PAPER *Special Section on New Technologies and their Applications of the Internet III*

An Anomaly Intrusion Detection System Based on Vector Quantization

Jun ZHENG[†a)], ***Student Member and*** **Mingzeng HU**[†b)], ***Nonmember***

SUMMARY Machine learning and data mining algorithms are increasingly being used in the intrusion detection systems (IDS), but their performances are laggard to some extent especially applied in network based intrusion detection: the larger load of network traffic monitoring requires more efficient algorithm in practice. In this paper, we propose and design an anomaly intrusion detection (AID) system based on the vector quantization (VQ) which is widely used for data compression and high-dimension multimedia data index. The design procedure optimizes the performance of intrusion detection by jointly accounting for accurate usage profile modeling by the VQ codebook and fast similarity measures between feature vectors to reduce the computational cost. The former is just the key of getting high detection rate and the later is the footstone of guaranteeing efficiency and real-time style of intrusion detection. Experiment comparisons to other related researches show that the performance of intrusion detection is improved greatly.

key words: *anomaly intrusion detection, usage profile, vector quantization, codebook, quantization error*

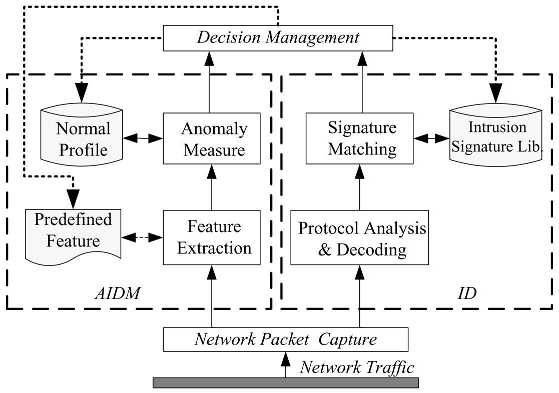

Fig. 1 Workflow of a general IDS with AID and MID.

1. Introduction

The network security becomes one of focus with the ever fast development of the Internet. In addition to network intrusion defensive techniques, such as firewall and encryption, Intrusion Detection System (IDS) is served as an important security barrier against computer intrusions. Generally, There are two general approaches in IDS: Misuse Intrusion Detection (MID) and Anomaly Intrusion Detection (AID). Integrated with MID and AID (Fig. 1), the IDS can get the holistic estimation of intrusion situations. Be similar to virus detection, MID is based on the signature matching to hunt for the signatures extracted from the known intrusions. However, AID mainly depends on creating the historical or long-term normal usage profile from which the anomaly analysis model looks for deviations of the short-term usage. Compared to a defined baseline of normal usage profile, the deviations can be treated as the suspicious anomalous events related to intrusions. So AID has the advantage that it can detect unknown intrusions.

In the machine learning and data mining community, many algorithms have been used in AID extensively [1], [2]. However, many papers pay more attention to maximize the detection rate while neglect the overall performance, especially the detection efficiency which is significant to the practical applications of algorithms. In order to address problems aforementioned, a paradigm for designing AID systems based on Vector Quantization (VQ) techniques [3] is proposed in this paper which concerns on enhancing the whole system performance. The design procedure optimizes the performance of AID by jointly accounting for accurate usage profile modeling by the larger VQ codebook and fast similarity measures between vectors for reducing the computational cost. The former is just the key of getting high detection rate and the later is the footstone of guaranteeing efficiency and real-time style of intrusion detection. Our final aim is to implement a high-performance AID system which can work cooperating with other MID systems (e.g. Snort [4]). The paper is organized as follows: Section 2 presents the related work of AID. Section 3 explains creating the normal usage profile and similarity measures used in the paper. Section 4 presents the new AID method based on VQ. Section 5 proposes a speedup strategy of similarity measure for enhancing the efficiency of AID. Section 6 describes the details of our experiments over DARPA data set. Finally, Sect. 7 gives conclusions and future work.

2. Related Work

The most classical AID model comes from Denning's the earlier landmark paper [5] which inspires the statistical methodology for AID [6], [7]. However, as computer intrusions become more complex, more and more machine learning and data mining algorithms are used for intrusion detection. Without maintaining the profile of normal behavior,

Manuscript received March 11, 2005.
Manuscript revised June 23, 2005.

[†]The authors are with the Department of Computer Science and Engineering, Harbin Institute of Technology (HIT), Harbin, 150001, China.
a) E-mail: zhengjun@hit.edu.cn
b) E-mail: mzhu@hit.edu.cn

DOI: 10.1093/ietisy/e89–d.1.201

the distance-based unsupervised clustering [8]–[11] and its related outlier detection algorithms [12] seem to be promising recently. Knorr and Ng [12] propose: "A point p in a data set is an outlier with respect to parameters k and d if no more than k points in the data set are at a distance of d or less from p." But these methods have some shortcomings when applied in AID. Just like [12], the outlier detection approaches are very sensitive to the choice of the parameters d, k, p which will be specified by the users, and in the same way, clustering methods equally do in choice of inter-cluster distance or extra-cluster distance. Strictly, both general clustering and outlier detection depend mightily on the precondition that the number of normal instances is significantly larger than the number of anomalies for the certain temporal space, and then the anomalies can be regarded as the outliers to the main body of data. These methods can not get higher detection rate in case of DoS/DDoS attacks or network probing attacks [8], [13] because of their huge mount of activities and the higher proportion attack traffic during the certain time window which is necessary and important for real-time style generally. So in practice, the setting of time interval granularity is a related problem.

On the other hand, AID can be considered as the binary classification of normal/anomalous in the context of machine learning. The classifier is trained on the dataset with the labeled data and then used to recognize the attack data. To improve the ability to distinguish between normal class and anomalous is the main objective that drives a classification algorithm. Many supervised classification algorithms (classifiers) have been applied to AID, for example, [14]–[22]. Support Vector Machine (SVM) and other kernel based classifiers can achieve the higher classification rate [14], [19]. But the problems of algorithm complexity and its kernel parameter selection embarrass the efficiency and usability in practice of intrusion detection. The same problem of efficiency arises in the applications of Hidden Markov Model (HMM) [16], [18], Neural Network [14], [17], K-Nearest-Neighbor (KNN) [21] and Learning Vector Quantization (LVQ) [22], especially in case of the huge amount of network traffic data in the network-based intrusion detection.

