Falsification Prevention and Protection Technologies and Products

Data Mining for Security

By Kenji YAMANISHI,* Jun-ichi TAKEUCHI* and Yuko MARUYAMA*

ABSTRACT It becomes increasingly important to detect intrusions with unknown patterns in order to protect our business from cyber terrorism threats. This paper introduces data mining technologies designed for this purpose; SmartSifter (outlier detection engine), ChangeFinder (change-point detection engine), AccessTracer (anomalous behavior detection engine). All of them are able to learn statistical patterns of logs adaptively and to detect intrusions as statistical anomalies relative to the learned patterns. We briefly overview the principles of these engines and illustrate their applications to network intrusion detection, worm detection, and masquerader detection.

KEYWORDS **Intrusion detection, Worm detection, Masquerade detection, Anomaly detection, Data mining**

1. INTRODUCTION

In recent years, intrusion detection technologies are indispensable for network/computer security as the threat of cyber terrorism becomes a serious matter year by year. Most of conventional technologies such as IDS (Intrusion Detection Systems) take a signature-based approach to it, in which a number of human-made rules describing computer worm/virus of known patterns are constructed and an alarm is made when a record matching one of the rules appears. The signature-based approach suffers from the following two critical issues: 1) it is not able to detect worms/virus of unknown types, and 2) it requires a lot of computation time for signature matching.

Meanwhile, we may employ a policy-based approach in order to detect unknown computer worms/virus. In it a general security policy is constructed and an alarm is made when some record violates the policy. However, it is not able to detect new computer worm/virus in case they eventually satisfy the policy.

Data mining-based anomaly detection is a kind of technology for detecting computer worms/virus of unknown patterns more adaptively and effectively than signature-based and policy-based ones. This is to learn statistical regularities from past examples and to detect worms/virus as anomalies which are largely deviated from the learned regularities.

The authors have recently developed the following three data mining engines for the purpose of anomaly detection:

1) Outlier detection engine: SmartSifter[2,3,6]
2) Change-point detection engine: ChangeFinder[4]
3) Anomalous behavior detection engine: AccessTracer[7]

SmartSifter detects intrusions as statistical outliers. ChangeFinder detects the emergence of computer worms/virus by tracking a change point in a time series of log data. AccessTracer detects masqueraders' activities by tracking anomalous behaviors in a session stream such as a series of UNIX commands. They were all designed in order to realize "security intelligence[10]" which can be thought of as additional value for NEC's security solution.

The purpose of this paper is to give a brief overview of the three engines with their applications to real domains.

The rest of this paper is organized as follows: Section 2 introduces SmartSifter with its applications to intrusion detection. Section 3 introduces ChangeFinder with its applications to worm detection. Section 4 introduces AccessTracer with its applications to masquerade detection. Section 5 gives concluding remarks.

2. OUTLIER DETECTION ENGINE: SmartSifter

The basic principle of SmartSifter is to learn a statistical model of the data generation mechanism from past examples and then to calculate an anomaly score for each datum, with high score indicating high possibility of its being an outlier. We expect that we can detect intrusion data efficiently by investigating only data of high scores, thereby drastically reduce

*Internet Systems Research Laboratories

the total cost of investigation for detecting new worms/virus.

2.1 Outlier Detection Process

In this subsection, according to references [2] and [5], we show how SmartSifter works by describing the details of its "statistical model," "learning," and "scoring."

(1) Statistical Model

Log data may be represented by a multi-dimensional data where some attributes are discrete variables (e.g., service type, IP address, etc.) while others are continuous ones (time, duration, source bytes, etc.) SmartSifter employs a histogram density for a statistical model for discrete variables and a Gaussian mixture model for that for continuous ones. Here a Gaussian mixture model takes a form of a linear combination of a finite number of Gaussian distributions (See **Fig. 1**). SmartSifter constructs a statistical model of log data by combining the histogram density with the Gaussian mixture model under the assumption that data is independently identically distributed.

