Signature Based Intrusion Detection using Latent Semantic Analysis

Jean-Louis Lassez, Ryan Rossi, Stephen Sheel
Department of Computer Science
Coastal Carolina University
{jlassez, raross, steves}@coastal.edu

Srinivas Mukkamala
Department of Computer Science
New Mexico Tech
srinivas@cs.nmt.edu

Abstract

We address the problem of selecting and extracting key features by using singular value decomposition and latent semantic analysis. As a consequence, we are able to discover latent information which allows us to design signatures for forensics and in a dual approach for real-time intrusion detection systems. The validity of this method is shown by using several automated classification algorithms (Maxim, SVM, LGP). Using the original data set we classify 99.86% of the calls correctly. After feature extraction we classify 99.68% of the calls correctly, while with feature selection we classify 99.78% of the calls correctly, justifying the use of these techniques in forensics. The signatures obtained after feature selection and extraction using LSA allow us to classify 95.69% of the calls correctly with features that can be computed in real time. We use Support Vector Decision Function and Linear Genetic Programming for feature selection on a real data set generated on a live performance network that consists of probe and denial of service attacks. We find that the results reinforce our feature selection method.

1. Introduction

Latent Semantic Analysis (LSA) has been successfully used in applications such as speech recognition, natural language processing, cognitive modeling, document classification and cross language information retrieval [1-7].

LSA is based on Singular Value Decomposition (SVD) which has many applications [8-12].

In particular an early and major use of SVD is in noise removal and dimension reduction. This approach has also been applied to intrusion detection, for instance it is shown that substantial dimension reduction can be achieved while maintaining the performance of a kNN classifier [8].

Therefore it is natural to see how we can extend the latent semantic analysis capabilities of SVD to the case of intrusion detection.

In the next section we give examples of latent relations between calls and features such as synonymy, polysemy, hypernymy and hyponymy by analyzing the principal directions. These relations give us information such as potentially redundant or undesirable features, detection of calls susceptible to create false positives or obfuscation. In the third section, we perform feature selection and extraction by using the notions of Latent Semantic Analysis. We select features relying on the principal direction of the calls. This allows us to rank the features in accordance to their contribution. From the principal directions of calls we are able to find signatures for each class. Such signatures can form the basis of a real-time intrusion detection system. We then perform feature extraction by using the principal directions of the features. These directions can be used for forensics. Validation of the signatures, features selected and extracted is seen in the fourth section using automated classification algorithms. We use SVDF and LGP for feature selection and on a real data set generated from a live performance network.

2. Discovery of Latent Relations

The DARPA intrusion detection data set is used for offline analysis. In the DARPA intrusion detection evaluation program, an environment was set up to acquire raw TCP/IP dump data for a network by simulating a typical U.S. Air Force LAN. The LAN was operated like a real environment, but being blasted with multiple attacks [14,15]. The data consists of vectors, representing network connections, for which numerical measures have been collected. For each TCP/IP connection, 41 various quantitative and qualitative features were extracted [16] for intrusion analysis. Attacks are classified into the following categories: DOS (denial of service), R2L (unauthorized access from a remote machine), U2Su (unauthorized access to local super user privileges) and Probing (surveillance).

From the forty-one features of the data set, the first eight are basic features of TCP connections and are

computed in real-time [10]. The remaining features that are not computed in real-time can be used for forensics.

Features			
1.	Duration	**22.**	is_guest_login
2.	protocol type	**23.**	count
3.	service	**24.**	srv_count
4.	flag	**25.**	serror_rate
5.	src_bytes	**26.**	srv_serror_rate
6.	dst_bytes	**27.**	rerror_rate
7.	land	**28.**	srv_rerror_rate
8.	wrong_fragment	**29.**	same_srv_rate
9.	urgent	**30.**	diff_srv_rate
10.	hot	**31.**	srv_diff_host_rate
11.	num_failed_logins	**32.**	dst_host_count
12.	logged_in	**33.**	dst_host_srv_count
13.	num_compromised	**34.**	dst_host_same_srv_rate
14.	root_shell	**35.**	dst_host_diff_srv_rate
15.	su_attempted	**36.**	dst_host_same_src_port_rate
16.	num_root	**37.**	dst_host_srv_diff_host_rate
17.	num_file_creations	**38.**	dst_host_serror_rate
18.	num_shells	**39.**	dst_host_srv_serror_rate
19.	num_access_files	**40.**	dst_host_rerror_rate
20.	num_outbound_cmds	**41.**	dst_host_srv_rerror_rate
21.	is_host_login		

Table 1. Features of the DARPA data set.