The most related researches to our approach are the papers [23], [24]. Based on the basic Self-Organizing Map (SOM) algorithm, the paper [23] aims at the network traffic anomaly detection regaring university campus. Because SOM is also used in our method to create the codebook of VQ, we will take some comparisons in the following sections. The paper [24] proposes to use new unsupervised learning rule called after the improved competing learning neural network (ICLN) and claims that it can be achieve faster training process than the basic SOM. ICLN is similar to the LVQ learning rule in some extent but mixed with the unsupervised approach. Different from the paper [23], ICLN is trained on the dataset including labeled data in an unsupervised way. After the training, the cluster class will be assigned in the manual way with setting the threshold in advance. For example, if more than 50% of the connections in a cluster were intrusions, the cluster and its centroid weight vector would be labeled as the intrusion,which seemed unwieldy a little in practice. Consider the large training dataset,the small numbers of nodes or clusters in the paper can not guarantee the higher detection accuracy which is based on the distance between the input vectors and its nearest-neighbor cluster centroid. The detailed analysis and comparisons mentioned above will be presented further in the following experiment sections. To the best of our knowledge, except the paper [24], there is little research that has been focused on advancing the AID efficiency which is fundamental to the system applicability.

3. Creating Normal Usage Profile

3.1 Normal Usage Profile

As mentioned in Sect. 1, it is of first importance in AID to create normal usage profile. The ***profile*** is the structure that characterizes the behavior of subjects with respect to objects in terms of statistical metrics and models of observed activity [5]. These structures can be treated as non-empty sets of usage patterns which also can be defined as sets of similar normal activities between subjects and objects. That is, the similar activities can be characterized and represented by the pattern, also named as the sub-profile. Therefore, the profile is the non-empty set of patterns (sub-profile): $profile = \bigcup pattern_i / \emptyset$.

Just as given in Table 1, the profile of usage behavior can be based on any concerned objects in addition to network traffic. However, due to dynamics and complexes of network traffic, it is more difficult to establish the veracious profiles of network traffic in the network based intrusion detection.

3.2 Similarity Measure

To create the normal usage profile of some spatial-temporal network traffic, the similarity measures of network traffic usage are necessary in this paper. According to the similarity measures of network traffic data, the data space of normal network traffic is partitioned into different subspaces and the similar traffic data will be grouped into the same subspace. Every subspace can be represented by the sub-profiles presenting the similar network traffic usage. In this paper, similarity is modeled using a distance function: the distance function along with a set of data space defines a metric space (Euclidian space). Further, we define the network traffic data, i.e. TCP flow, in form of feature vector:

Table 1 Profile classification.

Data Object	Profile
System Audit Log (e.g. BSM) [6]	Host Computer
System Call Sequence[18]	System Call
User Command Sequence[16]	User Command
Network Packet/Connection/Flow [20]	Network Traffic

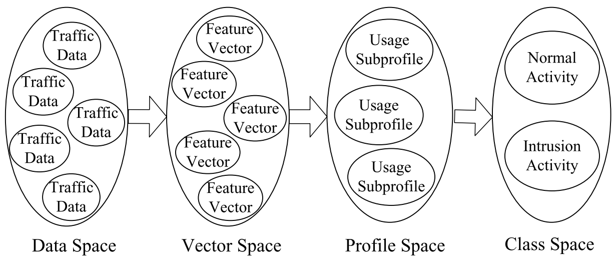

Fig. 2 Space translation in AID.

Definition 1: Every TCP flow is a data point in the n-dimension Euclidian space R^n and R^n also is called as vector space.

$$\text{TCPF low} = \{X | X \in R^n\} \tag{1}$$

Every TCP flow is expressed by the form of feature vector $X = (x_1, x_2, \cdots, x_n)$. And x_i is the feature attribute.

In the paper, just as Fig. 2, four spaces are involved in anomaly intrusion detection. Figure 2 can help the reader to understand the following of paper easier.

4. Vector Quantization for Anomaly Intrusion Detection

4.1 Profile Creating Based on Vector Quantization

Vector Quantization (VQ) [3] is an efficient compression technique especially used in data compressions of image and speech based on the ***similarity measures between feature vectors***. Through the compression function of VQ, it is easy to construct the index structure for the high-dimension multimedia database [3], [25]–[27]. In the following, we will give the detail description of VQ and application to AID.

Definition 2: Vector Quantization Q can be defined as a mapping function from Euclidean space R^n into a certain finite subset C:

$$Q : R^n \to C, C = \{Y_0, Y_1, \cdots, Y_{m-1} | Y_i \in R^n\} \tag{2}$$

The representative vector in C is called ***codeword***, C is called ***codebook*** and m is the codebook dimension.

Definition 3: Given the input vector $X = (x_1, x_2, \cdots, x_n)$ and the codeword $Y_i = (y_{i1}, y_{i2}, \ldots, y_{in})$ $(i = 1, 2, \ldots, m)$, every input vector will be assigned with a codeword in which the distortion (Euclidian distance) between this codeword and the input vector is smallest among all codewords. The distortion is defined as Quantization Error (QE):

$$QE = \|X - Y_i\| = \left[\sum_{j=1}^{k} (x_j - y_{ij})^2\right]^{1/2} \tag{3}$$

Definition 4: The usage sub-profile P of network traffic can be defined:

$$P_i = \{X \in R^n, Q(X) = Y_i\} \ (i = 1, 2, \ldots, m) \tag{4}$$

All X with the sub-profile P_i can be represented by codeword Y_i. And the whole profile is represented by the codebook C.

The potential of VQ application to the usage profile establishment of network traffic is that VQ can compartmentalize the traffic feature vector space to groups by comparing the similarities of feature vectors. The profile can be recapitulated and indexed by the codebook.

- Via the codebook structure of VQ, we can get the profile to character the network traffic usage behaviors. The profile of normal usage is represented by the codebook on which the VQ-index structure is based. The sub-profile is indexed by the same codeword in the VQ-index profile structure;
- Serving as the precondition of anomaly measures, the structural profile creating makes it possible in detection phase to measure anomaly deviations between the test vectors and the indexed sub-profile in a quantitative way.

The structurally constrained profile based on the codebook of VQ has two advantages:

- It needn't make assumption of data probability distributions or have the apriori information about normal usage behavior. Some AID methods need the assumption that data present some certain probability distribution, for example [28];
- It needn't create the complex describing rules which contain the information for characterizing the normal usage behavior and intrusion behavior [20].

4.2 Codebook Design

The most important step to implement the VQ framework is the codebook construction which is treated as the training process by the normal network traffic with a certain algorithm. The LBG clustering algorithm [25] and Competitive Learning [29] are the most widely used approaches to design codebook of VQ. In this paper, we use SOM (Self-Organizing Map) [30] to train the large structural codebook. After the training, the map with weight vectors in SOM can be served as the codebook of VQ. The following is the algorithm description of SOM.

