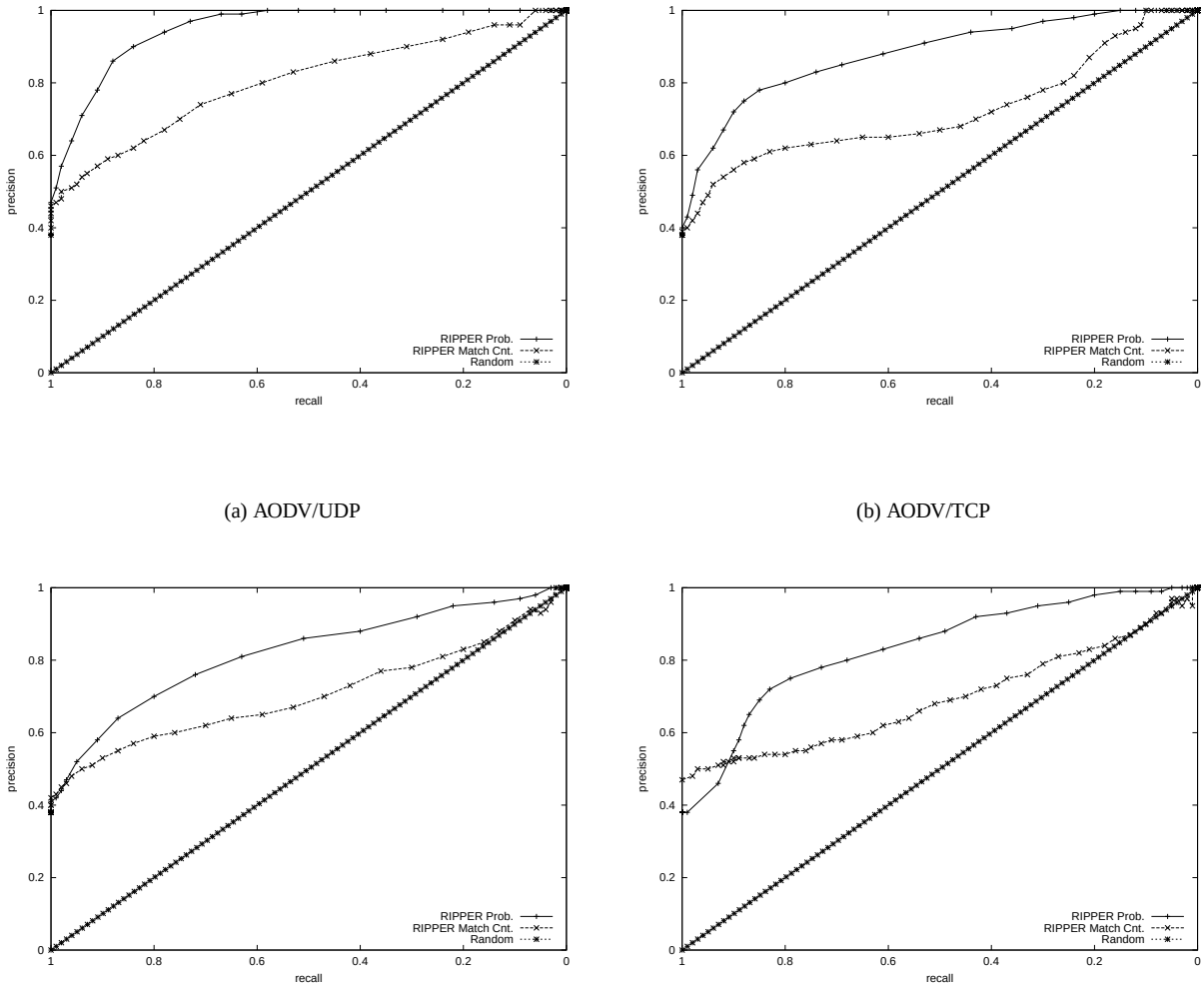
(c) DSR/UDP



(d) DSR/TCP

Figure 1: Recall-Precision curves using average probability with various classifiers