The rows in the data set represent network connections or calls and the columns represent features of the network connections [8, 10].

Let $M \in \Re^{nxm}$, we decompose M into three matrices using Singular Value Decomposition:

$$M = U\,S\,V^T$$

where $U \in \Re^{nxm}$, $S \in \Re^{mxm}$ and $V^T \in \Re^{mxm}$. The matrix S contains the singular values located in the $[i,i]_{1,..,n}$ cells in decreasing order of magnitude and all other cells contain zero. The eigenvectors of MM^T make up the columns of U and the eigenvectors of M^TM make up the columns of V. The matrices U and V are orthogonal, unitary and span vector spaces of dimension n and m, respectively. The inverses of U and V are their transposes.

$$\begin{bmatrix} | & | & & | \\ d_1^f & d_2^f & \cdots & d_k^f \\ | & | & & | \end{bmatrix} \begin{bmatrix} s_1 & 0 & 0 & 0 \\ 0 & s_2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & s_k \end{bmatrix} \begin{bmatrix} — & d_1^c & — \\ — & d_2^c & — \\ & \vdots & \\ — & d_k^c & — \end{bmatrix}$$

$$U \qquad\qquad S \qquad\qquad V^T$$

The columns of U are the *principal directions of the features* and the rows of V^T are the *principal directions of the calls*. The principal directions are ordered according to the singular values and therefore according to the importance of their contribution to M.

The singular value decomposition is used by setting some singular values to zero, which implies that we approximate the matrix M by a matrix:

$$M_k = U_k\,S_k\,V_k^T$$

A fundamental theorem by Eckart and Young states that M_k is the closest rank-k least squares approximation of M [10]. This theorem can be used in two ways. To reduce noise by setting insignificant singular values to zero or by setting the majority of the singular values to zero and keeping only the few influential singular values in a manner similar to principal component analysis.

In latent semantic analysis we extract information about the relationships between calls and features as they change when we set all, but the most significant, singular values to zero. The singular values in S provide contribution scores for the principal directions in U and V^T.

We use the terminology "principal direction" for the following reason. In zoomed clusters it was shown that (assuming unit vectors) the principal eigenvector is an "iterated centroid" that is a version of the notion of centroid, where outliers are given a lower weight [17]. Furthermore, in text analysis it is usual to consider that the main information is provided by the direction of the vectors rather than by their length.

The similarities between calls and features are measured by the cosine of the angle between vectors representing calls or features, as is done in Information Retrieval [5]. *Similarities that do not appear in M but appear in M_k are called latent relations.* They appear in M_k either because the noise has been removed or because major components were hidden by less important ones. Therefore in order to find synonymy, polysemy, hypernymy and hyponymy relations between calls or features, we use cosine scoring in M and M_k.

Finding synonymy between the features is important as it identifies redundancy. As an example, using the normal calls, we find that the tenth feature is not similar to any other. However, when we use M_k where k = 3, we find that the vectors of the tenth and sixteenth features have a cosine of .95, which suggests potential redundancy.

This relation was hidden in M, but when we reduce the dimensionality, the latent structure of the data becomes visible. The validation of this method to find synonymous features is seen in section 5 where we select a minimal number of features while keeping a very high accuracy of classification.

We can also find polysemy relations between the features in M_k that are not apparent in M. For instance, the sixteenth and seventeenth feature have a cosine of .90 while the seventeenth and tenth feature have a cosine of .75.

$$10 \approx 16 \approx 17$$
$$10 \neq 17$$

The sixteenth feature is seen to be more general than the tenth and seventeenth features as it can replace both features, while the tenth and seventeenth features cannot replace each other. This shows a case of hypernymy and hyponymy.

