Probabilistic Techniques for Intrusion Detection Based on Computer Audit Data

Nong Ye, *Member, IEEE,* Xiangyang Li, Qiang Chen, Syed Masum Emran, and Mingming Xu

***Abstract*—This paper presents a series of studies on probabilistic properties of activity data in an information system for detecting intrusions into the information system. Various probabilistic techniques of intrusion detection, including decision tree, Hotelling's T^2 test, chi-square multivariate test, and Markov chain are applied to the same training set and the same testing set of computer audit data for investigating the frequency property and the ordering property of computer audit data. The results of these studies provide answers to several questions concerning which properties are critical to intrusion detection. First, our studies show that the frequency property of multiple audit event types in a sequence of events is necessary for intrusion detection. A single audit event at a given time is not sufficient for intrusion detection. Second, the ordering property of multiple audit events provides additional advantage to the frequency property for intrusion detection. However, unless the scalability problem of complex data models taking into account the ordering property of activity data is solved, intrusion detection techniques based on the frequency property provide a viable solution that produces good intrusion detection performance with low computational overhead.**

***Index Terms*—Anomaly detection, computer audit data, intrusion detection, pattern recognition.**

I. Introduction

VULNERABILITIES and bugs of information systems are often exploited by malicious users to intrude into information systems and compromise security (e.g., availability, integrity and confidentiality) of information systems [1]–[10]. As information systems become increasingly complex, vulnerabilities and bugs of information systems are inevitable for technical and economic reasons. Hence, the possibility of intrusions into information systems always exists. In order to protect information systems, it is highly desirable to detect intrusive activities while they are occurring in information systems.

An information system consists of host machines and communication links between host machines. Existing intrusion detection efforts [11]–[43] focus mainly on two sources of activity data in an information system: network traffic data and computer audit data. Network traffic data contain data packets traveling over communication links between host machines, and thus capture activities over communication networks. Audit trail data capture activities occurring on individual host machines. Activity data of an information system contain not only useful information to uncover intrusive activities but also much irrelevant information.

This paper presents a series of studies performed at the Information and Systems Assurance Laboratory of Arizona State University to reveal a few probabilistic properties of computer audit data that are important to intrusion detection. Intrusion detection techniques, including decision tree, Hotelling's T^2 test, chi-square multivariate test and Markov chain, are used in these studies. Section II review attributes of activity data used in existing work on intrusion detection. Section III generalizes probabilistic properties from attributes of activity data. Sections IV describes computer audit data used in our studies, and presents these studies and their results concerning probabilistic properties of activity data. Section V gives a conclusion.

II. Attributes of Activity Data in Existing Work

There are two general approaches to detecting intrusions [11]–[43]: anomaly detection (named behavior-based approach in some literature [11]) and pattern recognition (named knowledge-based approach [11] or misuse detection [29] in some literature). Pattern recognition techniques [11], [16]–[25] identify and store signature patterns of known intrusions, match activities in an information system with known patterns of intrusion signatures, and signal intrusions when there is a match. Pattern recognition techniques are efficient and accurate in detecting known intrusions, but cannot detect novel intrusions whose signature patterns are unknown.

Anomaly detection techniques establish a profile of a subject's normal activities (a norm profile), compare observed activities of the subject with its norm profile, and signal intrusions when the subject's observed activities differ largely from its norm profile [26]–[43]. The subject may be a user, file, privileged program, host machine, or network. Denning [29] provides a justification of the anomaly detection approach to intrusion detection. Anomaly detection techniques can detect both novel and known attacks if they demonstrate large differences from the norm profile. Since anomaly detection techniques signal all anomalies as intrusions, false alarms are expected when anomalies are caused by behavioral irregularity instead of intrusions. Hence, pattern recognition techniques and anomaly detection techniques are often used together to complement each other.

Manuscript received September 1, 2000; revised February 1, 2001. This work was supported in part by the Air Force Research Laboratory—Rome (AFRL-Rome) under Agreement F30602-98-2-0005, the Air Force Office of Scientific Research (AFOSR) under Grant F49620-98-1-0257, and the Defense Advanced Research Projects Agency (DARPA)/AFRL-Rome under Grant F30602-99-1-0506.

N. Ye, X. Li, Q. Chen, and M. Xu are with the Information and Systems Assurance Laboratory, Arizona State University, Tempe, AZ 85287 USA (e-mail: nongye@asu.edu).

S. M. Emran is with the iDEN BSC Development Group, Motorola, Schaumburg, IL 60196 USA.

Publisher Item Identifier S 1083-4427(01)05288-2.

Existing efforts on intrusion detection have considered mainly the following attributes of activities in information systems:

1) occurrence of individual events, e.g., audit events, system calls, commands, error messages, IP source address, and so on;
2) frequency of individual events, e.g., number of consecutive password failures;
3) duration of individual events, e.g., CPU time of a command, and duration of a connection;
4) occurrence of multiple events combined through logical operators such as AND, OR, and NOT;
5) frequency histogram (distribution) of multiple events, and sequence or transition of events;
6) sequence or transition of events.

Attributes 1, 2, 4, and 6 often appear in intrusion signatures that are represented in manually coded rules [16]–[18] or automatically learned rules [19]–[22] in some pattern recognition techniques. Attribute 6 appears in state transition diagrams [23], [24] and colored Petri nets [25] that are used in some pattern recognition techniques to represent intrusion signatures.

Several anomaly detection techniques exist and differ in the representation of a norm profile and the inference of a deviation from the norm profile. Specification-based anomaly detection techniques describe security policies and authorized activities of a well-defined subject (e.g., a privileged program or a network server) in terms of formal logic and activity graph [26]–[28]. Statistical-based anomaly detection techniques build a statistical profile (e.g., statistical distribution) of a subject's normal activities from historic data [29]–[33]. Anomaly detection techniques based on regression [34] or artificial neural networks [35], [36] learn from historic data to predict the next event from a series of the past events. Anomaly detection techniques based on immunology capture a large set of event sequences as the norm profile from historic data of a subject's normal activities, and use either negative selection or positive selection algorithms to detect the difference of incoming event sequences from event sequences in the norm profile [36]–[39]. There are also anomaly detection techniques that use a first-order or high-order Markov model of event transitions to represent a norm profile [39]–[43]. A first-order model of event transitions assumes that the next event depends on only the last event in the past. A higher order Markov model of event transitions assumes that the next event depends on multiple events in the past.