The recall-precision curves with three classifiers (C4.5, RIPPER and NBC) are shown in Figure 1 based on average probability measures. We find that performance from the three classifiers is quite different. Quantitatively, we can use the area between the curve and the "random guess" diagonal line, or the Area Under the Curve (AUC), to evaluate the accuracy of corresponding classifier. Using this measure, C4.5 shows almost perfect performance with its curves very close to the left and top borders, far better than the other two classifiers. With experiments using average match count as the measure (not shown), we observe that RIPPER improves performance dramatically when we use average probability instead of average match count. The comparison results, using recall-precision curves, are demonstrated in Figure 2. Similar performance improvement, however, does not appear to be very obvious when C4.5 or NBC are used. Consequently, in Figure 1, RIPPER is shown as the second best classifier. A further observation we have is that results from AODV are significantly better than those from DSR. For instance, in the C4.5 case, the optimal point for AODV/TCP is (0.99, 0.97) but it is (0.86, 0.93) for DSR/TCP, where a simplified criterion is used that optimal point



(a) AODV/UDP

(b) AODV/TCP

(c) DSR/UDP

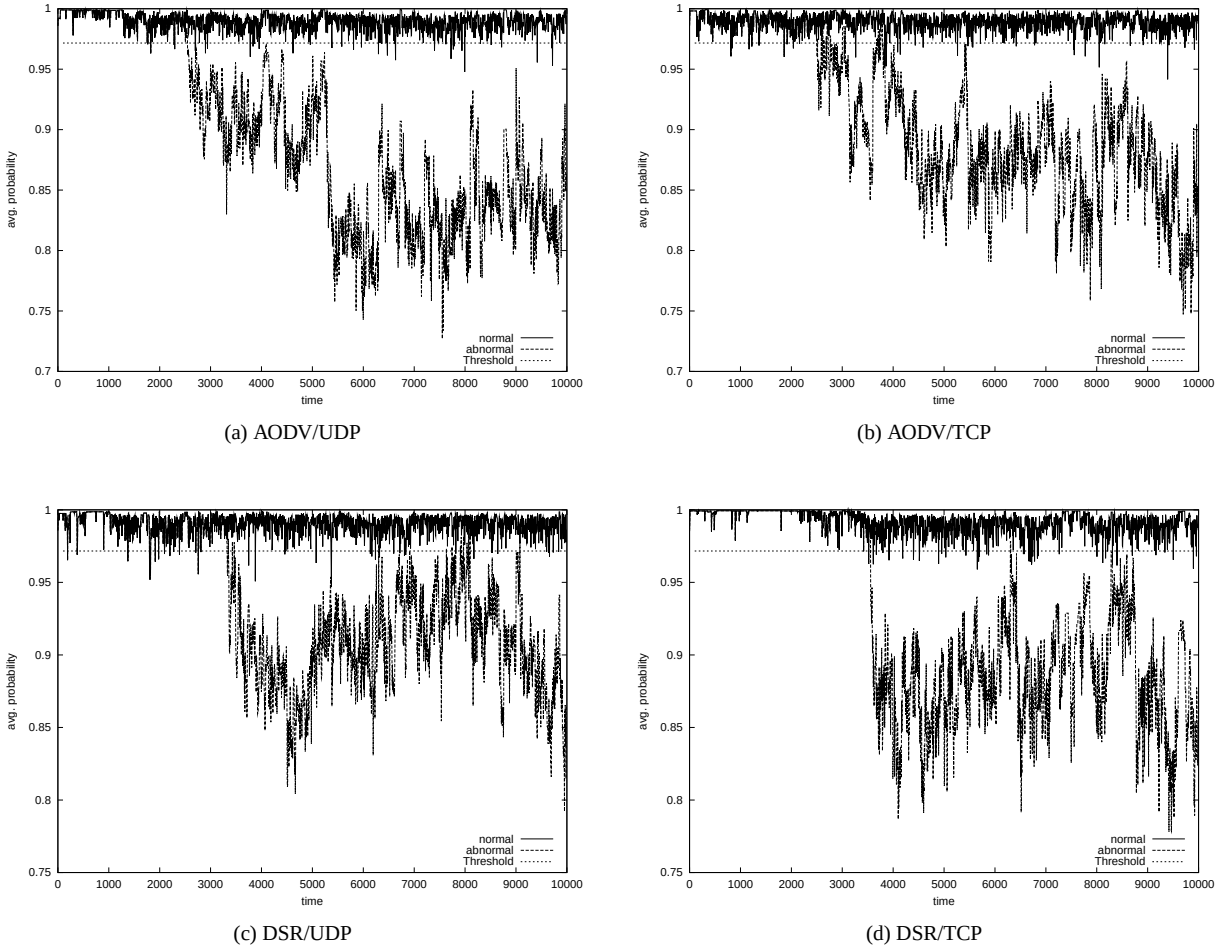
(d) DSR/TCP

Figure 2: Comparing average match count vs. average probability with RIPPER

occurs with the closest distance to (1, 1).

We also show the outputs of average probability from both normal and abnormal traces in Figure 3 with C4.5 for all four scenarios. Since multiple traces have been studied in each test condition (normal vs. abnormal), we use the averaged outcome of the same test condition in the figures. We observe that identical curves for both normal and abnormal traces are shown during the initial 2500 seconds. The reason is the setup of our traces specifies start time of intrusions is at 2500s. After that, it can be identified that normal traces have almost flat curves, which implies that our model successfully captures most of normal events. On the other hand, abnormal traces show oscillating curves with

results ranging from 0.75 to 0.95 for AODV, or 0.8 to 0.95 for DSR, indicating the abnormal activities are detected.