Finding polysemy between calls is important to identify obfuscation. As an example of polysemy, we find that in M, a normal call and U2R call are similar with a cosine of .91. However, when we use M_k with k = 4, we find that the normal call and U2R call are not similar with a cosine of .76. This example indicates obfuscation in M, but when we use M_k the obfuscation becomes more evident and we might be able to catch the attack.

These remarks warrant a more systematic study.

3. Feature Selection and Extraction by Principal Directions

We now analyze the features to find which contribute to each category of attacks.

For each class, we consider the first two principal directions (rows of V^T). There are 41 columns which represent the features. If a feature has very similar values in the two principal directions for all classes we discard that feature. Indeed, the values in an arbitrary principal direction that are very similar cannot be used to discriminate features.

There are two types of features, useless and redundant. Useless features are features which appear useful in M, but in M_k they are found to have no impact on the data and thus can be thrown out. However with synonymous/redundant features, we have to keep at least one of the features. In Table 2 below we find that the twenty-third and twenty-fourth features are synonmous, so we only keep one of them. This also applies to the thirty-second and thirty-third features. This leads to selecting only the 1, 5, 6, 23 and 33 features which can be found in Table 2.

		Features				
Class	**PD**	**1**	**5**	**6**	**23**	**33**
Norm	**1**	0	0.01	1.00	0	0
	2	0	1.00	-0.01	0	0
DOS	**1**	0	-0.99	-0.14	-0.02	-0.01
	2	0	0.01	0.17	-0.63	-0.33
U2R	**1**	-0.01	-0.05	-1.00	0	0
	2	-0.04	-1.00	0.05	0	-0.01
R2L	**1**	0	-1.00	0	0	0
	2	0	0	-1.00	0	0
Probe	**1**	0	-1.00	0	0	0
	2	-1.00	0	0	0	0.01

Table 2. Signatures of the classes using the first two principal directions.

Each row can be used as a signature for its corresponding class. For instance, we can choose as signature for the normal class:

0	0.01	1.00	0	0

as it is significantly different from all potential signatures of the other classes.

In another experiment we use the principal directions to select and rank the features of the classes according to feature contribution. We look at the first four principal directions of each class. From this, we chose the feature in every direction whose absolute value is substantially higher than the others. As an example, in the first principal direction of the Normal class we find that the highest absolute value is in the sixth feature. We can tell how significant this feature is by looking at the corresponding singular values. For example, in DOS the thirty-second feature is not considered significant because the singular value associated to the third principal direction is relatively smaller than the others. In Table 3, we have italicized/bolded the most significant features for each class.

Principal Directions	**Feature Ranking**				
	Normal	**DOS**	**U2R**	**R2L**	**Probe**
1	***6***	***5***	***6***	***5***	***5***
2	***5***	***23***	***5***	***6***	***1***
3	***1***	32	***32***	1	23
4	33	6	***1***	32	32

Table 3. Feature selection and ranking by the first four principal directions

Principal Directions	**Feature Ranking**	
	Normal	**Attacks**
1	***6***	***5***
2	***5***	***6***
3	***1***	***24***
4	33	1

Table 4. Feature selection and ranking by grouping the attacks and using the first four principal directions

From Table 4, we see that the fifth and sixth features contribute the most to the attacks and normal calls, respectively.

We have used the principal direction of the calls to find signatures which can form the basis of a real-time intrusion detection system. However, the singular value decomposition also provides us with a dual approach. That is we now consider the principal directions of the features; those can be used for forensics.

Therefore, we perform feature extraction by only considering the matrices U_k and S_k of the singular value decomposition.

$$A_k = U_k \; S_k$$

We use the first k columns of US as the new features since these are the principal directions of the features with the scaling in S applied to the directions. In section 5, we use a simple machine learning algorithm called Maxim to validate our feature extraction and selection results.

3.1 Support Vector Based Feature Selection

It is of great interest and use to find exactly which features underline the nature of connections of various classes. This is precisely the goal of data visualization in data mining. The problem is that the high-dimensionality of data makes it hard for human experts to gather any knowledge. If we knew the key features, we could greatly reduce the dimensionality of the data and thus help human experts become more efficient and productive in learning about network intrusions [19].