The input vector: $X = (x_1, x_2, \cdots, x_n)$

The codeword: $W = (w_1, w_2, \cdots, w_n)$

Step 1. Initialize every codeword of SOM with random values: $W_j(0)$

Step 2. Compute the distance between the input vector X_i and the codeword $W_j(t)$, designate the winner node j^* with the smallest distance.

$$j^* = \arg \min_{1\le j\le m} \|X_i - W_j(t)\| \tag{5}$$

The Euclidean distance is chosen as Quantization Errors (QEs):

$$D = \left\|X_i - W_j(t)\right\| = \left[\sum_{k=1}^{n} (x_{ik} - w_{jk}(t))^2\right]^{1/2} \tag{6}$$

Step 3. Update the winner vectors of the winner node and its neighborhood:

$$w_{jk}(t+1) = w_{jk}(t) + \alpha(t)[x_{ik} - w_{jk}(t)], j \in N(t) \tag{7}$$

$N(t)$: non-increasing neighborhood function;
$\alpha(t)$: learning rate function, $0 < \alpha(t) < 1$.
Step 4. Repeat *Step 2* and *Step 3* until SOM learning stabilizes

4.3 Anomaly Intrusion Detection Based on Vector Quantization

In order to find TCP network intrusions as deviations from normal usage profile, we measure the similarity between the short-term usage behavior and long-term using the QEs which are the Euclidean distance between the input vector and the closest codeword. The detection process is also seemed as the nearest-neighbor (NN) search in the codebook based on the VQ-index structure.

Definition 5: Given a collection of n-dimensional points, VQ codebook C, an input point p, find a codeword q, the closest to p than any other codewords in C. That is $\{q\,|q \in C \cdot \text{and} \cdot \forall r \in C, \|r-p\| > \|q-p\|\}$ and $\|q-p\| = QE$. Given the distance deviation threshold ε, p is an anomaly intrusion point when $\|q-p\| \geq \varepsilon$.

5. Speedup Strategy of Similarity Measure

5.1 Computing Complexity Analysis

As aforementioned in Sect. 4, the computing complexity in our AID system is mainly concentrated on the similarity measures between the n-dimension feature vectors in both codebook training phase and detection phase, where measuring Euclidian distance function is the essential operation. The computational cost of measuring squared Euclidean distance is very high because a prohibitive number of mathematical operations are required especially when the input feature vector number and the dimension are large.

- *Profile Creating Phase* Look back to Sect. 4.2, SOM finds the winner codeword (weight vector) for each input vector by the full search to compute its Euclidean distance to all codewords. If the codebook size is m and the vector dimension is n, every time a vector is input, it needs to compute n times of power calculation, $(2n-1)$ times addition. So when every TCP flow feature vector is input to codebook, there are $m \cdot n$ times of power calculation, $m \cdot (2n-1)$ times of addition and $(m-1)$ times of comparison;
- *Detection Phase* The computational complexity of NN full-search is $O(mn)$ for every input TCP flow feature vector. The detection time is not satisfied to fulfill the real-time style of IDS because of large amount of TCP flow input vectors in the real high-speed networking environments.

5.2 Speedup Strategy of Similarity Measures

In our method, we implement the fast VQ with new fast NN search speedup strategy [31] to accelerate similarity measures in order to implement the high efficient AID system. It serves as the filter that eliminates the unnecessary similarity measures. To suppose the input feature vector is $X = (x_1, x_2, \cdots, x_n)$ and the codeword is $Y = (y_1, y_2, \ldots, y_n)$. For X and Y, compute:

$$S_x = \sum_{l=1}^{n} x_l, S_j = \sum_{l=1}^{n} y_{jl} \qquad (j = 1, 2, \ldots, m) \tag{8}$$

It is easy to prove that:

$$\begin{aligned} D(X, Y_j)^2 &= \sum_{l=1}^{n} (x_l - y_{jl})^2 \\ &\geq n \cdot \left[\sum_{l=1}^{n} \frac{(x_l - y_{jl})}{n}\right]^2 = \frac{(S_x - S_j)^2}{n} \end{aligned} \tag{9}$$

When computing QE, we first take the current minimum Euclidean distance D as $D_{\min}$ in Eq. (6):

$$(S_x - S_j)^2 \geq n \cdot D_{\min}^2 \tag{10}$$

If Y satisfies Eq. (10), then the Euclidean distance computing between codeword Y and X can be avoided so that computing cost can be reduced. To procure the faster algorithm, we will compute the S_j of the codeword Y_j and sort them in the way of ascending Y_j in terms of S_j value as shown in Fig. 3. In codebook creating and detection phase, the algorithm will search the codeword Y_j which is the nearest neighbor to the input TCP flow feature vector X by the dimidiate search algorithm according to the sort. Here, Eq. (10) is the computing filter rule served as reducing the compute cost of the Euclidean distance.

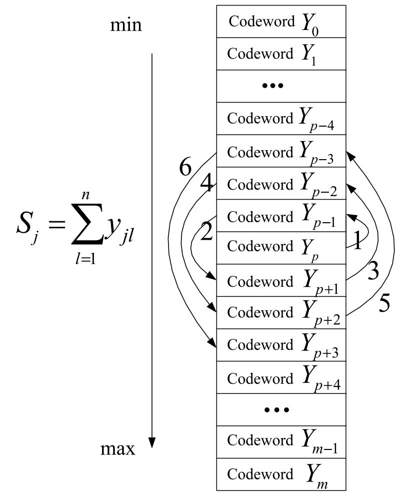

Fig. 3 Codeword sort.

6. Performance Evaluation and Results

6.1 Data Preprocessing

It is necessary to do data preprocessing to extract the feature attributes from TCP flows before the detection, and then, the date normalization will be processed to project whole feature attributes to a unit range. In this paper, the experiments are based on the TCP traffic. The program of data preprocess is based on our amended edition of *Libnids* [32].

6.1.1 TCP Flow Feature Attribute

The aim of feature extraction is to achieve the maximum difference degree between normal usage behaviors and intrusion behaviors. A feature vector of the TCP traffic flow is shown in Table 2. The *FlowState* attribute is the most important among all feature attributes and we will introduce the new method to present it in the following section. In order to get the correlation and statistical information in a certain time interval, the time window is used in the paper (120 seconds).

6.1.2 Quantization of TCP Flow State

There are nine flags (Fig. 4) involved in the connection establishment of TCP 3-way handshake protocol and the connection close of TCP 4-way handshake protocol. We design a 9-bit number to identify the connection state. The flag will be set to 1 if the corresponding flag is observed during the establishment-close process. Otherwise, the corresponding flag will set to 0. The decimal function $Sum(Flag_0, Flag_1, \ldots, Flag_8)$ with the non-superposed value is used to quantitate the whole connection state:

Table 2 Feature attributes of TCP flow feature vector.