(a) AODV/UDP

(b) AODV/TCP

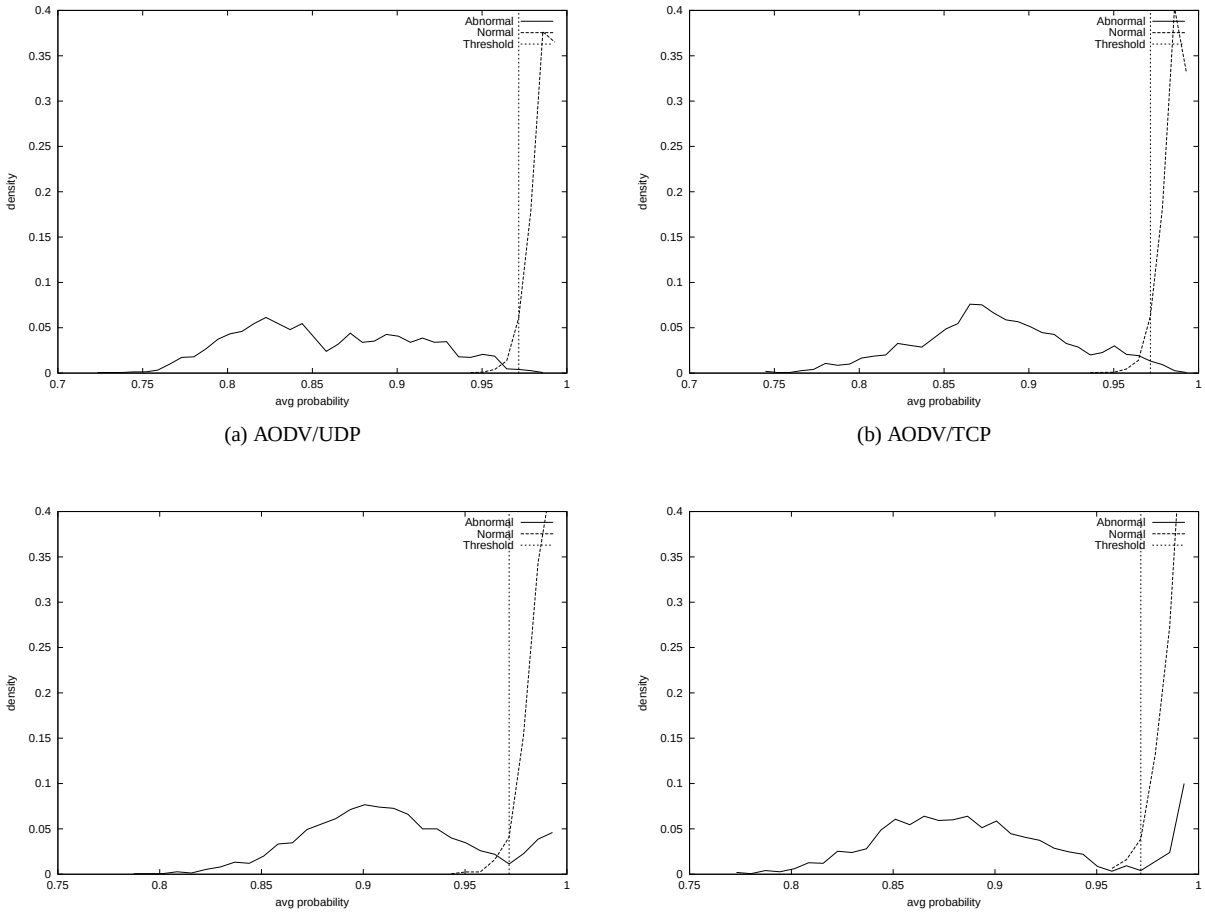
(c) DSR/UDP

(d) DSR/TCP

Figure 3: Average probability: normal vs. abnormal traces with C4.5

We then plot the output average probability values using density distribution curves in Figure 4. The threshold is represented by a vertical line. The right of the threshold line is considered normal. The other side is abnormal. The difference between AODV and DSR is further confirmed here by observing that AODV could have better accuracy than DSR since the areas under abnormal curves of DSR have bigger regions to the right of threshold lines, where the anomalies are not detected.

The results so far are from setup with mixed intrusions from different categories (routing and traffic) in the same trace. Figure 5 shows results when a different set of intrusion sessions are used. For the sake of clarity, we only show results with one of the four scenarios. We here choose the AODV/UDP scenario. Similarly, we use the classifier C4.5 only. We show output results with traces composed of only black hole attacks in Figure 5(a), and results with



(a) AODV/UDP

(b) AODV/TCP

(c) DSR/UDP

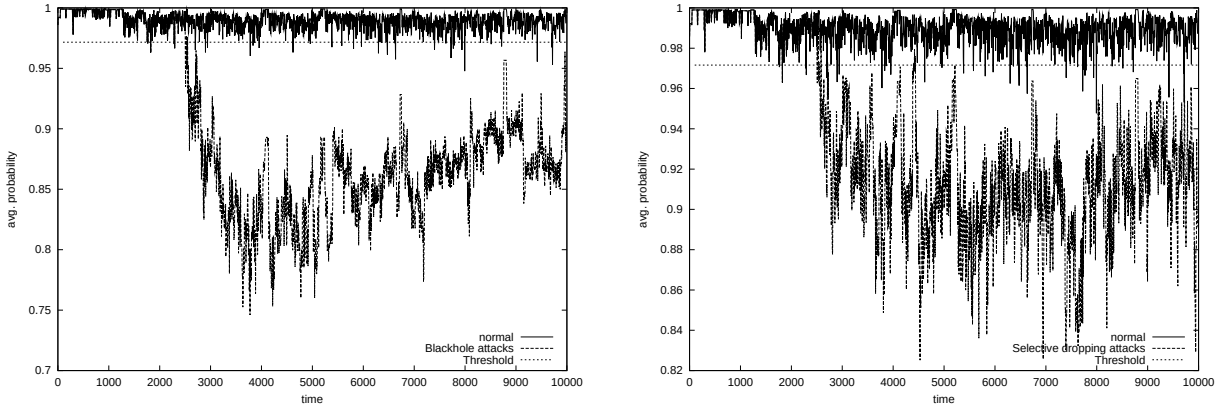
(d) DSR/TCP

Figure 4: Average probability density distribution: normal vs. abnormal traces with C4.5

only packet dropping attacks in Figure 5(b). Each trace has three intrusions started on 2500s, 5000s and 7500s respectively, all lasting for 100 seconds. Results from a mixed intrusion scenario can be read from Figure 3(a) as a contrast. From results shown in Figure 5, we can easily identify that each intrusion type shows slightly different patterns but it still can be easily separated from normal traces with the specified threshold. These results also suggest that MANET may not recover from the implemented intrusions very well. For the black hole case, we find out that the attack script we implemented broadcast bogus `ROUTE REQUEST` messages using a maximum sequence number. The effect will never be automatically rectified under current routing protocol implementations in ns-2 because routes with maximum sequence number are always considered the freshest. The dropping attack is more confusing and we are still investigating the reason behind the phenomenon. The problem of failing to completely self-healing promptly also poses a problem to intrusion detection as there is no way to figure out exactly when the intrusion actions have ended and the observed anomalies are just the lasting damages. We are planning to further investigate approaches to

correct the problem.