Specification-based anomaly detection techniques can readily incorporate attributes 1, 2, 4, and 6. Statistical-based anomaly detection techniques can build a statistical norm profile based on attributes 2, 3, and 5. Attribute 6 is incorporated in anomaly detection techniques based on regression, artificial neural networks, immunology, and Markov models.

Although attributes 1–6 appear in existing work on intrusion detection, it is not clear which properties are critical to performance of intrusion detection for several reasons. First, existing studies investigating different attributes often use different data sets, making the comparison of activity data attributes difficult. Second, existing comparative studies using the same data set focus mainly on differences among various intrusion detection algorithms dealing with the same attribute of activity data [39].

The following sections present generalized probabilistic properties of activity data and our comparative studies on these probabilistic properties concerning their importance to intrusion detection.

III. Probabilistic Properties of Activity Data

Attributes 1–6 can be categorized into three groups: attributes 1, 2, 4, and 5 concerning the frequency property of events, attribute 3 concerning the duration property of events, and attribute 6 concerning the ordering property of events. There may be other aspects of activity data that are not represented by these three properties. This paper focuses on only these three properties.

For the frequency property of events, we can use a set of random variables, $(\mathrm{X}_1, \ldots, \mathrm{X}_\mathrm{n})$, to represent the frequency of n different types of events (e.g., commands, system calls or audit events) for a given sequence of events. If we are interested in attributes 1 and 2—single or multiple occurrences of the ith event type from a pool of n possible event types for a given sequence of events, we examine the value of $\mathrm{X_i}$ from the vector of $(\mathrm{X}_1, \ldots, X_\mathrm{n})$. A denial-of-service attack may manifest through an unusually high frequency of a single event type. If we are interested in attributes 4 and 5—single or multiple occurrences of multiple events in combination, we examine the multivariate frequency distribution of $(\mathrm{X}_1, \ldots, \mathrm{X}_\mathrm{n})$.

For the duration property of events, we can use n sets $\{X_1\}, \ldots, \{X_\mathrm{n}\}$ to represent the duration values of n different event types for a given sequence of events, where each set contains duration values of events of a certain type. Considering the execution of a program as an event, the duration of this event is the program execution time. A trojan horse program may manifest through a change in the program execution time.

For the ordering property of events, $(\mathrm{X}_1, \ldots, \mathrm{X}_\mathrm{t})$ gives a time-series representation of a given event sequence, where X_t denotes an event occurring at time t. To take into account the ordering of events, complex data models are usually required, as demonstrated by a number of studies [34]–[43]. These complex data models demand for computationally intensive learning and/or inference procedures that are not scalable to mountains of activity data from an information system. An information system, even as small as a host machine, usually produces large amounts of activity data at a high frequency regardless of whether there are users' application processes in the information system. The large computational overhead associated with the ordering property raises a question. Is the ordering property of events necessary for intrusion detection?

Note that the vector representation of the frequency property can be derived from the set representation of the duration property and from the times-series representation of the ordering property. An intrusion typically consists of a series of events in an information system. The vector representation of the frequency property contains the least amount of information about the given series of events among the representations of the three properties. However, the frequency distribution of multiple event types still provides information about the collective activity level of these event types. To answer the question on the order property, we can answer another question. Is the

frequency property sufficient for intrusion detection? If the frequency property of multiple event types for a given sequence of events is sufficient to produce good intrusion detection performance, another question follows. Can intrusion detection be stateless, in other words, is a single event at a given time sufficient to detect intrusions? Section IV presents a series of studies to answer these questions.

IV. Comparative Studies

Our studies use various probabilistic techniques to detect intrusions, including decision tree, Hotelling's T^2 test, chi-square multivariate test and Markov chain, because different properties of activity data require different data models. Decision tree is a data mining technique that we use here as a pattern recognition technique. The other techniques are anomaly detection techniques. Before we present these studies, we first describe computer audit data used in these studies.

A. Training and Testing Data

An anomaly detection technique requires data of only normal activities in an information system to build a norm profile. Data of both normal activities and intrusive activities in an information system are required to learn intrusion signatures for a pattern recognition technique. The testing data should contain data of both normal activities and intrusive activities to test the performance of intrusion detection.

We use computer audit data from a Sun SPARC workstation with the Solaris operating system, and focus on intrusions into a host machine that leave trails in computer audit data. The Solaris operating system from Sun Microsystems Inc. has a security extension called the basic security module (BSM). BSM supports the monitoring of activities on a host machine by recording security-relevant events. BSM auditable events fall into two categories: kernel events and user-level events. Kernel events are generated by system calls to the kernel of the Solaris operation system. User-level events are generated by application software.

There are more than 250 different types of BSM auditable events, depending on the version of the Solaris operating system. Since there are about 284 different types of BSM audit events on our host machine, we consider 284 event types in our studies. An BSM audit record for each event contains a variety of information, including the event type, user ID, group ID, process ID, session ID, the system object accessed, etc. In our studies, we extract and use only the event type, because many existing studies [26]–[43] use only the type of events and produce good intrusion detection performance. Hence, activities on a host machine are captured through a stream of audit events, each of which is characterized by the event type.

A sample of computer audit data of normal activities is downloaded from the Massachusetts Institute of Technology (MIT) Lincoln Lab at http://ideval.ll.mit.edu/1998/1998_index.html. At this web site, four sets of data are provided:

1) sample data;
2) four hour subset of training data;
3) seven weeks of training data;
4) two weeks of testing data.

We download the sample data which contain audit data of both normal and intrusive activities. These audit data of normal activities are generated by the MIT Lincoln Lab through simulating activities in a real information system used by the U.S. Air Force. Intrusions are simulated on the background of normal activities. Because intrusive activities in this small sample data are very limited, we use audit data of normal activities only from this sample data.

According to the provided description of the starting and ending times of attack activities, we obtain a block of audit data for the period when no attack activities occur. This block of audit data containing a stream of 3019 audit events [12]. These audit data of normal activities is divided into two different parts: the first 1613 audit events and the remaining 1406 audit events. The first part, consisting of 1613 audit events, is included in the training data set as the computer audit data of normal activities. The second part, consisting of 1406 audit events, is included in the testing data set as the computer audit data for normal activities.

