IEICE TRANS. COMMUN., VOL.E90–B, NO.4 APRIL 2007

PAPER

# Detecting Unknown Worms Using Randomness Check*

**Hyundo PARK**[†], *Nonmember*, **Heejo LEE**[†a)], *Member*, *and* **Hyogon KIM**[†], *Nonmember*

**SUMMARY** From the introduction of CodeRed and Slammer worms, it has been learned that the early detection of worm epidemics is important in order to reduce the damage resulting from outbreaks. A prominent characteristic of Internet worms is the random selection of subsequent targets. In this paper, we propose a new worm detection mechanism by checking the random distribution of destination addresses in network traffic. The proposed mechanism constructs a matrix from network traffic and checks the rank of the matrix in order to detect the spreading of Internet worms. From the fact that a random binary matrix holds a high rank value, ADUR (Anomaly Detection Using Randomness check) is proposed for detecting unknown worms based on the rank of the matrix. From experiments on various environments, it is demonstrated that the ADUR mechanism effectively detects the spread of new worms in the early stages, even when there is only a single host infected in a monitoring network. Also, we show that ADUR is highly sensitive so that the worm epidemic can be detectable quickly, e.g., three times earlier than the infection of 90% vulnerable hosts.

***key words:*** *Internet worm, early detection, randomness, traffic matrix, rank*

## 1. Introduction

An Internet worm is a malicious code that propagates by replicating itself onto other computers. Such a self-replicating malicious code scans vulnerable hosts on a network, and replicates itself to vulnerable hosts without user intervention. The first worm was the Morris worm, unleashed in 1988, since then, the number of incidents of Internet worms has grown, and continues to grow drastically. CodeRed and Nimda worms infected hundreds of thousands of vulnerable computers in 2001. A range of targets, from public institutes to personal users, suffered from the damage caused by CodeRed and Nimda. The amount of damage totaled millions of dollars [1]–[4].

With regard to the history of Internet worms, the Slammer worm [5] is known as the fastest spreading worm. It takes a mere 10 minutes, to infect 90 percent of vulnerable hosts on the Internet. The number of infected hosts doubles every 8.5 seconds. This speed is considerably faster than CodeRed, which doubles every 37 minutes. The first step toward countering a worm epidemic is "early detection." However, signature-based detection algorithms are not effective for detecting new worms or polymorphic worms, since they can always change codes. Anomaly-based approaches can be used for detecting such worms, at the expense of a higher degree of complexity and false alarms.

In this paper, a new method for detecting the spread of Internet worms, which is named ADUR (Anomaly Detection Using Randomness check), is proposed. The ADUR mechanism can detect a new worm by measuring the randomness of destination addresses in network traffic, where the randomness is formed when a worm propagates randomly over the Internet. In checking the randomness of address distribution, ADUR distinguishes between the state of normal conditions and the state of worm epidemics. The two main features of ADUR are the use of "matrix" representations and exclusive-or (XOR) operations. Matrix representations provide many benefits to implement particular operations, regardless of network size and traffic volume. The XOR operator diminishes the effect of normal traffic and magnifies the effect of worm traffic, which results in reduced false alarms. In measuring the dynamics of the rank of the traffic matrix, it is ensured that ADUR can detect a future worm epidemic in the early stages of propagation.

The main contributions of this study can be divided into three key areas. First, a novel approach is proposed to detect unknown worms based on the randomness of worm traffic. Second, we show that an anomaly detection algorithm based on a matrix construction and its simple XOR operation greatly increases the flexibility and the accuracy of detection. Finally, the algorithm provides additional benefits such as indicating propagating directions and disclosing infected subnet locations.

The subsequent sections of this paper are organized as follows. In Sect. 2, previous research on scan detection and related work is discussed. In Sect. 3, scanning methods of Internet worms, used for selecting a target host, are explored. Section 4 describes the method of checking the randomness of a binary matrix. In Sect. 5, the ADUR mechanism is proposed and the reason why it uses the XOR operator is presented. The evaluation of the proposed ADUR mechanism is presented in Sect. 6. Finally, the paper concludes in Sect. 7.

## 2. Related Work

In general, algorithms to detect worms are divided into two classes, signature based and anomaly based detection algorithms. The signature based worm detection algorithm uses

Manuscript received June 20, 2006.
Manuscript revised November 3, 2006.