Finally, we redraw those curves using the density function over average probability in Figure 6. Areas under normal curve while to the left of the threshold (false alarms) and under intrusive curves while to the right of the threshold (anomalies mistakenly accepted) are both very small. Although different intrusion scenarios show different distribution, we can see that the resulting plots between normal and abnormal traces are all distinct.



(a) With black hole attacks only

(b) With packet dropping attacks only

Figure 5: Average probability: different intrusion scenarios with AODV/UDP/C4.5

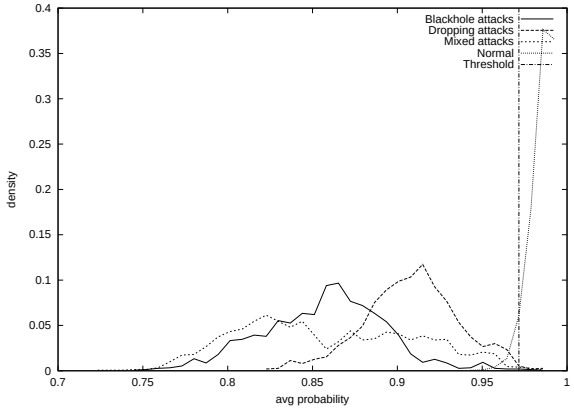


Figure 6: Average probability density distribution: different intrusion scenarios

5 Related Work

For intrusion prevention in MANET, general approaches such as key generation and management have been used in a distributed manner to ensure the authenticity and integrity of routing information. Zhou and Haas [ZH99] introduced a routing protocol independent distributed key management service. This approach uses redundancies in the network topology to provide reliable key management. The key idea is to use key sharing with a maximum threshold ratio of compromised nodes to total nodes. Stajano and Anderson [SA99] establish secure transient association between users by "imprinting" according to the analogy to ducklings acknowledging the first moving subject they see as their mother, but enabling the devices to be imprinted several times. Binkley [BT01] reported experiments on authentication of MAC and IP layers. Recently, Basagni et al. [BHBR01] used a network-wide secret key scheme to secure routing messages. Hubaux et al. [HBC01] proposed to use a PGP-like scheme to bootstrap trust relationships. Perrig et al. [PSW+01] studied the problem of authenticating broadcast in sensor networks. Their approach is to achieve asymmetry (which is required for authentication) through a delayed disclosure of symmetry keys. It requires that the sensor nodes and the base station are time synchronized. These prevention or proactive schemes are in general difficult to defend against internal attacks or passive attacks.

Researchers are starting to study intrusion detection and response technique for MANET. Zhang and Lee [ZL00] started a preliminary investigation and proposed a (high-level) fully distributed and cooperative intrusion detection and response architecture. Marti et al. [MGLB00] proposed to use "watchdog" and "pathrater" to identify nodes with routing misbehavior and to avoid such nodes in the routes used. The CONFIDANT [BB02] system has a monitor on each mobile node for observations, reputation records for first-hand and trusted second-hand observations, trust records to control trust given to received warnings, and a path manager for nodes to adapt their behavior according to reputation. However, their protocol targets specifically at the unfairness problem and the major concern is that individual node may deny forwarding packets, which restricts the protocol from being used against more general attacks.

There have been efforts on securing the routing protocols for MANET as well, e.g. [HPJ02, PH02]. Similar work in wired networks, such as [Che97, SMGLA97] usually come with large communication overhead and do not work well in MANET because of its dynamically changing network topology.