Computer audit data of intrusive activities are generated in our laboratory by simulating 15 intrusion scenarios that we have collected over years from various information sources. Table I gives the description of these intrusion scenarios. These intrusion scenarios are simulated in a random order. A student manually runs these intrusion scenarios on a Sun SPARC workstation with the same version of the Solaris operating system as the Solaris operating system that is used to generate audit data at the MIT Lincoln Laboratory, while the auditing facility is turned on. These intrusion scenarios generate a stream of 1751 audit events. The first 526 audit events produced from the first eight intrusion scenarios are included in the training data set as the computer audit data of intrusive activities. The remaining 1225 audit events from the remaining seven intrusion scenarios are included in the testing data set as the computer audit data of intrusive activities.

Hence, the training data set contains 1613 audit events of normal activities and 526 audit events of intrusive activities. The testing data contains 1406 audit events of normal activities and 1225 audit events of intrusive activities. Although audit data of normal activities and audit data of intrusive activities come from different host machines, the same Solaris operating system on these host machines produces the consistent information about event types. Since only the event type is extracted from each audit record, putting together audit data from different host machines in the training data set and the testing set does not become a concern in this study.

The same set of training data and the same set of testing data are used by each intrusion detection technique in our studies. Although the training data set contains computer audit data of both normal activities and intrusive activities, an anomaly detection technique uses only computer audit data of normal activities to build a norm profile. A pattern recognition technique uses computer audit data of both normal activities and intrusive activities in the training data set to learn intrusion signatures. Each intrusion detection technique is tested using the entire set of the testing data, containing computer audit data of both normal activities and intrusive activities. The training and the testing of each intrusion detection technique are performed off-line using

TABLE I
DESCRIPTION OF INTRUSION SCENARIOS USED IN THE STUDY

Scenario Number	Description of intrusive activities
1	Link the printer driver to a user program, and then execute the printing program. Before the printing program completes the job, replace the link with reference to the user program, in order to bypass the security check and let the user program obtain the same priority of the printing program.
2	Use the command of *rcp* to copy the password file from a remote host
3	Link the password file with the dead.letter. The dead.letter is created by the system when there are dead (return) mails. The system has the "write" right on the password file, which makes it possible for the user to change the password file through the dead mails.
4	Edit the *.rhost* file in order to remotely access the host later.
5	Attempt to edit and view the password file.
6	Execute a binary executable code that needs /dev/ttyb as an argument.
7	Attempt to login, but fail three times
8	Link the password file with a temporal log file that system could generate, in order to overwrite the password file later.
9	Use the command of *lp* to print a linked long file. Before the printing job is done, replace the link with reference to the password file, in order to print the password file.
10	Use the command of *rlogin* to login on a remote host without a password, provided that the account is in the *.login* file and in the .rhosts file on the host.
11	Use the command of *finger* to get sensitive information from a remote host.
12	Link a user file to a system-generated file.
13	Move system files for creating an executable file under the system directory to make a regular user become the root user.
14	Use the command of *rsh* to view and edit the password file in a remote server.
15	Link the shadow password file to a temporal log file that system could generate, in order to overwrite the password file later.

one file of the training data set and another file of the testing data set respectively.

B. Decision Tree and Results

There are two kinds of variables in a decision tree. One is target variable or class, and the other is predictor variable. A data point in the training data set contains the values of both predictor variables and a target variable. That is, the training data are labeled by the target values. A data point in the testing data set contains the values of only predictor variables. After testing, each data point in the testing data is assigned a target value and classified by the target value.

During training, a decision tree is constructed by recursively partitioning data points in the training data set into branches according to values of predictor variables until a stopping criterion is met [44]–[47]. Each branch contains a subset of data points with less inconsistency with respect to their target values. A common stopping criterion for a branch is that all data points in the branch have the same target value, which then produces a leaf in the decision tree. During testing, a data point is passed through the decision tree to reach a leaf according to its values of predictor variables, and is assigned the target value of this leaf.

Decision tree is used as a pattern recognition technique in our studies to construct a decision tree from the training data and to classify the testing data. Each data point in the training data set is labeled by the value of the target variable to indicate whether it is normal or intrusive. A normal data point has 0 as the target value. An intrusive data point has 1 as the target value. After training, each path from the root to a leaf of the decision tree represents a pattern of activities. We assign an indications and warning (IW) value to each leaf in the decision tree to indicate the likelihood of intrusion. The IW value for a leaf is computed by averaging the target values of the training examples in that leaf. The higher the IW value, the more likely the leaf represents intrusive activities. During testing, a data point is classified into a leaf according to the values of its predictor variables, and takes the IW value of the leaf.

We use the chi-squared automatic interaction detector (CHAID) decision tree algorithm in the AnswerTree 2.0 software from SPSS [48] for learning a decision tree from our training data. The CHAID algorithm uses chi-square statistics to identify optimal partitions. Details of our implementation can be found in [49].

To answer the question on whether a single event is sufficient to detect intrusions, we develop two different representations of predictor variables: a single-event representation and a frequency distribution representation. The single-event representation considers only a single event at a given time. The frequency distribution representation considers the frequencies of multiple event types within a given sequence of events.

In the single-event representation, there is only one predictor variable, X, with a value corresponding to the event type at a given time. Since there are 284 possible event types, the predictor variable can take one of 284 possible values.

In the frequency distribution representation, there are 284 predictor variables, $(X_1, X_2, \ldots, X_{284})$, for 284 event types respectively. The value for each of 284 predictor variables represents the frequency of an event type within a given sequence of audit events. We use the exponentially weighted moving average method [50] to compute the value of X_i, that is, the frequency of the ith event type in a sequence of audit events in the recent past.

$$X_i(t) = \lambda * 1 + (1 - \lambda) * X_i(t - 1)$$
if the current event—event t—belongs to the ith event type

$$X_i(t) = \lambda * 0 + (1 - \lambda) * X_i(t - 1)$$
if the current event—event t—is different from the ith event type

where

- $X_i(t)$ observed value of the ith variable in the vector of an observation $(X_1, X_2, \ldots, X_{284})$ for the current event—event t;
- λ smoothing constant that determines the decay rate;
- i $= 1, \ldots, 284$.

By using the exponentially weighted moving average method, more recent observations receive larger weights in the frequency computation. For example, the observation at the current event—event t—receives a weight of λ, the $(t - 1)$th observation receives a weight *of* $\lambda(1 - \lambda)$, and the $(t - k)$th observation receives a weight of $\lambda(1 - \lambda)^k$. An observation is made at each event.