Attribute	Describe
SrcIP	source IP address
DestIP	destination IP address
SrcPort	source port
DestPort	destination port
PktSize	average packet size in one TCP flow
SrcBytes	the number of bytes from source
DestBytes	the number of bytes from destination
FlowState	TCP flow closed state
Fre_SrcIP	frequency of a source IP in time-window
Fre_DestIP	frequency of a destination IP in time-window

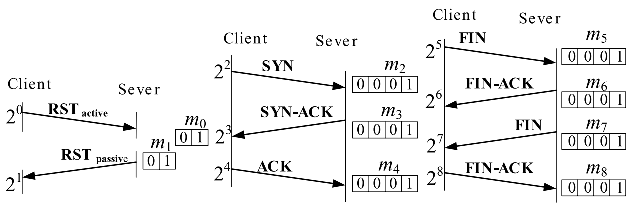

Fig. 4 Quantization of TCP flow state.

$$\begin{aligned}
FlowState &= Sum(Flag_0, Flag_1, \cdots, Flag_8) \\
&= \sum_{i=0}^{8} Flag_i \cdot 2^i = RST_{active} \cdot 2^0 \\
&\quad + RST_{passive} \cdot 2^1 + SYN \cdot 2^2 + ACK_{syn} \cdot 2^3 \\
&\quad + ACK \cdot 2^4 + FIN \cdot 2^5 + ACK_{fin} \cdot 2^6 \\
&\quad + FIN^* \cdot 2^7 + ACK^*_{fin} \cdot 2^8 \qquad (11)
\end{aligned}$$

According to Eq. (10), the TCP connection with the normal close can be described as follows: $Sum = (111111100)_2 = (508)_{10}$. For example, there are two instances of according to one SYN probing attack. The number 6 and 13 describe two abnormal TCP connection states in the scenarios of SYN probing attack.

- The target computer port is not open and responses a $RST_{passive}$: $Sum = RST_{passive} \cdot 2 + SYN \cdot 2^2 = 1 \times 2 + 1 \times 2^2 = 6$;
- The target computer port is open and responses an ACK_{syn}: $Sum = SYN \cdot 2^2 + ACK_{syn} \cdot 2^3 + RST_{active} \cdot 2^0 = 1 \times 2^2 + 1 \times 2^3 + 1 = 13$.

Consider the fact that the certain flag repeats in one TCP flow because of TCP retransmissions, we substitute the 32-bit number (4 bytes) for the 9-bit number, as in Fig. 4. But if the occurrence time of one certain flag is less than 15, the sum values can not repeat. The number of occurrence time will take value of 15 if it exceeds 15. $RST_{passive}/RST_{active}$ occurrence time is less than 3).

The TCP flow state is the most important feature attribute. Many attacks can result in the abnormal state of connection. Generally, 14 connection-closing states are summarized into one feature attribute in anomaly detection approach based on data mining [20]. We find that the method of tracking 14 connection states is clumsy relatively in real data preprocess program. What is more important, these 14 states cannot include all the complicated instances of TCP connection state. By the state quantization, every state of connection can be easy mapped to an int data range affording the state synopsis attribute directly leading to the whole improvement of feature vector.

6.2 Data Set and Parameters

6.2.1 Data Set

The Intrusion Detection Evaluation Data Set of 1999 DARPA [33], [34] has been widely used in the community of intrusion detection. The full detail description of the data set is given in [33]. The whole data set of DARPA 1999 includes five week data. The data sets of Week1, Week2 and Week3 are training data sets for different intrusion detection model. Week1 and Week3 data sets have entirety normal network traffic logs attack free and Week2 data set has labeled attack data in order to train the supervised learning classifiers. So the Week2 is not suit in our experiments. We utilize the data sets of Week1 and Week3 as our training

Table 3 Data set statistics (120s time-window).

Week	Date	TCP Flow Num	Total
Week1	Mon	38,493	223,790
	Tue	45,131	
	Wed	44,628	
	Thu	59,461	
	Fri	36,077	
Week3	Mon	42,502	317,258
	Tue	48,533	
	Wed	62,798	
	Thu	18,207	
	Ex Mon	11,031	
	Ex Tue	47,556	
	Ex Wed	86,631	
Week4	Mon	16,643	194,767
	Tue	53,121	
	Wed	44,141	
	Thu	49,999	
	Fri	30,863	
Week5	Mon	50,439	426,344
	Tue	90,687	
	Wed	43,904	
	Thu	128,499	
	Fri	112,815	

Table 4 Attacks in experiments.

Attack Name	Num
Back (DoS)	3
Selfping (Probing)	3
Portsweep (Probing)	12
Satan (Probing)	2
Mscan (Probing)	1
Ntinfoscan (Probing)	3
Apache2 (DoS)	3
Queso (Probing)	3
Neptune (DoS)	3
Processtable (DoS)	2
Total	35

data sets. The data sets of Week4 and Week5 are the test data sets including some attacks for detection. Table 3 gives the statistic information about the DARPA 1999 data set after our data preprocessing that we extract the TCP flow feature vectors from the original *outside.tcpdump* log files in the data set.

We examined only the *outside.tcpdump* logs to reduce the complexity of our experiments. The codebook of VQ is designed on the data sets attack free in week1 and week3. Consider the fact that the target of this paper is limited in the scope of TCP flows and not general, we test 35 instances of TCP attacks DoS and Probing mainly in the Week 4 and Week 5 as showed in Table 4. The experiment computer is Dawning Sever (Linux7.2/PIII800/RAM1G).

6.2.2 Parameter Selection in SOM

The parameters of SOM used to train codebooks can be referred in the following Table 5. Particularly, two dimension sizes of codebook (30×30 and 40×40) are selected to create the network traffic usage profile for comparisons.

Table 5 Parameters in SOM.

Parameter	Value
Topology	Hexa
Neighborhood function	Bubble
Dimension	30×30 40×40
Learning rate function	Linear
Training rate of first phase	0.5
Radius in first phase	20
Training length of first phase	50000
Training length of second phase	500000
Training rate of second phase	0.02
Radius in second phase	10

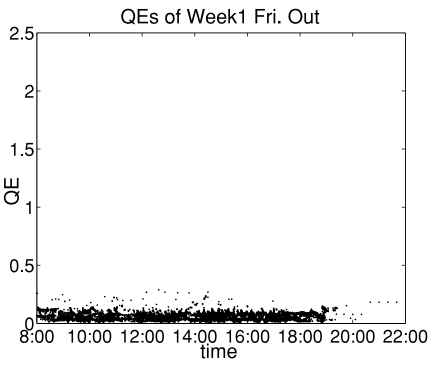

Fig. 5 *QEs* of week1 Fri.