Since anomaly detection will be one of our main research tasks. We give some more background here. Early ap-

proaches [AFV95] used statistical measures of system features, e.g., CPU usage, to build normal profiles. Lane et al. recently studied machine learning approaches for modeling user behavior [LB99]. There have been studies on modeling program behavior, using system call traces and learning-based approaches [FHSL96, SBDB01] and specification-based approaches [Ko00]. Ghosh and Schwartzbard [GS99] proposed using a neural network to learn a profile of normality. Fan et al. [FMS+01] transfer a one-class problem with only normal data into a new problem with two classes by introducing artificial anomalies. To the best of our knowledge, we are the first to investigate how inter-feature correlations can help to build anomaly detection models.

6 Conclusion and Future Work

We present in this paper a new data mining approach based on cross-feature analysis for anomaly detection in MANET routing. The results show that this approach is very useful when strong inter-feature correlation can be extracted automatically in the normal system data. We find that the our resulting model is fairly easy to comprehend and can be examined by human experts.

Future Work Although the framework is studied in the background of MANET routing, we believe that it is a *general* anomaly detection approach for network intrusion problem as well as a few financial fraud detection problems where only normal data could be trusted. We are currently applying the framework to some other data sets. Initial experiments using credit card fraud detection have revealed promising results.

We are developing technologies to reduce computational cost, where fewer number of models are involved in the combination process and each model could be simplified with a reduced feature set. We are currently studying approaches based on both correlation analysis and factor analysis.

References

[ACF+00] J. Allen, A. Christie, W. Fithen, J. McHugh, J. Pickel, and E. Stoner. State of the practice of intrusion detection technologies. Technical Report CMU/SEI-99TR-028, Carnegie Mellon University, 2000.

[AFV95] D. Anderson, T. Frivold, and A. Valdes. Next-generation intrusion detection expert system (NIDES): A summary. Technical Report SRI-CSL-95-07, Computer Science Laboratory, SRI International, Menlo Park, California, May 1995.

[BB02] S. Buchegger and J. L. Boudec. Nodes bearing grudges: Towards routing security, fairness, and robustness in mobile ad hoc networks. In *Proceedings of the Tenth Euromicro Workshop on Parallel, Distributed and Network-based Processing*, pages 403 – 410, Canary Islands, Spain, January 2002. IEEE Computer Society.

[BH02] L. Buttyan and J. P. Hubaux. Stimulating cooperation in self-organizing mobile ad hoc networks. *ACM Journal for Mobile Networks (MONET), special issue on Mobile Ad Hoc Networks*, 2002.

[BHBR01] S. Basagni, K. Herrin, D. Bruschi, and E. Rosti. Secure pebblenets. In *Proceedings of the 2001 ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2001)*, Long Beach, CA, October 2001.

[BT01] J. Binkley and W. Trost. Authenticated ad hoc routing at the link layer for mobile systems. *Wireless Networks*, 7(2):139–145, 2001.

[Che97] S. Cheung. An effi cient message authentication scheme for link state routing. In *Proceedings of the 13th Annual Computer Security Applications Conference*, 1997.

[CJL+01] T. Clausen, P. Jacquet, A. Laouiti, P. Muhlethaler, and a. Qayyum et L. Viennot. Optimized link state routing protocol. In *Proceedings of IEEE International Multi-Topic Conference(INMIC)*, Pakistan, 2001.

[Coh95] W. W. Cohen. Fast effective rule induction. In *Machine Learning: the 12th International Conference*, Lake Taho, CA, 1995. Morgan Kaufmann.

[FeV00] K. Fall and e Varadhan. *The ns Manual (formerly ns Notes and Documentation)*, 2000. Online reference: http://www.isi.edu/nsnam/ns/ns-documentation.html.

[FHSL96] S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff. A sense of self for Unix processes. In *Proceedings of the 1996 IEEE Symposium on Security and Privacy*, pages 120–128, Los Alamitos, CA, 1996. IEEE Computer Society Press.

[FMS+01] W. Fan, M. Miller, S. Stolfo, W. Lee, and P. Chan. Using artifi cial anomalies to detect unknown and known network intrusions. In *IEEE International Conference on Data Mining (ICDM 2001)*, San Jose, CA, November 2001.