In our studies, we let λ be 0.3—a commonly used value for the smoothing constant [48]. Fig. 1 shows the decay effect of the smoothing constant 0.3. We can see from Fig. 1 that after the $(t - 14)$th observation $(k = 14)$ the weight drops close to zero. That is, the frequency value of $X_i(t)$ at the current event—event t—takes into account about the past 15

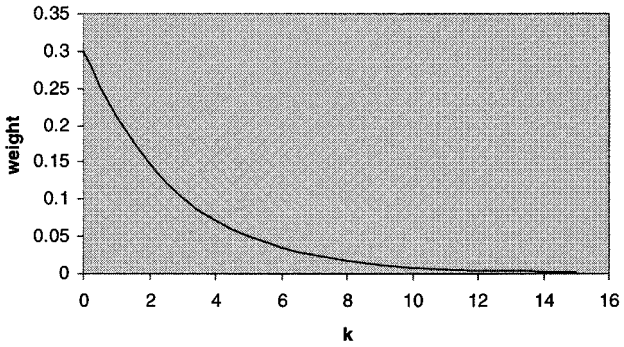

Fig. 1. Decay effect of the smoothing constant 0.3.

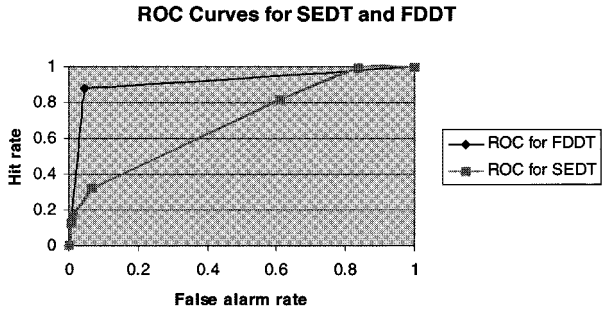

Fig. 2. ROC curves of decision trees.

audit events ($k = 0, \ldots, 14$). We initialize $X_i(0)$ to 0 for $i = 1, \ldots, 284$.

If we take a real-time unit (e.g., second) for t in $X_i(t)$, the frequency distribution representation can convey not only the relative frequency distribution of 284 audit events in a stream of audit events for a given timeframe, but also the intensity of individual events for that timeframe. However, the intensity of activities in an information system has large variations over time, e.g., from day to night. At night, there may be little activities in the information system. Inactive periods do not give us accurate estimates of the relative frequency distribution of audit events. Hence, we have separate studies on the intensity of individual events and the relative frequency distribution of multiple events. Another paper reports our intrusion detection work that examines the intensity of individual events for a given timeframe [51]. This paper focuses on the relative frequency distribution of 284 audit events within a given sequence of events by making an observation at every event rather than every time unit.

To accurately capture the relative frequency distribution of 284 audit events, we number only time points when audit events are observed in the frequency distribution representation. For example, given the following stream of audit events, we number them 1, 2, 3, ... for time t:

$$t = 0, \quad 1, \quad 2, \quad 3, \quad \ldots\ldots$$
$$EventType3,\ EventType8,\ EventType1,\ \ldots\ldots$$

For each audit event in the training data and the testing data, we obtain an observation vector of $(X_1, \ldots, X_{284})$. For the above example, at $t = 0$, all variables in the vector of $(X_1, \ldots, X_{284})$ have a value of 0. At time $t = 1$, X_3 has a value of 0.3 $(= 0.3 * 1 + 0.7 * 0)$, and all other variables have a value of 0. At time $t = 2$, X_3 has a value of 0.21 $(= 0.3 * 0 + 0.7 * 0.3)$, X_8 has a value of 0.3 $(= 0.3 * 1 + 0.7 * 0)$, and all other variables have a value of 0. At $t = 3$, X_3 has a value of 0.147 $(= 0.3 * 0 + 0.7 * 0.21)$, X_8 has a value of 0.21 $(= 0.3 * 0 + 0.7 * 0.3)$, X_1 has a value of 0.3 $(= 0.3 * 1 + 0.7 * 0)$, and all other variables have a value of 0.

Using the single-event representation, we obtain 1613 data points for normal audit events and 526 data points for intrusive audit events in the training data set. Each data point contains the value of a predictor variable X and a target value. Using such training data, the CHAID algorithm produces a decision tree, called the single-event decision tree (SEDT). SEDT is then used to obtain an IW value for each of 1406 data points for normal audit events and 1225 data points for intrusive data events in the testing data set. Fig. 2 shows the receiver operator characteristic (ROC) curve of the SEDT testing results.

Each point in an ROC curve indicates a pair of the hit rate and the false alarm rate for a signal threshold. For example, if the signal threshold for the IW values of the testing data is set to 0.5, we signal a testing data point whose IW value is greater than or equal to 0.5 as intrusive. There are no signals on testing data points whose IW values are less than 0.5. If there is a signal on a data point for an intrusive event in the testing data, this is a hit. If there is a signal on a data point for a normal event in the testing data, this is a false alarm. The hit rate is computed from dividing the total number of hits by the total number of intrusive events in the testing data. The false alarm rate is computed from dividing the total number of false alarms by the total number of normal events in the testing data. By varying the value of the signal threshold, we obtain an ROC curve. The closer the ROC is to the top-left corner (representing 100% hit rate and 0% false alarm rate) of the chart, the better detection performance the intrusion detection technique yields.

Using the frequency distribution representation, we also obtain 1613 data points for normal audit events and 526 data points for intrusive audit events in the training data set. Each data point contains an observation vector $(X_1, X_2, \ldots, X_{284})$ and a target value. Using such training data, the CHAID algorithm produces a decision tree, called the frequency-distribution decision tree (FDDT). FDDT is then used to obtain an IW value for each of 1406 data points for normal audit events and 1225 data points for intrusive data events in the testing data set. The ROC curve of the FDDT testing results is shown in Fig. 2.

The ROC curves from the SEDT testing results and the FDDT testing results reveal much better intrusion detection performance of FDDT than that of SEDT. In fact, the intrusion detection performance of SEDT is poor. For SEDT a hit rate of 81.5% brings up the false alarm rate to 60.9%, whereas for FDDT a hit rate of 88.1% brings up the false alarm rate to only 4.6%. Hence, the relative frequency of multiple event types within a given sequence of events gives a great advantage to intrusion detection. In other words, the frequency property of activity data is necessary for intrusion detection. Since a single event is not sufficient to produce good intrusion detection performance, stateless intrusion detection is not recommended.

Decision tree is a pattern recognition technique in which the learning of intrusion signatures requires both normal audit data

and intrusive audit data. The importance of the frequency property of activity data to intrusion detection is further verified below through two anomaly detection techniques (Hotelling's T^2 test and chi-square multivariate test) that use only normal audit data for training.