(2) Learning

SmartSifter learns the parameters of the statistical model from examples in an on-line manner, using our original on-line discounting learning algorithm[2]. This algorithm estimates the parameters of the statistical model by forgetting out-of-date statistics gradually every time a datum is input. It makes the learning adaptive to the change of the log patterns.

(3) Scoring

SmartSifter gives a score for each datum, which is calculated as the Shannon information of the data i.e., the information quantity of the data relative to the model learned so far. The higher the score of a datum is, with the higher possibility it is an outlier.

SmartSifter has the following novel features:

1) Adaptiveness: It detects outliers adaptively to the change of distributions of logs. As a result, it realizes adaptive intrusion detection.
2) Efficiency: It realizes on-line real-time outlier detection with low computational complexity.
3) High Accuracy: It is able to detect unknown types of intrusion, and thereby achieves high accuracy of intrusion detection, as illustrated below.

For example, consider the problem of detecting a DoS (Denial of Services) attack called Back (see, e.g., [11]), which is an attack for security holes in Apache (web server program). Since it tends to send a large amount of data to the server, the associated logs may be detected as statistical outliers. We have also empirically demonstrated that scanning activities and worms such as CodeRed and Slammer can be detected as statistical outliers. Note here that outliers that SmartSifter detects are not always true intrusions but rather may cause false alarms. It is an important issue how to tune SmartSifter in order to reduce the false alarm rates as much as possible in real domains.

We applied SmartSifter to a benchmark dataset called KDDCup99[11] in order to demonstrate its effectiveness in network intrusion detection problems[6]. We utilized three attributes (duration, src_bytes, dst_bytes), all of which are continuous ones. Here "src_byte" means the data amount sent to the server while "dst_byte" means the data amount received by the server. In our experiments, we used 500 thousands records that successfully logged in, 0.35% of which (= 1700 records) were intrusions.

Figure 2 shows how well SmartSifter was able to detect intrusions. The vertical axis shows the coverage of intrusions, i.e., the ratio of the number of detected intrusions over the total number of intrusions, while the horizontal axis shows the extraction rate, i.e., the ratio of the total number of data extracted for investigation over the total number of records. For example, x% in the horizontal axis means that the records of top x% highest scores are extracted. The real line shows the performance of SmartSifter, while the dot one shows that of Burge & Shawe-Taylor's method[1], which is also known to be an on-line outlier detection engine.

We observe that SmartSifter significantly outperforms Burge & Shawe-Taylor's method in terms of the coverage. Specifically, SmartSifter requires only 5% records in order to detect 80% of the intrusions while

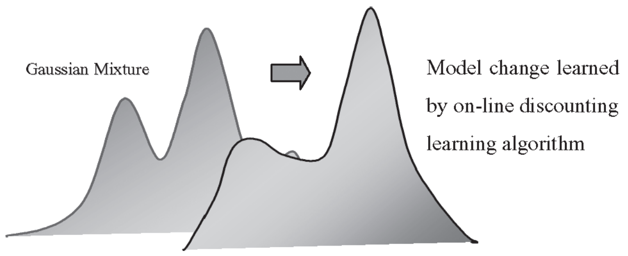

Gaussian Mixture
Model change learned by on-line discounting learning algorithm

Fig. 1 Principle of SmartSifter.

random search requires 80% data in order to detect the same number of intrusions. This implies that SmartSifter can drastically reduce the search cost of intrusions.

Figure 3 shows the user interface of SmartSifter. It displays the distribution of data with respect to specified attributes and where data of high scores are located.