[GS99] A. K. Ghosh and A. Schwartzbard. A study in using neural networks for anomaly and misuse detection. In *8th USENIX Security Symposium*, 1999.

[HBC01] J. Hubaux, L. Buttyan, and S. Capkun. The quest for security in mobile ad hoc networks. In *Proceeding of the ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, Long Beach, CA, 2001.

[HP00] Z.J. Haas and M. R. Pearlman. The zone routing protocol (ZRP) for ad hoc networks. Internet draft draft-ietf-manet-zone-zrp-04.txt, expired 2003, July 2000.

[HPJ02] Y. Hu, A. Perrig, and D. B. Johnson. Ariadne: A secure on-demand routing protocol for ad hoc networks. In *Proceedings of the Eighth Annual International Conference on Mobile Computing and Networking* (MobiCom 2002), September 2002.

[JM96] D. B. Johnson and D. A. Maltz. Dynamic source routing in ad hoc wireless networks. In Tomasz Imielinski and Hank Korth, editors, *Mobile Computing*, pages 153–181. Kluwer Academic Publishers, 1996.

[Ko00] C. Ko. Logic induction of valid behavior specifi cations for intrusion detection. In *Proceedings of the 2000 IEEE Symposium on Security and Privacy*, May 2000.

[LB99] T. Lane and C. E. Brodley. Temporal sequence learning and data reduction for anomaly detection. *ACM Transactions on Information and System Security*, 2(3):295–331, 1999.

[LSM99] W. Lee, S. J. Stolfo, and K. W. Mok. A data mining framework for building intrusion detection models. In *IEEE Symposium on Security and Privacy*, pages 120–132, 1999.

[MGLB00] S. Marti, T. J. Giuli, K. Lai, and M. Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Mobile Computing and Networking*, pages 255–265, 2000.

[Per00] C. E. Perkins. Ad hoc networking: An introduction. In C. E. Perkins, editor, *Ad Hoc Networking*. Addison-Wesley, 2000.

[PH02] P. Papadimitratos and Z. J. Hass. Secure routing for mobile ad hoc networks. In *SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS)*, San Antonio, TX, January 2002.

[PR99] C. E. Perkins and E. M. Royer. Ad hoc on-demand distance vector routing. In *2nd IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, New Orleans, LA, February 1999.

[PRDM01] C.E. Perkins, E. M. Royer, S. R. Das, and M. K. Marina. Performance comparison of two on-demand routing protocols for ad hoc networks. *IEEE Personal Communications Magazine special issue on Ad hoc Networking*, pages 16–28, February 2001.

[PSW+01] A. Perrig, R. Szewczyk, V. Wen, D. E. Culler, and J. D. Tygar. SPINS: security protocols for sensor networks. In *Mobile Computing and Networking*, pages 189–199, 2001.

[Qui93] J. R. Quinlan. *C4.5: Programs for machine learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[SA99] F. Stajano and R. Anderson. The resurrecting duckling: Security issues for ad-hoc wireless networks. *Security Protocols. 7th International Workshop Proceedings, Lecture Notes in Computer Science*, pages 172–194, 1999.

[SBDB01] R. Sekar, M. Bendre, D. Dhurjati, and P. Bollineni. A fast automaton-based method for detecting anomalous program behaviors. In *Proceedings of the 2001 IEEE Symposium on Security and Privacy*, May 2001.

[SMGLA97] B. R. Smith, S. Murthy, and J.J. Garcia-Luna-Aceves. Securing distance-vector routing protocols. In *Proceedings of Internet Society Symposium on Network and Distributed System Security*, pages 85–92, San Diego, California, February 1997.

[WVW00] F. Wang, B. Vetter, and S. F. Wu. Secure routing protocols: Theory and practice. Technical report, North Carolina State University, 2000.

[ZH99] L. Zhou and Z. J. Haas. Securing ad hoc networks. *IEEE Network*, 13(6):24–30, Nov/Dec 1999.

[ZL00] Y. Zhang and W. Lee. Intrusion detection in wireless ad-hoc networks. In *Mobile Computing and Networking*, pages 275–283, 2000.