The information about which features play key roles and which are more neutral is "hidden" in the SVM decision function [19,20]. The formula below is the formulation of the decision function in the case of using linear kernels.

$$F(X) = \langle W, X \rangle + b$$

The point X is predicted to be in class A or "positive class" if the F(X) is positive, and class B or "negative class" if F(X) is negative. We can rewrite the formula above to expand the dot product of W and X.

$$F(X) = \Sigma W_i X_i + b$$

One can see that the value of F(X) depends on the contribution of each factor, $W_i X_i$. Since X_i can take only $b \geq 0$, the sign of W_i indicates whether the contribution is towards positive classification or negative classification. The absolute size of W_i measures the strength of this contribution. In other words if W_i is a large positive value, then i^{th} feature is a key factor of "positive class" or class A. Similarly if W_i is a large negative value then i^{th} feature is a key factor of the "negative class" or class B. Consequently the W_i, which is close to zero, either positive or negative, carries little weight. The feature, which corresponds to this W_i, is said to be neutral feature and removing it has very little effect on the classification.

Having retrieved this information directly from SVMs' decision function, we rank the W_i, from largest positive to largest negative. This essentially provides the soft partitioning of the features into the key features of class A, neutral features, and key features of class B. We say soft partitioning, as it either depends on a threshold on the value of W_i, which will define the partitions, or the proportions of the features, which we want to allocate to each of the partitions. Both the threshold and the value of proportions can be set by the human expert.

Support Vector Decision Function Ranking:

The input ranking is done as follows: First the original data set is used for the training of the classifier. Then the classifier's decision function is used to rank the importance of the features. The procedure is:

1. Calculate the weights from the support vector decision function;
2. Rank the importance of the features by the absolute values of the weights;

SVDF feature ranking results are presented in Table 5. Classification accuracies using 6 most important features are presented in Table 4 [21].

SVM	Feature Ranking				
	Normal	**DOS**	**U2R**	**R2L**	**Probe**
1	6	23	5	32	5
2	32	24	1	3	33
3	12	39	2	1	23
4	34	25	12	23	2
5	4	36	4	24	24
6	29	38	29	29	4

Table 5. Feature selection and ranking SVMs' linear kernel

3.2 Linear Genetic Programming (LGP) Based Feature Selection

The performance of each of the selected input feature subsets is measured by invoking a fitness function with the correspondingly reduced feature space and training set and evaluating the intrusion detection accuracy. Once the required number of iterations is completed, the evolved high ranked programs are analyzed for how many times each input appears in a way that contributes to the fitness of the programs that contain them. The best feature subset found is then output as the recommended set of features to be used in the actual input for the classifier.

In the feature selection problem the main interest is in the representation of the space of all possible subsets of the given input feature set. Each feature in the candidate feature set is considered as a binary gene and each individual consists of fixed-length binary string representing some subset of the given feature set. An individual of length d corresponds to a d-dimensional binary feature vector Y, where each bit represents the elimination or inclusion of the associated feature. Then, $y_i = 0$ represents elimination and $y_i = 1$ indicates inclusion of the i^{th} feature. Fitness F of an individual program p is calculated as the mean square error (*MSE*) between the predicted output (O_{ij}^{pred}) and the desired output (O_{ij}^{des}) for all ***n*** training samples and ***m*** outputs [22].

$$F(p) = \frac{1}{n \cdot m} \sum_{i=1}^{n} \sum_{j=1}^{m} (O_{ij}^{pred} - O_{ij}^{des})^2 + \frac{w}{n} CE = MSE + w \cdot MCE$$

Classification Error (CE) is computed as the number of misclassifications. Mean Classification Error (MCE) is added to the fitness function while its contribution is proscribed by an absolute value of Weight (W).

LGP	Feature Ranking				
	Normal	DOS	U2R	R2L	Probe
1	10	23	14	22	35
2	6	13	39	19	27
3	5	8	17	6	31
4	13	7	25	11	12
5	40	12	36	12	3
6	3	30	1	3	5

Table 6. Feature selection and ranking using Linear Genetic Programming

LGP feature ranking results are presented in Table 6. We find that the features selected using LGP and SVM are similar to the features selected using the first four principal directions. These results reinforce our feature selection method [21].