2.2 Intrusion Knowledge Generation

Once SmartSifter detects outliers, we are interested in generating a rule which explains their patterns. We may call such a rule an outlier-filtering rule. For example, it is written in a form of "If-then-else" type rule, as follows:

If "src_byte<9.688 & flag=SF" then normal
else if "service=http" then outlier
else normal

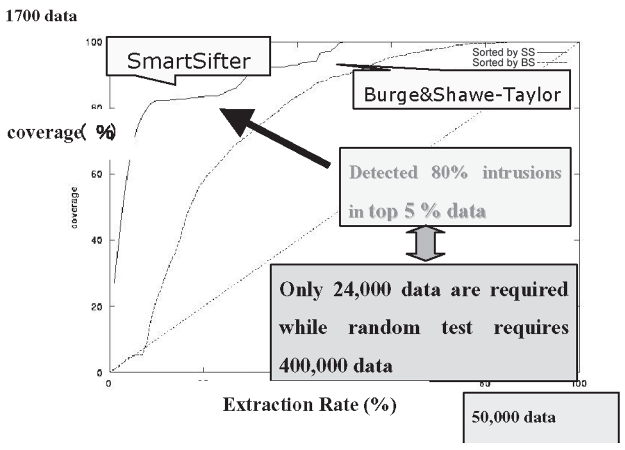
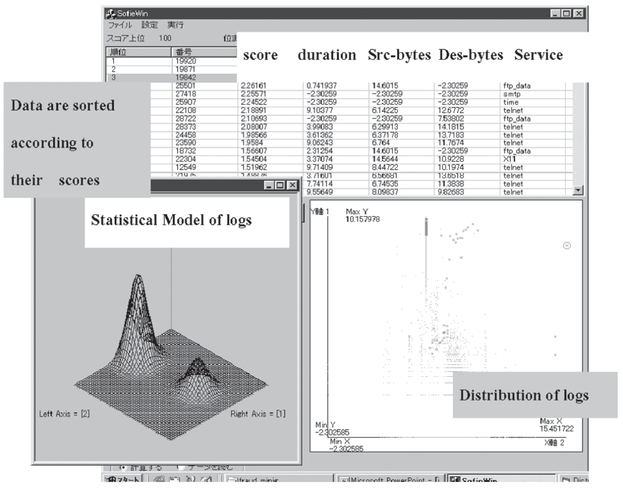


Fig. 2 Intrusion detection using SmartSifter.



順位	番号	score	duration	Src-bytes	Des-bytes	Service
1	19920					
2	19871					
3	19842					
	25501	2.26161	0.741937	14.6015	-2.30259	ftp_data
	27418	2.25571	-2.30259	-2.30259	-2.30259	smtp
	25907	2.24522	-2.30259	-2.30259	-2.30259	time
	22108	2.18891	9.10377	6.14225	12.6772	telnet
	28722	2.10693	-2.30259	-2.30259	7.53802	ftp_data
	28373	2.08007	3.99083	6.29913	14.1815	telnet
	24458	1.98566	3.61362	6.37178	13.7183	telnet
	23590	1.9584	9.06243	6.764	11.7674	telnet
	18732	1.56607	2.31254	14.6015	-2.30259	ftp_data
	22304	1.54504	3.37074	14.5644	10.9228	X11
	12549	1.51962	9.71409	8.44722	10.1974	telnet
	[illegible]	[illegible]	3.71601	6.56681	13.6518	telnet
			7.74114	6.74535	11.3838	telnet
			9.55649	8.09837	9.82683	telnet

Fig. 3 User interface of SmartSifter.

Generating such outlier-filtering rules is important in the following two regards:

1) It explicitly explains why the groups of outliers that SmartSifter detects are exceptional.
2) It can be used for preprocessing of SmartSifter for new data, in order to achieve higher accuracy for outlier detection.

We have developed a method for automatically generating outlier-filtering rules on the basis of supervised learning technique. **Figure 4** shows the flow of this method. The basic principle is shown below.