C. Hotelling's T^2 Test, Chi-Square Multivariate Test and Results

Hotelling's T^2 test is a multivariate statistical process control technique that detects anomalies in a process of a system. Let $\mathbf{X} = (X_1, X_2, \ldots, X_p)$ denote an observation of p variables from a process at time t. Using a data sample of size n, the sample mean vector $\overline{\mathbf{X}}$ and the sample variance–covariance matrix $\mathbf{S}$ of p variables are determined as follows [52]:

$$\overline{\mathbf{X}} = (\overline{X_1}, \overline{X_2}, \ldots, \overline{X_p}) \tag{1}$$

$$\mathbf{S} = \frac{1}{n-1}\sum_{i=1}^{n}(\mathbf{Xi} - \overline{X})(\mathbf{Xi} - \overline{\mathbf{X}})'. \tag{2}$$

Hotelling's T^2 statistic for an observation, $\mathbf{X}$, is determined as follows [52]:

$$T^2 = (\mathbf{X} - \overline{\mathbf{X}})' S^{-1} (\mathbf{X} - \overline{\mathbf{X}}). \tag{3}$$

A large computed value of T^2 indicates a large deviation of the observation $\mathbf{X}$ from the in-control population. Details of Hotelling's T^2 test and its application to intrusion detection can be found in [53].

When we apply Hotelling's T^2 test to intrusion detection, we use the same training data and the same testing data as those in decision tree studies, except that only audit events of normal activities are used for training a norm profile for Hotelling's T^2 test. Since only 11 event types actually appear in the training data set of 1613 audit events, the vector $\mathbf{X}$ of $(X_1, X_2, \ldots, X_{284})$ is reduced into a vector $\mathbf{X}$ with only 11 variables for the eleven event types, respectively. We perform the training and the testing for Hotelling's T^2 test using the vector $\mathbf{X}$ with only 11 variables. That is, using 1613 data points of $\mathbf{X}$ with 11 variables for normal audit events in the training date set, we compute $\overline{\mathbf{X}}$ and $\boldsymbol{S}$ in formulas (1) and (2) which fully describe the norm profile.

Using $\overline{\mathbf{X}}$ and $\boldsymbol{S}$, the T^2 value in (3) is then computed for each data point in the testing data set. The computed T^2 value is small if the data point conforms to the norm profile. The ROC curve for the testing results of Hotelling's T^2 test is plotted using various signal thresholds on the values of the computed T^2 value for the testing data points, as shown in Fig. 3.

With both the mean vector $\overline{\mathbf{X}}$ and the variance–covariance matrix $\boldsymbol{S}$, Hotelling's T^2 test provides a complete data model of multivariate data $\mathbf{X}$ in the frequency-distribution representation of the frequency property. Hotelling's T^2 test detects both mean shifts and counter-relationships in a multivariate manner. However, Hotelling's T^2 test is computationally intensive, requiring large memory to store the variance–covariance matrix and much computation time to compute the matrix and its inverse. It is not scalable to large amounts of computer audit data produced by an information system in real time.

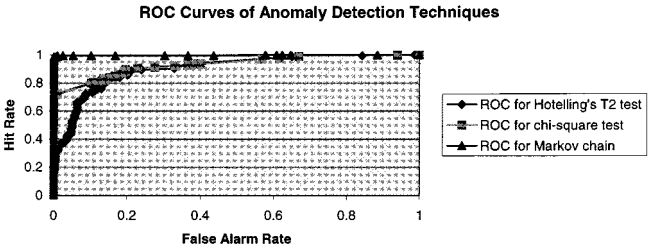

Fig. 3. ROC curves of anomaly detection techniques.

Hence, we develop the chi-square multivariate test with less computational overhead. The test statistic for the chi-square multivariate test is

$$X^2 = \sum_{i=1}^{p} \frac{(X_i - \overline{X_i})^2}{\overline{X_i}}. \tag{4}$$

In contrast to the T^2 test statistic, the X^2 test statistic does not account for the correlated structure of the p variables. With only the mean vector $\overline{\mathbf{X}}$ in (4), the chi-square multivariate test detects only the mean shift on one or more of the p variables. Details of the chi-square multivariate test and its application to intrusion detection can be found in [54].

When we apply the chi-square multivariate test to intrusion detection, we use the same training data and the same testing data as those for Hotelling's T^2 test. Using 1613 data points of $(X_1, X_2, \ldots, X_{284})$ for normal audit events in the training date set, we compute $\overline{X}$ which characterizes the norm profile. For each of those event types that do not appear in the training data set, we let the average of the variable for that event type take a very small value, 10^{-5} in this study, at the end of training such that the denominators in (4) are not zero. Using $\overline{X}$, the X^2 value in (4) is then computed for each data point of $(X_1, X_2, \ldots, X_{284})$ in the testing data set. The computed X^2 value is small if the data point conforms to the norm profile. The ROC curve for the testing results of the chi-square multivariate test is plotted using various signal thresholds on the computed X^2 values for the testing data points, as shown in Fig. 3.

The comparison of the ROC curves for Hotelling's T^2 test and the chi-square multivariate test reveals better intrusion detection performance of the chi-squared multivariate test than that of Hotelling's T^2 test. While the chi-square multivariate test detects mainly mean shifts, Hotelling's T^2 test detects both mean shifts and counter-relationships. In fact, Hotelling's T^2 test is more sensitive to counter-relationships than mean shifts because the T^2 test statistic is determined largely by the correlated structure of variables (variance–covariance matrix) [53]. Hence, the better intrusion detection performance of the chi-square multivariate test than Hotelling's T^2 test indicates that mean shifts may be more important to intrusion detection than counter-relationships.

The ROC curves for Hotelling's T^2 test and the chi-square multivariate test show better intrusion detection performance of these two anomaly detection techniques than performance of the decision tree based on the single-event representation, even though these two anomaly detection techniques use less data (only normal audit events) during training. This confirms the importance of the frequency property of activity data.

D. Markov Chain and Results

We apply a Markov model that takes into account the ordering property of multiple events for intrusion detection. The application of a Markov model helps answer the question about whether the ordering property of activity data provides additional advantage to intrusion detection, or whether we can detect intrusions from only the frequency property of activity data without the ordering property. Since first-order and high-order Markov models produce comparable intrusion detection performance [40]–[43], we apply Markov chain—a first-order Markov model that considers only one-step event transitions.