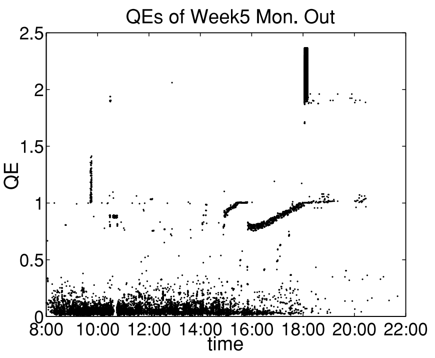

Fig. 6 *QEs* of week5 Mon.

6.3 Experiment Results

As given in the Sect. 4.3, we use *QEs* to evaluate the on-detecting network traffic feature vectors. Figures 5 and 6 presents the *QE* distributions (30 × 30 codebook) in outside traffic in DARPA data set. The DARPA data of week1 Fri. begins in 8:00 morning and ends about in 22:08 evening (22:04 about in week5 Mon.) after which few data can be observed in those two days [33].

Figure 5 just describes one part of *QE* distributions in the normal usage profile of training data. From the Fig. 5, we can observe that *QEs* are well regulated and don't change with the large deviations. The strong contrast in Fig. 6 is the sharp variations of *QEs* due to the high values of attack traffic *QEs*. Markedly, on 18:04, Neptune attack (a sort of SYN-flood DoS attack) can be viewed with *QE* values ranging from 1.8038 to 2.3063 in Fig. 6.

6.3.1 Feature Attribute Contribution

To better understand the feature attributes, we investigate of the attribute contribution in the attack detection. The contribution C_k of every eigenvector in input X is computed (Y_l is the nearest-neighbor codeword of X):

$$C_k = \frac{(x_k - y_{lk})^2}{\sum_{k=1}^{n}(x_k - y_{lk})^2} \cdot 100\% \quad (k = 1, 2, \ldots, n) \qquad (12)$$

The flow of Neptune (ID51.180445) is sampled and checked to get the quantitative contribution of every attribute in the process of distance deviation computation. For the flow "10.20.30.40:40029 → 172.016.112.050:366", the codeword Y_{33} is the nearest-neighbor, QE=2.031091, $QE^2 = \sum_{k=1}^{n}(x_k - y_{33,k})^2 = 4.1246$. The contribution percentage of every feature attribute in the Neptune flow is showed in Fig. 7. Equally, Fig. 8 presents the feature attribute contributions in the Portsweep probing attack (ID51.094334).

From the two figures (Figs. 7 and 8), we can infer that the *SrcPort*, *FlowState*, *Fre_SrcIP* and *Fre_DestIP* play the determinate significant roles in attack detections. The *SrcIP* and the *DestIP* don't play the significant roles in the attack detection according to the DARPA dataset which is mainly based on the LAN with limited scopes of the *SrcIP* and the *DestIP* [34]. In the Neptune detection, the limited effect even can be neglected. At the beginning of design, we considered the characteristics of network traffic in general network environments so that the feature attributes of the *SrcIP* and the *DestIP* are included to the feature vector.

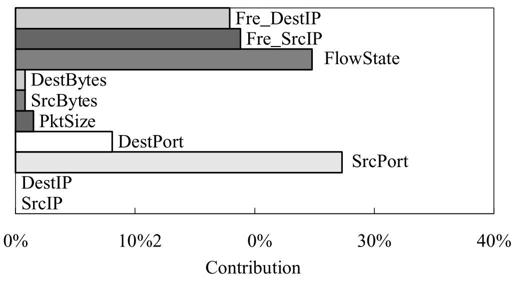

Fig. 7 Feature attribute contributions in Neptune.

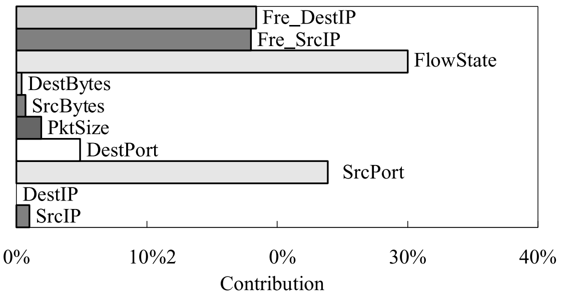

Fig. 8 Feature attribute contributions in Portsweep.

It is necessary to design the different feature vectors according to the different application network environments in the network IDS.

6.3.2 Codebook of Normal Usage Profile

In terms of creating of normal traffic usage profile, the usage (hits) of every node (codeword) in the codebooks of 30×30 and 40×40 can be viewed from the Fig. 9(I-IV). The hit of every codeword in the codebook illustrates the number of similar vectors attached to the same sub-profile.

From Fig. 9, we can infer that the codebooks divide the train data space equably by similarity measures of network traffic usage, where the hit spikes in the 4 figures exhibit the certain common trends. We can be confirmed that the codebook has the ability to picture the usage profile of network traffic stably: the different initializations can not affect the trends of usage profile presented in the codebook. In conclusion, the codebook of VQ does well in creating the usage profile of network traffic.

6.3.3 Detection Rates and False Positive Rates

In Table 6, the whole detailed detection rates (DRs) and

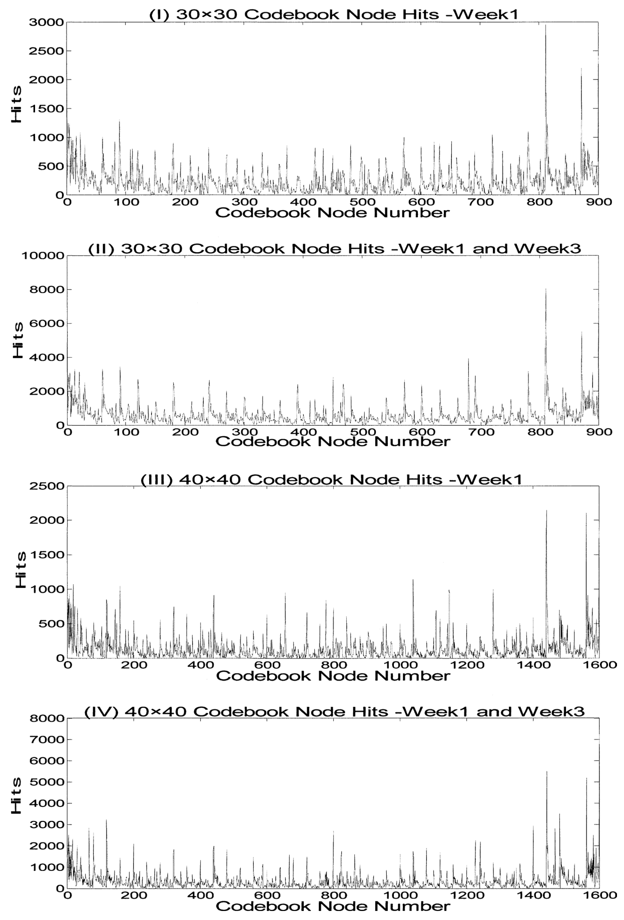

Fig. 9 Usage (hits) of every codeword in codebooks with different sizes in training phase.