Once data are given scores by SmartSifter, then we give positive labels to data of high scores and negative labels to the data sampled randomly from the remaining dataset. Here the numbers of positive labeled data and negative ones are pre-determined. Then we learn a classification rule which discriminates from the positive labeled data from the negative one, where we employ a supervised learning algorithm using the information criterion called ESC (Extended Stochastic Complexity) as a rule selection criterion[3,5]. Once a rule is generated, it is used for filtering outliers for a new data set. This process is repeated every time a new data set is added into the system.

Note that the resulting rule may possibly capture a pattern of a specific type of intrusions and produce new security knowledge. For example, the "If-then-else" type rule above shown, which was generated for KDDCup99 by our system, characterizes the features of the attack called Back. This implies that our system was able to automatically produce the knowledge about Back.

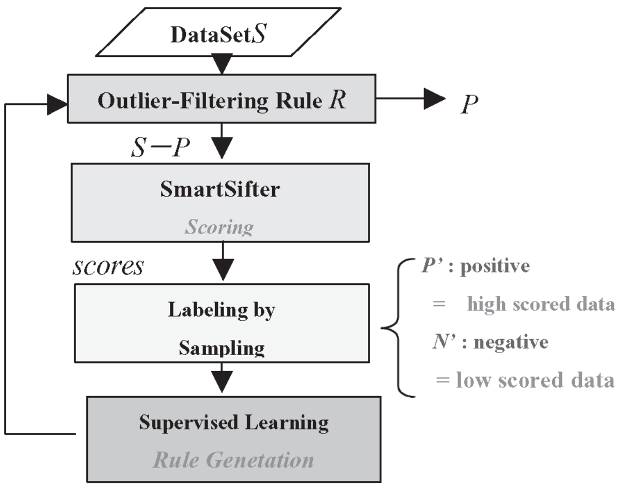


Fig. 4 Outlier-filtering rule generation process.

We may use outlier-filtering rules with SmartSifter in the manner as shown in Fig. 4 to enhance the outlier-detection power of SmartSifter. It is demonstrated in the reference [3] that for KDDCup99 dataset, combining SmartSifter with outlier-filtering rules achieved more than 50% higher coverage than SmartSifter itself.

3. CHANGE-POINT DETECTION ENGINE: ChangeFinder

A computer worm or virus may often appear burstly rather than point-wise. In fact, when a new type of worm emerges, the number of access logs tends to suddenly increase. The technology of on-line change-point detection is expected to be effective in detecting computer worms/virus having such a property as early as possible. Here the goal of on-line change-point detection is to detect the earliest time point when the nature of time series has significantly changed. ChangeFinder is designed to conduct this function efficiently. Note that SmartSifter cannot be applied to change-point detection since it cannot deal with time series models but rather independent models only.

ChangeFinder has basically the same principle as SmartSifter in that both of them learn a statistical model adaptively from data and give a score to each datum on the basis of the learned model. The difference between them is that ChangeFinder further employs the technique of 2 stage learning through smoothing. According to the reference [4], the details of this technique are summarized below.

(1) First Stage of Learning and Scoring

ChangeFinder employs as a statistical model a time series model called an auto-regression (AR) model and learns it from data using the on-line discounting learning algorithm [4] every time a datum is input. Then it calculates an anomaly score for each time point as the Shannon information of the datum relative to the model learned so far.

(2) Smoothing

ChangeFinder prepares a window of a fixed size and constructs a time series of moving averages of the anomaly scores for the data points by sliding the window. Here the moving average is taken over all the data points included in the window.