Let $\mathrm{X_t}$ be the value of a random variable or the state of a system at time t. A Markov chain is a stochastic process with the following assumptions [55], [56]:

$$P(X_{t+1} = i_{t+1} | X_t = i_t, X_{t-1} = i_{t-1}, \ldots, X_0 = i_0) = P(X_{t+1} = i_{t+1} | X_t = i_t), \text{ and} \quad (5)$$

$$P(X_{t+1} = i_{t+1} | X_t = i_t) = P(X_{t+1} = j | X_t = i) = p_{ij} \quad (6)$$

for all t and all states, where $\mathrm{p_{ij}}$ is the probability that the system is in a state j at time $\mathrm{t}+1$ given the system is in state t at time i. Equation (5) states that the probability distribution of the state at time $t+1$ depends on the state at time t, and does not depend on the previous states leading to the state at time t. Equation (6) specifies that a state transition from time t to time $\mathrm{t}+1$ is independent of time.

If the system has a finite number of states, 1, 2, …, s, the Markov chain model can be defined by a transition probability matrix [55], [56]

$$P = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1s} \\ p_{21} & p_{22} & \cdots & p_{2s} \\ \vdots & \vdots & \vdots & \vdots \\ p_{s1} & p_{s2} & \cdots & p_{ss} \end{bmatrix} \quad (7)$$

and an initial probability distribution

$$Q = [\, q_1 \quad q_2 \quad \cdots \quad q_s \,] \quad (8)$$

where q_{i} is the probability that the system is in state i at time 0, and

$$\sum_{j=1}^{j=s} p_{ij} = 1. \quad (9)$$

The probability that a sequence of states $\mathrm{X_{t-k}}, \ldots, \mathrm{X_t}$ at time $\mathrm{t-k}, \ldots, \mathrm{t}$ occurs in the context of the Markov chain model is computed as follows [55], [56]:

$$P(X_{t-k}, \ldots, X_t) = q_{xt-k} \prod_{i=k}^{1} P_{X_{t-i} X_{t-i+1}}. \quad (10)$$

In this study, the transition probability matrix and the initial probability distribution of a Markov chain model are learned from the training data that provide observations of the system state $\mathrm{X_0}, \mathrm{X_1}, \mathrm{X_2}, \ldots, \mathrm{X_{N-1}}$ at time $\mathrm{t} = 0, \ldots, \mathrm{N}-1$. The transition probability matrix and the initial probability distribution are computed from the training data as follows [47]:

$$p_{ij} = \frac{N_{ij}}{N_{i.}} \quad (11)$$

$$q_i = \frac{N_i}{N} \quad (12)$$

where

- N_{ij} number of observation pairs $\mathrm{X_t}$ and $\mathrm{X_{t+1}}$ with $\mathrm{X_t}$ in state i and $\mathrm{X_{t+1}}$ in state j;
- $N_{i.}$ number of observation pairs $\mathrm{X_t}$ and $\mathrm{X_{t+1}}$ with $\mathrm{X_t}$ in state i and $\mathrm{X_{t+1}}$ in any one of the states $1, \ldots, s$;
- Ni number of X_t's in state i;
- N total number of observations.

When we apply the Markov chain to intrusion detection, we use the same training data and the same testing data as those for Hotelling's $\mathrm{T^2}$ test and the chi-square multivariate test. We numbers only time points when audit events occur for time t. $\mathrm{X_t}$ has 284 possible states representing 284 possible events at time t.

Using the stream of 1613 audit events for normal audit events in the training date set, we compute the transition probability matrix P and the initial probability distribution Q according to (7) and (8) which characterize the norm profile. Using P and Q, we compute the probability that a sequence of the past 15 audit events at time t in the testing data, $\mathrm{X_{t-14}}, \ldots, \mathrm{X_t}$, occurs in the context of the Markov chain model as follows:

$$P(X_{t-14}, \ldots, X_t) = q_{xt-14} \prod_{i=14}^{1} P_{X_{t-i} X_{t-i+1}}. \quad (13)$$

Recall that Hotelling's $\mathrm{T^2}$ test and the chi-square multivariate test compute the test statistic based on the past 15 audit events when the smoothing factor is set to 0.3.

The higher probability we obtain from (13) for an event sequence, the more likely the event sequence is normal. An intrusive event sequence is expected to receive a low probability of support from the Markov chain model of the norm profile.

We assign a small probability of 10^{-5} to initial states and state transitions in the testing data if they have a zero probability value in the transition probability matrix P and the initial probability distribution Q, so that the final result from (13) is not zero. Details of the implementation can be found in [55].

The ROC curve for the testing results of the Markov chain is plotted using various signal thresholds on the computed probability values for event sequences in the testing data, as shown in Fig. 3.

We compare the ROC curve for the Markov chain based on the ordering property of activity data with the ROC curves for Hotelling's $\mathrm{T^2}$ test and the chi-square multivariate test based on the frequency property of activity data. The comparison reveals slightly better performance of the Markov chain. This indicates that the ordering property of activity data provides some additional advantage than the frequency property to intrusion detection, even using a simple first-order Markov model that considers only one-step event transitions.

V. Conclusion

From existing work on intrusion detection, we generalize three properties of activity data in an information system: the frequency property, the duration property, and the ordering property. Through a series of studies using the same training data and the same testing data, we provide answers to several questions concerning which properties are necessary to intrusion detection. Our studies show that the frequency property of multiple event types for a given sequence of events is necessary for intrusion detection. A single event at a given time is not sufficient for intrusion detection. Second, the ordering property provides additional advantage than the frequency property to intrusion detection.

Note that intrusive audit data in our studies are "pure" data without white noises from normal activities. Intrusions usually occur in an information system while normal activities are also occurring in the information system. Hence, in real time intrusive audit data are mixed with white noises of normal audit data. For such noisy data, the first-order Markov model of one-step event transitions may not produce good intrusion detection performance. For noisy data, high-order Markov model or event more complex data models may be warranted, which challenges us with the scalability problem of these complex data models. Since the two anomaly detection techniques (Hotelling's T^2 test and the chi-square multivariate test) based on the frequency property provide rather good intrusion detection performance, the frequency property provides a viable tradeoff between computational complexity and intrusion detection performance. When using the frequency property for intrusion detection, a complete data model as in Hotelling's T^2 test detecting both mean shifts and counter-relationships may not be necessary. A simplified data model as in the chi-square multivariate test detecting only multivariate mean shifts may be sufficient. Further studies of these properties of activity data using large amounts of noisy computer audit data are currently ongoing in our laboratory, and will be presented in future reports.