[†]The authors are with the Department of Computer Science and Engineering, Korea University, Seoul 136-713, South Korea.

*This work was supported in part by the ITRC program of the Korea Ministry of Information & Communications, the BK21 program of the Korea Ministry of Education, and the Basic Research Program of the Korea Science & Engineering Foundation.

a) E-mail: heejo@korea.ac.jp

DOI: 10.1093/ietcom/e90–b.4.894

a pattern matching method with the information of previous known worms. The anomaly based worm detection algorithm uses various characteristics of worms.

The signature based worm detection algorithm is widely used on most systems for detecting worms. This system gathers the information of specific worms after the worm propagates. For example, the CodeRed worm reveals a pattern composed of specific values in network traffic. This value can be the signature of the CodeRed worm [1]. If this signature is detected on network traffic in the future, it is a worm propagation state. These algorithms have the advantage of having a low probability of false-positive alarm. However, they cannot handle an unknown worm properly.

Unlike signature based worm detection algorithms, anomaly based worm detection algorithms detect new worms using the activity characteristics of an Internet worm. Thus, they can detect an unknown worm when the worm has the same characteristics as Internet worms. The worm detection algorithms can be divided into three categories. The first method examines the sudden increase of new connection attempts. The second method examines the sudden increase of connection failures. The third method examines the sudden increase of abnormal connection attempts.

The first method detects a new worm by counting the number of new connection attempts. Leckie et al. proposed a probabilistic model to detect unusual access patterns by analyzing the connection attempts [10]. In normal states, this model obtains the probability distribution of source addresses, destination addresses and port numbers. Then, the probability distribution is used as a base for determining a worm epidemic.

The second method detects new worms by counting the number of connection failures. Symptoms of connection failures include TCP_RST packets, ICMP destination unreachable messages and TCP timeouts. The algorithm is proposed by Vincent Berk et al. [8], [9] detect worms using ICMP destination unreachable messages. The majority of the Internet worms generate the target host IP address using a random generator so that the number of failed connections between source and target hosts will increase in the worm propagation state. One drawback of the algorithm is the inability of detecting the Internet worm using IP spoofing since the failure messages will not return to the infected host. An algorithm, named threshold random work (TRW), is proposed by Jung et al. [11]. TRW regards the SYN packet in the initial connection of the TCP protocol as a means of worm scanning. Abnormal traffic patterns are found by a sequential hypothesis testing, using information from the SYN packet. This method will not work properly for detecting UDP worms. As well, if the worm propagates rapidly over the Internet, connection failures increases suddenly and a monitoring system applying TRW consumes excessive memory due to the per-host recording of TCP connections. And the TRW method is ineffective to detect worms using distributed scans, such as Curious Yellow [14].

The third method detects new worms by counting the number of abnormal connection attempts. One approach relies on the correlation of the DNS queries with outgoing connections from an enterprise network [12]. The other approach is derived from the correlation of ARP (Address Resolution Protocol) activities from individual network attached devices [13]. However, in a particular case, normal connections without DNS queries bring network traffic such as P2P applications. In this case, DNS based detection algorithm result in high false positives.

Previous research refers to algorithms requiring complex computation or depending on the specific protocols. Otherwise, most works require to select an appropriate threshold to a certain location of a network. Here we define four main requirements to obtain a viable solution to detect unknown worms in various network environments.

1. In order to operate on high-speed networks, an algorithm should be light-weight.
2. In order to operate at any place of a network, an algorithm should be location-free for transplantation.
3. In order to detect every possible worm, an algorithm should not rely on a specific protocol, i.e., TCP or UDP.
4. In order to operate on any network environment, an algorithm should not require any threshold value.

In this paper, a new anomaly worm detection mechanism is proposed using a randomness check, named ADUR. The ADUR mechanism fulfills the above conditions.

## 3. Scanning Methods of Active Worms

In this section, the scanning methods of Internet worms are described and their relationship is presented for the randomness of worm traffics.