(3) Second Stage of Learning and Scoring

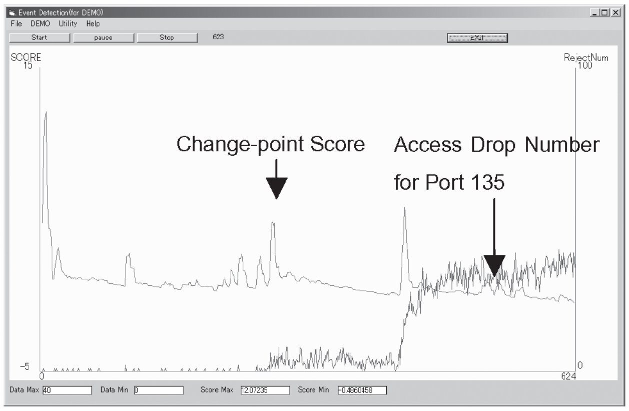
ChangeFinder employs another AR model and learns it from the time series of the moving-averaged scores. Then it calculates a change-point score for each time point as the Shannon information of the moving-averaged score at the point relative to the model learned so far.

For example, ChangeFinder is effective in detecting the DoS attack called SYN Flood, since it tends to make bursty TCP-incomplete connections using masqueraded IDs and cause traffic concentration.

Figure 5 shows an example of applications of ChangeFinder to detecting a computer worm called MS Blast. The figure shows a time series of access frequencies at port 135 and a change-point score curve. We observe that there are two distinct peaks in the change-point curve, all of which correspond to the earliest stages of the real emergence of MS Blast. It was actually reported that MS Blast emerged in two steps. Further note that ChangeFinder conducts the change-point detection in real-time, with computation time of order O(nk2) where n is the sample size and k is the dimension of a datum. This implies that ChangeFinder is quite effective in detecting the emergence of computer worms as early as possible.

4. ANOMALOUS BEHAVIOR DETECTION ENGINE: AccessTracer

In previous sections we were concerned with the issue of how anomalous an individual data point is. In other words, SmartSifter and ChangeFinder were designed to detect local anomalies in a data set. However, there are some situations where it is required to detect global dynamics of anomalies, such as anomalous behavior patterns, in a set of time series. AccessTracer is designed to detect such a type of anomalies. For example, it can be applied to detecting masqueraders' behavior patterns from UNIX command



Data Max 40 Data Min 0 Score Max 12.07235 Score Min -0.4860458

Fig. 5 Worm detection using ChangeFinder.

histories and detecting intrusion patterns such as Trojan horses from a sequence of system calls.

For a given time series, we divide it into a number of subsequences, each of which is called a session. AccessTracer takes as input a session stream and learns a statistical model of behavior patterns from it. Then it detects an anomalous behavior and identifies what a new behavior pattern is generated. Below we give a brief sketch of the principle of AccessTracer according to the reference [7].

(1) Statistical Model

AccessTracer employs as a statistical model of behavior patterns a mixture of hidden Markov models (HMMs), which is a linear combination of HMMs where each HMM represents a behavior pattern and the number of mixture components represents the number of distinct behavior patterns. For example, a mixture of HMMs is used for the representation of UNIX users' command history.

(2) Learning

AccessTracer learns the mixture of HMMs using the on-line discounting learning algorithm[7] every time a session is input. It learns not only the parameter values of the model but also the optimal number of mixture components. Hence it can track the change of number of behavior patterns over time. This is conducted on the basis of the theory of dynamic model selection developed in the reference [8] (See **Fig. 6**).

(3) Scoring

AccessTracer calculates an anomaly score for each session as the universal hypothesis test statistics, which can be thought of as an extension of Shannon-information for data-scoring to session-scoring. The higher the score is, the higher possibility of being an anomalous session it has. A session of locally highest score corresponds to the onset of an anomalous session stream.

(4) Behavior Pattern Identification

In the process of dynamically tracking the change of the number of mixture components, its increase implies that a new behavior pattern has emerged while its decrease implies that an existing behavior pattern has disappeared. AccessTracer does not only track the changes but also identifies what a mixture component has newly emerged or disappeared. For example, when learning from a UNIX command session stream, the number of command patterns may be increased after a masquerader comes into action. Hence detecting such a change leads to the masquerade detection and identifying a new mixture component leads to the understanding of a masquerader's behavior pattern.