References

[1] W. Stallings, *Network and Inter-Network Security Principles and Practice*. Englewood Cliffs, NJ: Prentice-Hall, 1995.

[2] C. Kaufman, R. Perlman, and M. Speciner, *Network Security: Private Communication in a Public World*. Englewood Cliffs, NJ: Prentice-Hall, 1995.

[3] T. Escamilla, *Intrusion Detection: Network Security Beyond the Firewall*. New York: Wiley, 1998.

[4] B. Simons, "Building big brother," *Commun. ACM*, vol. 43, no. 1, pp. 31–32, Jan. 2000.

[5] P. G. Neumann, "Risks of insiders," *Commun. ACM*, vol. 42, no. 12, p. 160, Dec. 1999.

[6] M. Godwin, "Net to worry," *Commun. ACM*, vol. 42, no. 12, pp. 15–17, Dec. 1999.

[7] S. Jajodia, P. Ammann, and C. D. McCollum, "Surviving information warfare attacks," *Comput.*, vol. 32, no. 4, pp. 57–63, Apr. 1999.

[8] P. Mann, "Pentagon confronts mounting cyber risks," *Aviation Week Space Technol.*, vol. 150, no. 12, pp. 82–83, Mar. 22, 1999.

[9] B. H. Barnes, "Computer security research: A British perspective," *IEEE Softw.*, vol. 15, pp. 30–33, Sept./Oct. 1998.

[10] A. Boulanger, "Catapults and grappling hooks: The tools and techniques of information warfare," *IBM Syst. J.*, vol. 37, no. 1, pp. 106–114, 1998.

[11] H. Debar, M. Dacier, and A. Wespi, "Toward a taxonomy of intrusion-detection systems," *Comput. Networks*, vol. 31, pp. 805–822, 1999.

[12] R. Lippmann *et al.*, "Evaluating intrusion detection systems: The 1998 DARPA off-line intrusion detection evaluation," in *Proc. DARPA Inform. Survivability Conf. Expo.*, Los Alamitos, CA, Jan. 2000, pp. 12–26.

[13] D. Schnackenberg and K. Djahandari, "Infrastructure for intrusion detection and response," in *Proc. DARPA Inform. Survivability Conf. Expo.* Los Alamitos, CA, Jan. 2000, pp. 3–11, .

[14] T. Bass, "Intrusion detection systems and multi-sensor data fusion," *Commun. ACM*, vol. 43, no. 4, pp. 99–105, Apr. 2000.

[15] M. Stillerman, C. Marceau, and M. Stillman, "Intrusion detection for distributed applications," *Commun. ACM*, vol. 42, no. 7, pp. 62–69, July 1999.

[16] U. Lindqvist and P. A. Porras, "Detecting computer and network misuse through the production-based expert system toolset (P-BEST)," in *Proc. IEEE Symp. Security Privacy*, Oakland, CA, May 1999.

[17] P. A. Porras and P. G. Neumann, "EMERALD: Event monitoring enabling responses to anomalous live disturbances," in *Proc. NISSC*, Oct. 1997.

[18] P. G. Neumann and P. A. Porras, "Experience with EMERALD to date," in *Proc. Workshop Intrusion Detection Network Monitoring*, Santa Clara, CA, Apr. 1999, pp. 73–80.

[19] W. Lee, and S. J. Stolfo, "Data mining approaches for intrusion detection," in *Proc. 7th USENIX Security Symp.*, San Antonio, TX, Jan. 1998.

[20] W. Lee, S. J. Stolfo, and K. W. Mok, "A data mining framework for building intrusion detection models," in *Proc. IEEE Symp. Security Privacy*, May 1999.

[21] ——, "Mining audit data to build intrusion detection models," in *Proc. Fourth Int. Conf. Knowledge Discovery Data Mining*, New York, NY, Aug. 1998.

[22] ——, "Mining in a data-flow environment: Experience in network intrusion detection," in *Proc. Fifth ACM SIGKDD Int. Conf. Knowledge Discovery Data Mining*, San Diego, CA, Aug. 1999.

[23] G. Vigna and R. Kemmerer. NetStat: A network-based intrusion detection approach. presented at Proc. 14th Annu. Comput. Security Appl. Conf. [Online]. Available: http://www.cs.ucsb.edu/~kemm/netstat.html.

[24] G. Vigna, S. T. Eckmann, and R. A. Kemmerer, "The STAT tool suit," in *Proc. DARPA Inform. Survivability Conf. Expo*, Los Alamitos, CA, Jan. 2000, pp. 46–55.

[25] S. Kumar, "Classification and detection of computer intrusions," Ph.D. dissertation, Dept. Comput. Sci., Purdue Univ., West Lafayette, IN, 1995.

[26] C. Ko, G. Fink, and K. Levitt, "Execution monitoring of security-critical programs in distributed systems: A specification-based approach," in *Proc. IEEE Symp. Security Privacy*, 1997, pp. 134–144.

[27] S. Staniford-Chen *et al.*, "GrIDS—A graph-based intrusion detection system for large networks," in *Proc. 19th Nat. Inform. Syst. Security Conf.*, Oct. 1996.

[28] T. Bowen *et al.*, "Building survivable systems: An integrated approach based on intrusion detection and damage containment," in *Proc. DARPA Inform. Survivability Conf. Expo., Vol. II*, Los Alamitos, CA, Jan. 2000, pp. 84–99.

[29] D. E. Denning, "An intrusion-detection model," *IEEE Trans. Softw. Eng.*, vol. SE-13, pp. 222–232, Feb. 1987.

[30] D. Anderson, T. Frivold, and A. Valdes, "Next-generation intrusion detection expert system (NIDES): A summary," SRI Int., Menlo Park, CA, Tech. Rep. SRI-CSL-97-07, May 1995.

[31] H. S. Javitz and A. Valdes, "The SRI statistical anomaly detector," in *Proc. IEEE Symp. Res. Security Privacy*, May 1991.

[32] ——, "The NIDES statistical component description of justification," SRI Int., Menlo Park, CA, Tech. Rep. A010, Mar. 1994.

[33] Y. Jou *et al.*, "Design and implementation of a scalable intrusion detection system for the protection of network infrastructure," in *Proc. DARPA Inform. Survivability Conf. Expo.*, Los Alamitos, CA, 2000, pp. 69–83.

[34] W. DuMouchel and M. Schonlau, "A comparison of test statistics for computer intrusion detection based on principal components regression of transition probabilities," in *Proc. 30th Symp. Interface: Comput. Sci. Stat*.