Table 6 Thresholds for codebook 40 × 40 and 30 × 30.

Threshold(QE)	0.92	0.88	0.84	0.78	0.76	0.72	0.68	0.66
40 × 40 DR (%)	45.67	51.23	84.12	96.12	98.03	98.63	99.23	100
40 × 40 FPR(%)	0.61	0.63	0.78	0.91	1.03	2.04	3.21	10.32
Threshold(QE)	0.96	0.92	0.88	0.82	0.80	0.76	0.72	0.70
30 × 30 DR(%)	44.13	46.32	81.11	94.12	96.62	97.87	98.00	100
30 × 30 FPR(%)	0.81	0.83	0.89	1.09	1.12	2.42	3.74	18.87

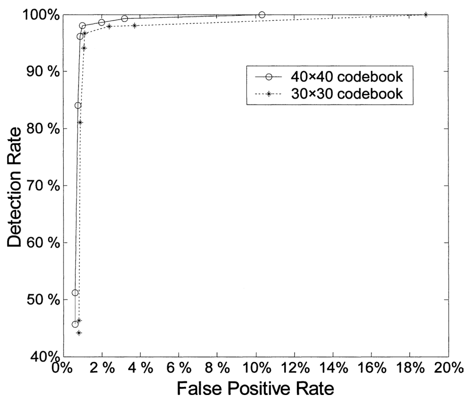

Fig. 10 ROC curves for the codebook of 40 × 40 and 30 × 30.

false positive rates (FPRs) are presented according the different thresholds (QEs). By varying detection thresholds, the Receiver Operating Characteristic (ROC) curves of DR and FPR are illustrated in Fig. 10 for fair comparisons of two different sizes of codebook. Figure 10 shows that the DR is trend to 100% fast by changing the threshold smaller; but the FPR also rises too. In the ROC curves, the eight points in inflexion areas suggest the ideal situations with the higher DRs and the lower PFRs. Table 6 just presents such situations where two codebooks achieve the better performance with DR-FPR-ε (98.03%, 1.03%, 0.76) and (96.62%, 1.12%, 0.80). Adjusting detection thresholds is necessary for the certain sizes of codebook in order to achieve the tradeoff outcomes with some selected parameters.

The overall outcome of evaluation is present in Table 7. It descries the detailed attack DR and FPR circumstance separately using the different codebook sizes of 30 × 30 and 40 × 40. Obviously, the larger size of the codebook can achieve the better attack detection result, the higher DR and the lower FPR. Increasing the codebook size can achieve the better outcome because the codebook with power resolution can represent profile in the fine granularity, which is very important for similarity measures between feature vectors. Contrarily, the lower resolution with very limited numbers of nodes or codewords can result in the NN search errors and the higher FPR.

Table 7 Detailed attack DRs and FPRs.

Date	Attack ID	Attack Name	DR 40×40	DR 30×30	FPR 40×40	FPR 30×30
W4 Mon	41.091531	Portsweep	2/3(66.7%)	1/3(33.3%)	0.27%	0.31%
	41.111531	Portsweep	5/5(100%)	5/5(100%)		
	41.122222	Portsweep	4/5(80%)	4/5(80%)		
	41.162715	Portsweep	4/10(40%)	0/10(0%)		
W4 Wed	43.080401	Satan	12/16(75%)	7/16(43.8%)	0.11%	0.11%
	43.164334	Portsweep	3/3(100%)	3/3(100%)		
W4 Thu	44.080000	Ntinfoscan	9/15(60%)	9/15(60%)	0.12%	0.14%
W4 Fri	45.111010	Portsweep	5/8(62.5%)	0/8(0%)	0.76%	0.96%
	45.181010	Portsweep	5/5(100%)	5/5(100%)		
W5 Mon	51.084334	Portsweep	3/3(100%)	3/3(100%)	2.87%	3.15%
	51.094334	Portsweep	93/106 (87.9%)	90/106 (84.9%)		
	51.102700	Apache2	955/1014 (95.5%)	948/1014 (93.5%)		
	51.140100	Apache2	11/12(91.7%)	11/12(91.7%)		
	51.180445	Neptune	19752/20480 (96.8%)	19731/20480 (96.3%)		
	51.201715	Selfping	1/1(100%)	1/1(100%)		
W5 Tue	52.094514	Selfping	0/1(0%)	0/1(0%)	3.18%	3.25%
	52.103409	Back	47/47(100%)	47/47(100%)		
	52.113855	Neptune	40955/40960 (99.9%)	40950/40960 (99.9%)		
	52.165435	Queso	7/7(100%)	7/7(100%)		
	52.181637	Neptune	986/1024 (96.3%)	950/1024 (92.7%)		
W5 Wed	53.045454	Selfping	5/5(100%)	5/5(100%)	0.33%	0.37%
	53.102617	Back	40/40(100%)	40/40(100%)		
	53.110516	Queso	5/7(71.4%)	4/7(57.1%)		
	53.123735	Portsweep	10/13(76.9%)	8/13(61.5%)		
	53.134015	Queso	5/7(71.4%)	4/7(57.1%)		
	53.150110	Process-table	183/375 (48.8%)	124/375 (33.1%)		
	53.152648	Back	17/40(42.5%)	0/40(0%)		
	53.171350	Apache2	175/340 (51.5%)	155/340 (45.6%)		
	53.195130	Portsweep	100/100 (100%)	100/100 (100%)		
W5 Thu	54.103459	Portsweep	3/3(100%)	3/3(100%)	0.59%	0.67%
	54.110416	Ntinfoscan	6/16(37.5%)	5/16(31.2%)		
	54.145832	Satan	8932/9129 (97.9%)	8817/9120 (96.7%)		
	54.183002	Ntinfoscan	8/17(47.1%)	6/17(35.3%)		
	54.195951	Mscan	5724/5724 (100%)	5724/5724 (100%)		
Total DR and FPR			98.034%	96.623%	1.03%	1.12%

6.3.4 Computation Cost Comparison

(1) Basic VQ and Fast VQ

In the paper, we implement the AID system based on VQ with the speedup strategy which can be regarded as the fast VQ. The speedup strategy of similarity measures accelerates the training of codebook using SOM and functions to reduce the computation cost of detection. We evaluate the difference of computation cost between the basic VQ (SOM without the speedup strategy) and the fast VQ (SOM with the speedup strategy). It is also the comparison between the method that paper [23] proposed and our methods in Fig. 11. In addition to the data process, the whole difference between the paper [23] and ours is the speedup strategy and the dimension of codebook.