Figure 7 shows the user interface of AccessTracer and illustrates how it analyzes users' UNIX command stream. We divided an original sequence of UNIX commands into a number of sessions, each of which consists of 10 commands, to form a session stream. The horizontal axis shows the session number, and the two graphs are shown in the upper-side of the display. One is a graph of anomaly score for sessions and the other is a graph of the optimal number of mixture components in the mixture of HMMs. In the lower-side of the display, there are shown clusters of command patterns, each of which corresponds to a component of the mixture model. For each command pattern, a list of typical commands with high

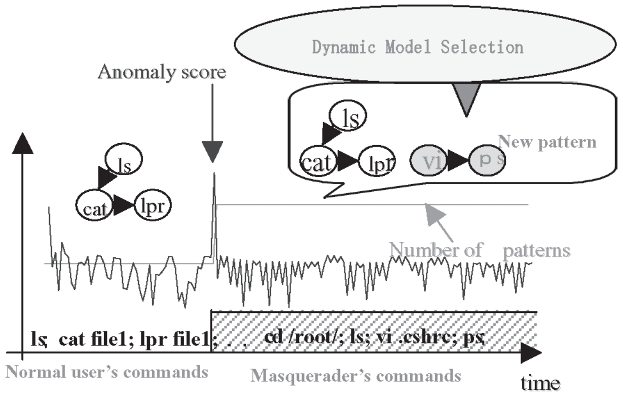


Fig. 6 Masquerade detection using AccessTracer.

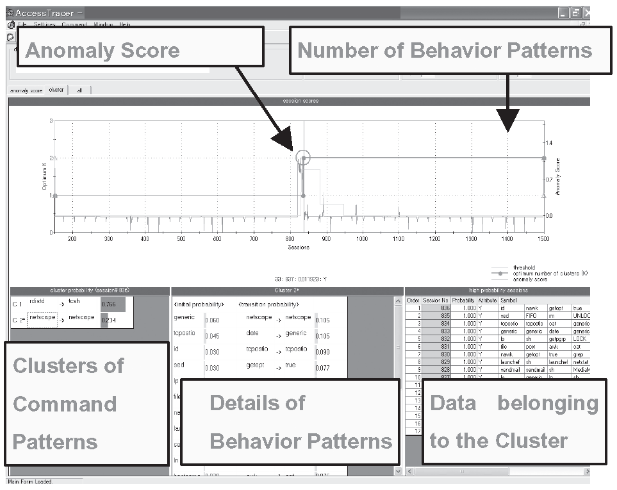

Anomaly Score
Number of Behavior Patterns
Clusters of Command Patterns
Details of Behavior Patterns
Data belonging to the Cluster

Fig. 7 User interface of AcccessTracer.

frequencies, a list of typical transitions with high probabilities, and data belonging to the cluster are displayed.

We evaluated AccessTracer using a benchmark data set collected by Schonlau et al.[12] in order to demonstrate its effectiveness in masquerade detection using UNIX command streams. We have reported in the reference [7] that AccessTracer was able to reduce the false alarm rate more than 50% in comparison with the Naive Bayes method, which was proven to perform best to date[9]. Furthermore it was proven in the reference [8] that AccessTracer was able to successfully identify a specific command pattern of a masquerader in a comprehensive form. This implies that AccessTracer is effective in masquerader's pattern identification as well as masquerader detection.

5. CONCLUDING REMARKS

This paper overviewed the three data-mining based anomaly detection engines: Outlier detection engine SmartSifter, change-point detection engine ChangeFinder, and anomalous behavior detection engine AccessTracer. All of them were designed for the purpose of effectively and efficiently detecting security incidents of unknown types such as unknown worms, viruses, masquerades, etc. We expect that they would be more effective if they were used in combination with existing security products such as firewalls and IDSs.

The features of the three engines are their adaptiveness and real-time performance. These engines would also be applied to a wide range of areas other than security, including activity monitoring, fraud detection in finance, medical sciences, etc.