4. Validation of Signatures

Here we use a simple machine learning algorithm called Maxim to verify our results and show the benefits of dimension reduction. It is a conceptually simpler and a more efficient alternative to Support Vector Machines for an arbitrary number of classes. Maxim has proven to perform essentially as well as or better than SVM for these types of problems [23].

We are interested in classifying a call X by comparing its similarity to a set of previously classified training calls. The call X will be assigned to the class whose calls are most similar to X. Given a set of I class-labeled training calls $\{X_i, \xi(X_i)\}$, i = 1..I, where $\xi(X_i)$ is the class of X_i, and for an unclassified call X, we define the class similarity of X with respect to a class C as

$$S_C(X) = \sum_{X_k \in C} \alpha_k s(X_k, X)$$

Where s is the similarity function and $\alpha_k \geq 0$ reflects the relative importance given to each X_k with respect to the classification. We can therefore predict the class of X using the following decision function:

$$\xi(X) = \arg c\{\max(S_c(X))\}$$

The classification results were striking. When using Maxim with the RBF kernel against the DARPA data set, we classify 99.86% of the calls correctly. When using only the first four principal directions of the features, we classify 99.68% of the calls correctly. We see that the reduction of dimensionality only affects the accuracy minimally.

Feature Extraction	Time (Sec)	Sigma	False Positives	Accuracy
41	17295	.003	≤ 0.14%	99.86%
4	10273	.003	≤ 0.32%	99.68%
2	8738	.003	≤ 1.76%	98.24%

Table 7. Classification results of the dimension and data reduction compared with the original data.

Feature Selection	Time (Sec)	Sigma	False Positives	Accuracy
5,6,1,23,33	10872	.003	≤ 0.22%	99.78%
5,6,1,23	10664	.003	≤ 0.44%	99.56%
5,6	9847	.003	≤ 1.98%	98.02%

Table 8. Classification results of the feature selection using the principal directions of the calls.

In Table 8 it is shown that by using our feature ranking method, one can select optimal features for classification. One can clearly see the fifth and sixth features play a significant role in the data set as we classify 98.02% of the calls correctly using simply these features. We demonstrate the twenty-third/twenty-fourth and the thirty-second/thirty-third features are redundant as it is shown that by selecting only one feature from either group, we can essentially achieve the accuracy of classification that was achieved using the entire data set. The fact that we have striking results by using the principal directions of the calls for selecting the features also validates our signatures found by using these directions. The principal directions can be seen as legitimate representatives of the classes.

These results verify our feature extraction, selection and signatures as we achieve a very high accuracy of classification while maintaining a low false positive & false negative rate.

We propose a signature based intrusion detection system by using the principal direction of the calls. Any incoming call c can be processed by using the cosine between the call and the signature. To simplify matters even further, one could simply use the principal direction of the normal calls such that if a threshold t is less than the cosine between the call and the signature, one can classify the call as an attack.

We use the signatures from Table 2 and compute the cosine between an incoming call, where we only consider the features that correspond to the signature. The features we used from the original data set are 1, 5, 6, 23 and 33.

Signature	Thre.	False Positives	False Negatives	Accuracy
Normal	.1	0.53%	3.77%	95.69%
DOS	.455	0.67%	1.26%	98.07%
U2R	.001	23.33%	0%	76.67%
R2L	.001	23.36%	0.22%	76.42%
Probe	.8	0.90%	0.81%	98.29%

Table 9. Signature validation

We used the first signature in Table 2 for Normal, R2L and U2R and the second signature for DOS and Probe. The calls from U2R and R2L are classified incorrectly between both classes.