Besides the codebooks of 40 × 40 and 30 × 30, we use the additional codebook 25×25. Figure 11 shows the different performances of VQ in codebook training time and detection time using various codebook sizes. The comparison outcome is very impressive because of computing time cost

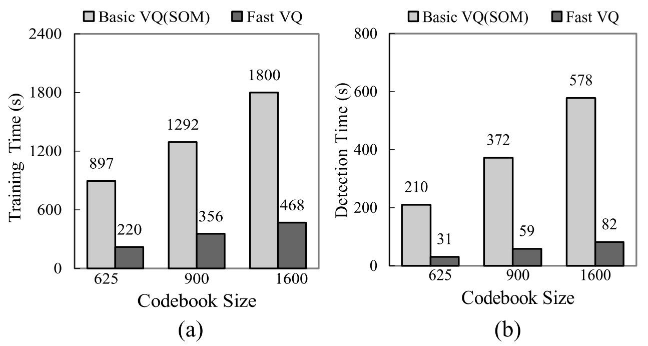

Fig. 11 Time cost comparison between basic VQ (SOM) and fast VQ using three codebooks with different sizes (a) training time cost comparison; (b) detection time cost comparison.

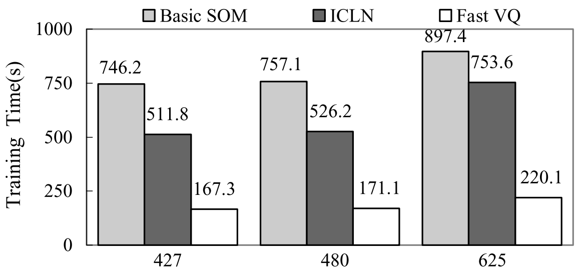

Fig. 12 Time cost comparison between basic SOM, ICLN and fast VQ.

reducing greatly. Recall that the train data sets have 541,048 TCP flows and the test data sets have 621,111 TCP flows, the results of detection and training time cost of fast VQ with the speedup strategy are strongly impressed. With 621,111 TCP flows according to the codebook size of 40×40, the detection time is only 82 seconds, which can illuminate the efficiency of fast VQ used in AID system. The fast VQ does well in ensuring the on-line style of detection. The detection speed reaches about 7575 units every second which outperforms greatly the SVM classifier detection speed, 3445 units every second in [35] even omitting the effect of different data dimensions. The comparison between our method and SVM in validates the higher performance of fast VQ used in our AID system.

(2) SOM, ICLN and Fast VQ

Different methods from the paper [23], [24] are compared directly. In Fig. 12, according to the training time, we compare ICLN [24] to SOM [23] and the fast VQ with the 541,048 training TCP flows. We set the same parameters of 432 nodes (16×27 codebook) used in the paper [23]. Our experiment results validate that the fast VQ does outperform SOM and ICLN in items of the computation time cost regarding the same node number. Furthermore, just because of the speedup strategy and the lower computation cost, we have more freedom to choose the larger codebook with more nodes (1600 nodes in our methods and 432 nodes in paper [23] and only 20 initial nodes in the paper [24]) to increase the resolution of codebook. The factors result in NN search errors can be reduced extensively. Therefore, the speedup strategy is also necessary to assure the higher DR and lower FPR.

With only 20 nodes in ICLN and SOM, the paper [24] shows the large difference of time cost between ICLN and SOM. But the situations in our experiments are not exactly coincident with the paper [24]. One reason can explain the inconsistency is that increasing the node number will decrease the difference of time cost between ICLN and SOM.

7. Conclusions and Future Work

In this paper, we focus on enhancing the overall performances of AID system to achieve more efficiency and usability for the high speed network, which is usually neglected at present. In order to reduce the computation cost, the speedup strategy of similarity measures is implemented to satisfy the need of real-time detection style.

(1) The vector similarity measures via the Euclidian distance computations are the element operations which are used extensive frequently in the algorithm training phase. The Euclidian distance computations are the first important basic operations in many machine algorithms, which are also the main computational cost bottleneck especially when confronted with the huge volume of training data. Consider this case, we design and implement the speedup strategy in the AID system to eliminate the unnecessary vector similarity measures to overcome the computational cost bottleneck.

(2) Because we utilize the large training dataset (541,048) to train algorithms in the network based AID, it is even more significant that the speedup strategy can assure the case that we have more freedom to set much more nodes in algorithms and enlarge the resolution of profiles to recapitulate the dynamic and complex usage behaviors of network traffic. With the larger structural codebook, the normal usage profile of the temporal-spatial network traffic could be constructed quantitatively well to describe the normal usage behaviors.

The performance evaluation experiments have clearly confirmed that our AID system could achieve the higher performance compared to other researches. In the future work, some optimization algorithms used in other disciplines can be applied to improve the AID performance further. Besides network traffic, our AID system can be extended to different concerned audit objects using the appropriate data preprocess.

Acknowledgments

The work reported in this paper is supported in part by the National Natural Science Foundation of China (Grant no.60403033) and the National High Technology Development Program of China 863 (Grant no.2002AA142020).

References

[1] T. Verwoerd and R. Hunt, "Intrusion detection techniques and approaches," Comput. Commun., vol.25, no.15, pp.1356–1365, Sept. 2002.