REFERENCES

[1] P. Burge and J. Shawe-Taylor, "Detecting cellular fraud using adaptive prototypes," in Proceedings of AI Approaches to Fraud Detection and Risk Management, pp.9-13, 1997.

[2] K. Yamanishi, J. Takeuchi, et al., "On-line Unsupervised Oultlier Detection Using Finite Mixtures with Discounting Learning Algorithms," *Data Mining and Knowledge Discovery Journal*, **8**, 3, pp:275-300, Kluwer Academic Publishers, May 2004.

[3] K. Yamanishi and J. Takeuchi, "Discovering outlier filtering rules from unlabeled data," in Proceedings of the Seventh ACM SIGKDD International Conference on Data Mining and Knowledge Discovery, ACM Press, pp.389-394, 2001.

[4] K. Yamanishi and J. Takeuchi, "A Unifying Approach to Detecting Outliers and Change-Points from Non stationary Data," in Proceedings of the Eighth ACM SIGKDD International Conference on Data Mining and Knowledge Discovery, ACM Press, pp.676-681, 2002.

[5] K. Yamanishi, "Data and Text Mining," Iwanami Statistical Science Frontiers, Series 10 Cp.179-242, 2003 (in Japanese).

[6] J. Takeuchi and K. Yamanishi, "Experimental Evaluation of Outlier Detection Engine SmartSifter," in Proceedings of the 23rd Symposium on Information Theory and Its Applications, pp.419-422, 2000 (in Japanese).

[7] Y. Matsunaga and K. Yamanishi, "An Information-theoretic Approach to Detecting Anomalous Behaviors," in Proceedings of the 2nd Forum on Information Technologies (Information Technology Letter), pp.123-124, 2003 (in Japanese).

[8] Y. Maruyama and K. Yamanishi, "Dynamic Model Selection with Its Applications to Computer Security," in Proceedings of IEEE Information Theory Workshop, 2004 (http://ee-wcl.tamu.edu/itw2004/program.html).

[9] R. A. Maxion and T. N. Townsend, "Masquerade detection using truncated command lines," in Proceedings of Int. Conf. on Dependable Systems and Networks, pp.219-228, 2002.

[10] http://www.labs.nec.co.jp/DTmining/

[11] http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[12] http://www.research.att.com/~schonlau

Received December 1, 2004

* * * * * * * * * * * * * * *

Kenji YAMANISHI received M.E. degree from the University of Tokyo in 1987. He joined NEC Corporation in 1987, and is now a Research Fellow of Internet Systems Research Laboratories. He received Dr.Eng. degree from University of Tokyo in 1992. He worked for NEC Research Institute in USA as a visiting scientist from 1992 to 1995. He is engaged in research and development of data mining technologies. He authored a chapter "Data and Text Mining" of the book "Statistics of Language and Psychology" published by Iwanami publishers.

Dr. Yamanishi is a member of IEEE, ACM, IEICE, SITA, and JSAI.

Jun-ichi TAKEUCHI received the B.Sc. degree in physics and the Dr.Eng. degree in mathematical engineering from the University of Tokyo in 1989 and 1996, respectively. He joined NEC Corporation in 1989, where he is currently a Principal Researcher in Internet Systems Research Laboratories. From 1996 to 1997 he was a Visiting Research Scholar at Department of Statistics, Yale University. His research interest includes machine learning and data mining.

Dr. Takeuchi is a member of IEEE, IEICE, SITA, and JSIAM.

Yuko MARUYAMA (maiden name: Yuko Matsunaga) received the B.E. degree and the M.E. degree in mathematical engineering from the University of Tokyo in 2000 and 2002, respectively. She joined NEC Corporation in 2002, where she is currently a researcher in Internet Systems Research Laboratories. Her research interest includes Shannon theory, coding theory, machine learning, and data mining.

* * * * * * * * * * * * * * *