[35] H. Debar, M. Becker, and D. Siboni, "A neural network component for an intrusion detection system," in *Proc. IEEE Comput. Soc. Symp. Res. Security Privacy*, Oakland, CA, May 1992, pp. 240–250.

[36] A. K. Ghosh, A. Schwatzbard, and M. Shatz. Learning program behavior profiles for intrusion detection. presented at Proc. First USENIX Workshop Intrusion Detection Network Monitoring. [Online]. Available: http://www.rstcorp.com/~anup/.

[37] S. Forrest, S. A. Hofmeyr, and A. Somayaji, "Computer immunology," *Commun. ACM*, vol. 40, no. 10, pp. 88–96, Oct. 1997.

[38] H. Debar, M. Dacier, M. Nassehi, and A. Wespi, "Fixed vs. variable-length patterns for detecting suspicious process behavior," in *Proc. Fifth Euro. Symp. Res. Comput. Security*, Louvain-la-Neuve, Belgium, September 16–18, 1998, pp. 1–15.
[39] C. Warrender, S. Forrest, and B. Pearlmutter, "Detecting intrusions using system calls: Alternative data models," in *Proc. IEEE Symp. Security Privacy*, 1999, pp. 133–145.
[40] W. DuMouchel. Computer intrusion detection based on Bayes factors for comparing command transition probabilities. Nat. Inst. Statist. Sci., Tech. Rep. no. 91. [Online]. Available: http://www.niss.org/downloadabletechreports.html.
[41] W.-H. Ju and Y Vardi. A hybrid high-order Markov chain model for computer intrusion detection. Nat. Inst. Statist. Sci., Tech. Rep. 92. [Online]. Available: http://www.niss.org/downloadabletechreports.html.
[42] M. Schonlau *et al.*. Computer intrusion: Detecting masquerades. Nat. Inst. Statist. Sci., Tech. Rep. 95. [Online]. Available: http://www.niss.org/downloadabletechreports.html.
[43] S. L. Scott. Detecting network intrusion using a Markov modulated nonhomogeneous Poisson process. [Online]. Available: http://www-rcf.usc.edu/~sls/fraud.ps.
[44] P. E. Utgoff, N. C. Berkman, and J. A. Clouse, "Decision tree induction based on efficient tree restructuring," *Mach. Learn., 10*, pp. 5–44, 1997.
[45] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees*. London, U.K.: Wadsworth, 1984.
[46] S. R. Safavian and D. Landgrebe, "A survey of decision tree classifier methodology," *IEEE Trans. Syst., Man, Cybern.*, vol. 21, pp. 660–674, May/June 1991.
[47] T. M. Mitchell, *Machine Learning*. New York: McGraw-Hill, 1997.
[48] SPSS Inc., *AnswerTree 2.0: User's Guide*. Chicago, IL: SPSS, Inc.
[49] N. Ye, X. Li, and S. M. Emran, "Decision trees for signature recognition and state classification," in *Proc. IEEE SMC Inform. Assurance Security Workshop*, West Point, NY, June 2000, pp. 189–194.
[50] R. A. Johnson and D. W. Wichern, *Applied Multivariate Statistical Analysis*. Upper Saddle River, NJ: Prentice-Hall, 1998.
[51] N. Ye and S. Vilbert, "The EWMA-EWMV technique for detecting anomalies in information systems," *IEEE Trans. Reliability*, submitted for publication.
[52] T. P. Ryan, *Statistical Methods for Quality Improvement*. New York: Wiley, 1989.
[53] N. Ye, Q. Chen, and S. M. Emran, "Hotelling's T^2 multivariate profiling for anomaly detection," in *Proc. IEEE SMC Inform. Assurance Security Workshop*, West Point, NY, June 2000, pp. 175–181.
[54] ——, "Chi-squared statistical profiling for anomaly detection," in *Proc. IEEE SMC Inform. Assurance Security Workshop*, West Point, NY, June 2000, pp. 182–188.
[55] N. Ye, "A Markov chain model of temporal behavior for anomaly detection," in *Proc. IEEE SMC Inform. Assurance Security Workshop*, West Point, NY, June 2000, pp. 166–169.
[56] W. L. Winston, *Operations Research: Applications and Algorithms*. Belmont, CA: Duxbury, 1994.
[57] P. Buttorp, *Stochastic Modeling of Scientific Data*. London: Chapman & Hall, 1995.

Nong Ye (M'92) received the B.S. degree in computer science from Peking University, Beijing, China, the M.S. degree in computer science from the Chinese Academy of Sciences, Beijing, and the Ph.D. degree in industrial engineering from Purdue University, West Lafayette, IN.

Since 1998, she has been an Associate Professor with the Department of Industrial Engineering, Arizona State University, Tempe. From 1994 to 1998, she was an Assistant Professor at the University of Illinois, Chicago. From 1991 to 1994, she was an Assistant Professor at Wright State University, Dayton, OH. Her research interests include assuring process quality and preventing faults and errors in information systems, human–machine systems, and manufacturing systems.

Dr. Ye is a Senior Member of the Institute of Industrial Engineers.

Xiangyang Li received the M.S. degree from the Chinese Academy of Aerospace Administration in 1996. He is currently pursuing the Ph.D. degree at the Department of Industrial Engineering, Arizona State University, Tempe.

His research interests include information system security and assurance, data mining, and system modeling and simulation.

Qiang Chen received the B.S. and M.S. degrees in manufacturing engineering from Beijing University of Aeronautics and Astronautics (BUAA), Beijing, China, in 1993 and 1999, respectively. He is currently pursuing the Ph.D. degree at Arizona State University, Tempe.

From 1993 to 1996, he worked as an Information Management Engineer with the Beijing Aircraft Maintenance and Engineering Company. His research interests include intrusion detection, data noise cancellation, and knowledge discovery.

Syed Masum Emran received the B.Sc.Engg. from Bangladesh University of Engineering and Technology, in 1997, and the M.S. degree in computer science from Arizona State University, Tempe, in 2000.

He is currently working as a Software Engineer at the iDEN BSC Development Group, Motorola. His research interests include network security and computer intrusion detection.

Mingming Xu received the B.S. degree in automation control from Shanghai Jiaotong University, Shanghai, China, and the M.S. degree in computer science from Fudang University, Fudang, China. Since 1999, he has been pursuing the Ph.D. degree at Arizona State University, Tempe.