4.1 Real-Time Data Collection and Feature Selection for Probe and DoS

Experiments are performed on a real network using two clients and the server that serves the New Mexico Tech Computer Science Department network. The clients had CIA (computational intelligent agent) installed on them to identify or detect probes that are targeted to the server we are protecting. Our primary goal in these experiments is to detect probes targeting the server we are trying to protect. Our network parser gives the summary of each connection made from a host to the server and constructs a feature set to input into a classifier for classification. The output from a classifier is either normal or probe for each connection. Nmap an open source tool is used to collect probe data [13]. Probing is a class of attacks where an attacker scans a network to gather information or find known vulnerabilities. An attacker with a map of machines and services that are available on a network can use the information to look for exploits. There are different types of probes: some of them abuse the computer's legitimate features; some of them use social engineering techniques. This class of attacks is the most commonly heard and requires very little technical expertise. Nmap is installed on the clients that have CIA installed. A variety of probes SYN stealth, FIN stealth, ping sweep, UDP scan, null scan, xmas tree, IP scan, idle scan, ACK scan, window scan, RCP scan, and list scan with several options are targeted at the server. Normal data included multiple sessions of ftp, telnet, SSH, http, SMTP, pop3 and imap. Network data originating from a host to the server that included both normal and probes is collected for analysis; for proper labeling of data for training the classifiers normal data and probe data are collected at different times.

We used the RBF (radial basis function) kernel function that defines the feature space in which the training set examples will be classified. Table 10 summarizes the results of probe detection on a real data set using SVMs and LGPs.

Class	SVM	LGP
Normal	99.75%	100%
Probe	99.99%	100%

Table 10. Probe detection accuracies on a real data set

A passive sniffer can be placed at the router to collect data for detecting DoS attacks. The architecture comprises of three components: a packet parser, classifier and a response module. The network packet parser uses the WINPCAP library to capture packets and extracts the relevant features required for DoS detection. The output of the parser includes the twelve DoS-relevant features as selected by SVDF described in section 3.1 [21].

The output summary of the parser includes the eleven features of duration of the connection to the target machine, protocol used to connect, service type, status of the connection (normal or error), number of source bytes, number of destination bytes, number of connections to the same host as the current one during a specified time window (in our case .01seconds), number of connections to the same host as the current one using same service during the past 0.01 seconds, percentage of connections that have SYN errors during the past .01 seconds, percentage of connections that have SYN errors while using the same service during the past .01 seconds, and percentage of connections to the same service during the past .01 seconds.

We experimented with more than 24 types of DoS attacks. In the experiments performed we used different types of DoS attacks: SYN flood, SYN full, MISFRAG, SYNK, Orgasm, IGMP flood, UDP flood, Teardrop, Jolt, Overdrop, ICMP flood, FIN flood, and Wingate crash, with different service and port options. Normal data included multiple sessions of http, ftp, telnet, SSH, http, SMTP, pop3 and imap. Network data originating from a host to the server that included both normal and DoS is collected for analysis; for proper labeling of data for training the classifier normal data and DoS data are collected at different times. Table 11 summarizes the results of DoS detection on a real data set using SVMs and LGPs.

Class	SVM	LGP
Normal	99.48%	95.26%
DOS	79.91%	94.28%

Table 11. DoS detection accuracies on a real data set

5. Conclusion

We showed how one can perform feature selection and extraction using the singular value decomposition paired with the notion of latent semantic analysis. From this, we can discover hidden information that allows us to design signatures for forensics and eventually real-time intrusion detection systems. Furthermore, it is shown that by using

Maxim, one can achieve stunning accuracy with low false positives and false negatives which also validates our feature selection, extraction and signatures. We use SVDF and LGP for feature selection and on a real data set generated from a live performance network. We find that the results reinforce our feature selection method. Finally, we give insight on how one can use signatures for a real-time signature based intrusion detection system.

6. Acknowledgements

We would like to acknowledge many insightful discussions with Dr. Andrew Sung that helped clarify our ideas and Madhukumar Shankarapani.

7. References

[1] T. K. Landauer, P. W, Foltz, D. Laham, "Introduction to Latent Semantic Analysis," *Discourse Processes,* **25**, 1998, 259-284.

[2] T. K. Landauer, M. L. Littman, "Fully automatic cross language document retrieval using latent semantic indexing," *Proc. of the Sixth Annual Conference of the UW Centre for the New Oxford English Dictionary and Text Research.*, 1990, 31-38.