[2] J.M. Estvez-Tapiador, P. Garcia-Teodoro, and J.E. Diaz-Verdejo, “Anomaly detection methods in wired networks: A survey and taxonomy,” Comput. Commun., vol.27, no.16, pp.1569–1584, Oct. 2004.
[3] A. Gersho and R.M. Gray, Vector Quantization and Signal Compression, Kluwer Academic Publishers, Boston, 1992.
[4] http://www.snort.org,2004
[5] D.E. Denning, “An intrusion-detection model,” IEEE Trans. Softw. Eng., vol.13, no.2, pp.222–232, Feb. 1987.
[6] N. Ye, “Multivariate statistical analysis of audit trails for host-based intrusion detection,” IEEE Trans. Comput., vol.51, no.13, pp.810–820, Jan. 2002.
[7] S.M. Emran and N. Ye, “Robustness of chi-square and Canberra techniques in detecting intrusions into information systems,” Qual. Reliab. Eng. Int., vol.18, no.1, pp.19–28, Jan. 2002.
[8] E. Eskin, A. Arnold, and M. Prerau, “A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled-data,” in Applications of Data Mining in Computer Security, Kluwer, 2002.
[9] S.H. Oh and W.S. Lee, “A clustering-based anomaly intrusion detection for a host computer,” IEICE Trans. Inf. & Syst., vol.E87-D, no.8, pp.2086–2094, Aug. 2004.
[10] L. Portnoy, E. Eskin, and S.J. Stolfo, “Intrusion detection with unlabeled data using clustering,” Proc. ACM CSS Workshop on Data Mining Applied to Security, pp.123–130, Philadelphia, PA, June 2001.
[11] E. Eskin, “Anomaly detection over noisy data using learned probability distributions,” Proc. 17th Int. Conf. Machine Learning, pp.255–262, San Francisco, CA, July 2000.
[12] E. Knorr and R. Ng, “Algorithms for mining distance-based outliers in large datasets,” Proc. VLDB Conference, pp.392–403, New York, USA, Sept. 1998.
[13] A. Lazarevic, L. Ertoz, A. Ozgur, and J. Srivastava, “A comparative study of anomaly detection schemes in network intrusion detection,” Proc. SIAM Conf. Data Mining, pp.108–120, Orlando, FL, May 2003.
[14] S. Mukkamala, G. Janowski, and AH. Sung, “Intrusion detection using neural networks and support vector machines,” Proc. Hybrid Information Systems Advances in Soft Computing, pp.121–138, Physica/Springer, Heidelberg, Dec. 2001.
[15] A.H. Sung and S. Mukkamala, “Identifying important features for intrusion detection using support vector machines and neural networks,” Proc. 2003 Symposium on Applications and the Internet, pp.209–217, Orlando, USA, Jan. 2003.
[16] Y. Qiao, X.W. Xin, Y. Bin, and S. Ge, “Anomaly intrusion detection method based on HMM,” Electron. Lett., vol.38, no.13, pp.663–664, June 2002.
[17] J.M. Bonifaco and E.S. Moreira, “An adaptive intrusion detection system using neural network,” Research Report, UNESP, Brazil, 1997.
[18] C. Warrender, S. Forrest, and B. Pearlmutter, “Detecting intrusions using system calls: Alternative data models,” Proc. 1999 IEEE Symposium on Security and Privacy, pp.133–145, Oakland, USA, May 1999.
[19] S. Mukkamala, G.I. Janoski, and A.H. Sung, “Intrusion detection using support vector machines,” Proc. High Performance Computing Symposium - HPC 2002, pp.178–183, San Diego, USA, April 2002.
[20] L. Wenke, S.J. Stolfo, and K.W. Mok, “A data mining framework for building intrusion detection models,” Proc. 1999 IEEE Symposium on Security and Privacy, pp.296–304, Oakland, USA, May 1999.
[21] Y. Liao and V. Rao Vemuri, “Use of k-nearest neighbor classifier for intrusion detection,” Comput. Secur., vol.21, no.5, pp.439–448, Oct. 2002.
[22] L.P. Cordella, A. Limongiello, and C. Sansone, “Network intrusion detection by a multi-stage classification system,” Proc. Multiple Classifier Systems: 5th International Workshop, MCS 2004, pp.324–333, Cagliari, Italy, June 2004.
[23] M. Ramadas, S. Ostermann, and B. Tjaden, “Detecting anomalous network traffic with self-organizing maps,” Proc. Recent Advances in Intrusion Detection: 6th International Symposium, RAID 2003, pp.34–56, Pittsburgh, PA, USA, Sept. 2003.
[24] J.Z. Lei and A. Ghorbani, “Network intrusion detection using an improved competitive learning neural network,” Proc. Second Annual Conference on Communication Networks and Services Research (CNSR’04), pp.190–197, Fredericton, N.B., Canada, May 2004.
[25] Y. Linde, A. Buzo, and R.M. Gray, “An algorithm for vector quantizer design,” IEEE Trans. Commun., vol.28, no.1, pp.84–95, Jan. 1980.
[26] K. Goh and E. Chang, “Indexing multimedia data in high-dimensional and weighted feature spaces,” Proc. 6th Visual Database Conference, pp.127–140, Brisbane, Australia, May 2002.
[27] H. Ferhatosmanoglu, E. Tuncel, D. Agrawal, and A. El Abbadi, “Approximate nearest neighbor searching in multimedia databases,” Proc. 17th IEEE Int. Conf. on Data Engineering, pp.503–511, Heidelberg, Germany, April 2001.
[28] H. Teng, K. Chen, and S. Lu, “Adaptive, real-time anomaly detection using inductively generated sequential patterns,” Proc. IEEE Symposium on Research in Security and Privacy, pp.278–284, Oakland, USA, May 1990.
[29] N. Ueda and R. Nakano, “A new competitive learning approach based on an equidistortion principle for designing optimal vector quantizers,” IEEE Trans. Neural Netw., vol.7, no.8, pp.1211–1227, Sept. 1994.
[30] T. Kohonen, Self-Organization Maps, 3rd ed., Springer-Verlag, Berlin, 2000.
[31] Z. Lu, J. Pan, and S. Sun, “A fast codebook search algorithm for vector quantization,” Acta Electronica Sinica, vol.28, no.2, pp.133–135, Feb. 2000.
[32] http://libnids.sourceforge.net/
[33] http://www.ll.mit.edu/IST/ideval/index.html
[34] R. Lippmann, J.W. Haines, D.J. Fried, J. Korba, and K. Das, “The 1999 DARPA off-Line intrusion detection evaluation,” Comput. Netw., vol.34, no.4, pp.579–595, Oct. 2000.
[35] S. Mukkamala, A.H. Sung, and A. Abraham, “Intrusion detection using an ensemble of intelligent paradigms,” J. Netw. Comput. Appl., vol.28, no.2, pp.167–182, Feb. 2005.

Jun Zheng received his M.E. degree in Mechatronics Engineering from the Harbin Insitute of Technology (HIT), China, in 2001. He is currently a PhD candidate of the Research Center of Computer Network and Information Security Technology, the Department of Computer Science and Engineering, HIT. His research interests include network security, Quality-of-Service for secure networking and high-speed network traffic analysis.

Mingzeng Hu is a senior professor and doctor supervisor of the Department of Computer Science and Engineering in Harbin Institute of Technology (HIT), China. In 1989–1995, he was a Visiting Scholar and Researcher in York University, Canada. He is now the main director of the Research Center of Computer Network and Information Security Technology, HIT. His current research interests include advanced computer architecture, parallel computing and network security.