[3] B. Lemaire, "Tutoring Systems Based on Latent Semantic Analysis," In S.P. Lajoie and M. Vivet (Eds.), *Artificial Intelligence in Education*, 1999, 527-534.

[4] J. R. Bellegarda, "Large vocabulary speech recognition with multispan statistical language models," *IEEE Transactions on Speech and Audio Processing*, 8(1), 2000, 76–84.

[5] T. K. Landauer, T. S. Dumais, "A Solution to Plato's Problem: The Latent Semantic Analysis Theory of the Acquisition, Induction, and Representation of Knowledge," *Psychological Review*, 104, 1997, 211-240.

[6] W. Kintsch, Comprehension: A Paradigm for *Cognition*. Cambridge University Press, 1998.

[7] S. Deerwester, T. S. Dumais, T. K. Landauer, G. W. Furnas, R. A. Harshman, "Indexing by latent semantic analysis," *JSIS*, 41(6), 1990, 391-407.

[8] S. Rawat, A. Pujari, V. Gulati, "On the Use of Singular Value Decomposition for a Fast Intrusion Detection System," *Electrical. Notes Theory Computer Science* 142, 2006, 215-228.

[9] O. Alter, P. O. Brown, D. Botstein, "Singular value decomposition for genome-wide expression data processing and modeling," Proceedings of National Academy of Sciences. USA, **97**, 2000, 10101–10106.

[10] Y. Melnikov, A. Tarakanov, "Immunocomputing model of intrusion detection," *LNCS*, 2776, Springer, 2003, 453–456.

[11] D. V. S. Chandra, "Digital image watermarking using singular value decomposition," *IEEE Midwest Symposium on Circuits and Systems*, 2002, 264-267.

[12] M. Vasilescu and D. Terzopoulos, "Multilinear image analysis for facial recognition," *ICPR*, 2002, 511-514.

[13] A. H. Sung, S. Mukkamala, "The Feature Selection and Intrusion Detection Problems," *Proc. of the 9th Asian Computing Science Conference*, LNCS, 3321, Springer, 2004, 468-483.

[14] K. Kendall, "A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems," *Master's Thesis, MIT*, 1998.

[15] S. E. Webster, "The Development and Analysis of Intrusion Detection Algorithms," *S.M. Thesis, MIT*, 1998.

[16] J. Stolfo, F. Wei, W. Lee, A. Prodromidis, and P. K. Chan, "Cost-based Modeling and Evaluation for Data Mining with Application to Fraud and Intrusion Detection," *DARPA Information Survivability Conference & Exposition,* 1999, 130-144.

[17] J. Lassez, T. Karadeniz, and S. Mukkamala , "Zoomed Clusters*," Proceedings of The Thirteen International Conference on Neural Information Processing (ICONIP)*, LNCS, 4233, Springer, 2006, 824-830.

[18] S. Mukkamala, A. H. Sung "A Framework for Countering Denial of Service Attacks (Knowledge Discovery Approach)," *Proc. of IEEE International Conference Systems, Man, and Cybernetics*, 2004, 3273-3278.

[19] V. N. Vladimir, "The Nature of Statistical Learning Theory," *Springer*, 1995.

[20] T. Joachims, "Making Large-Scale SVM Learning Practical," LS8-Report, University of Dortmund, LS VIII-Report, 2000.

[21] S. Mukkamala, A. H. Sung, "Significant Feature Selection Using Computational Intelligent Techniques for Intrusion Detection," Advanced Methods for Knowledge Discovery from Complex Data, S. Bandyopadhyay, U. Maulik, L. Holder and D. Cook (Eds.) Springer, 2005, 285-306.

[22] W. Banzhaf, P. Nordin, R. E. Keller, and F. D. Francone, "Genetic Programming: An Introduction on the Automatic Evolution of Computer Programs and its Applications," Morgan Kaufmann Publishers, Inc, 1998.

[23] A. E. Bernal, T. Karadeniz, K. Hospevian, J-L. Lassez, "Similarity Based Classification," Advances in Intelligent Data Analysis V, LNCS, 2810, Springer, 2003, 187-197.