In order to demonstrate the rapid spreading of an Internet worm, we can use an analytical model, such as the model of analytical active worm propagation (AAWP) [19]. In the AAWP model, the number of infected hosts at time $i$ is shown in Eq. (1), where $N$ is the number of vulnerable hosts, $T$ is the address space used by the worm spreading, $s$ is the scanning rate and $n_i$ is the number of infected hosts at time $i$.

$$n_{i+1} = n_i + [N - n_i]\left(1 - \left(1 - \frac{1}{T}\right)^{sn_i}\right) \quad (1)$$

In Eq. (1), it is assumed that the starting time is 0, i.e., $i = 0$. The value of $n_0$ is equal to the initial hitlist size. Figure 1 shows the distribution of infected hosts as a function of time tick. Even with a different hitlist size, the number of infected hosts increases drastically when the value of $n$ passes a certain point, e.g., 10,000 in Fig. 1. This is caused by the fact that, as the propagation proceeds, the number of scanning packets also increases along with the increased number of infected hosts. Thus, the number of infection is accelerated by finding remaining susceptible hosts more rapidly.

There are various scanning methods for worm propagation [6], [7], [15]. We can classify scanning methods onto

**Table 1** Various scanning methods of Internet worms.

| Scanning method | Description | Example |
|---|---|---|
| Random scanning | Target hosts are chosen randomly. | CodeRed II [1], Slammer [5] |
| Hitlist scanning | The list of vulnerable hosts is used. Outgoing connections increase suddenly. | Warhol [16] |
| Topological scanning | The information of target is gathered on the infected host. Outgoing connections increase suddenly. | Morris [17] |
| Local scanning | Most targets are selected within the local network. Failed messages for connection requests increase. | CodeRed [18], Nimda [2] |
| Permutation scanning | This is to avoid the overlapping of scanning ranges. | Warhol [16] |

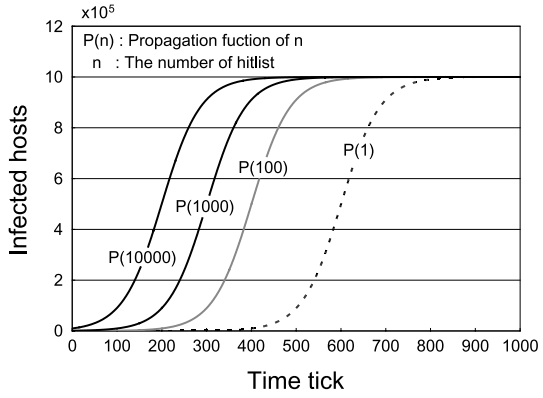



**Fig. 1** The number of infected hosts as a function of time tick when $n_0$ = 1, 100, 1000 and 10000, respectively.

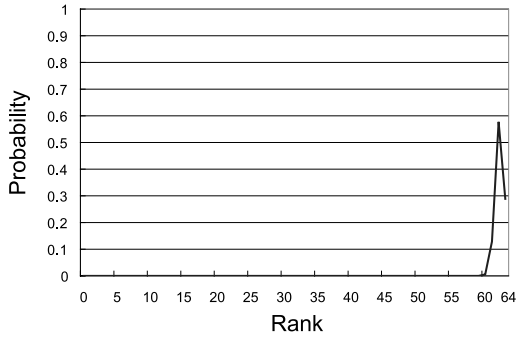



**Fig. 2** Probability distribution of the rank of a 64×64 random matrix.

five categories as shown in Table 1: random scanning, hitlist scanning, topological scanning, local scanning and permutation scanning. One easy way of spreading a worm code is by generating random IP addresses for next targets. However, this random scanning has a few limitations such as slow spreading during the early stages of infection and multiple probes of single hosts. Other scanning methods can overcome these general inefficiencies of random scanning.

A hitlist scanning worm has a list of IP addresses of vulnerable hosts. The vulnerable hosts are not likely to form a particular distribution pattern but can be distributed randomly. Thus, the scanning traffic of a hitlist worm may have the randomness property in destination addresses. A Topological scanning worm gathers the information of target hosts at an infected host. The sequence of target addresses, gathered on the infected host, also has the randomness in the destination address distribution. A local scanning worm selects random target addresses mainly within the local network. Permutation scanning is to divide the scanning ranges among infected worms in order to avoid the overlapping of scanning ranges. Eventually, the traffic generated by the permutation scanning worms contains the randomness property since the vulnerable hosts may not have a tendency to form a specific distribution but can be distributed randomly.

Thus, conventional worm propagation strategies produce the "randomness" in the address distribution of target hosts. This implies that the spreading of worms can be monitored by measuring the randomness of destination addresses in network traffic. In this study, an attempt is made to measure the degree of randomness in traffic, in order to catch the fast spreading of high-speed worms.

## 4. Matrix Rank as a Randomness Metric

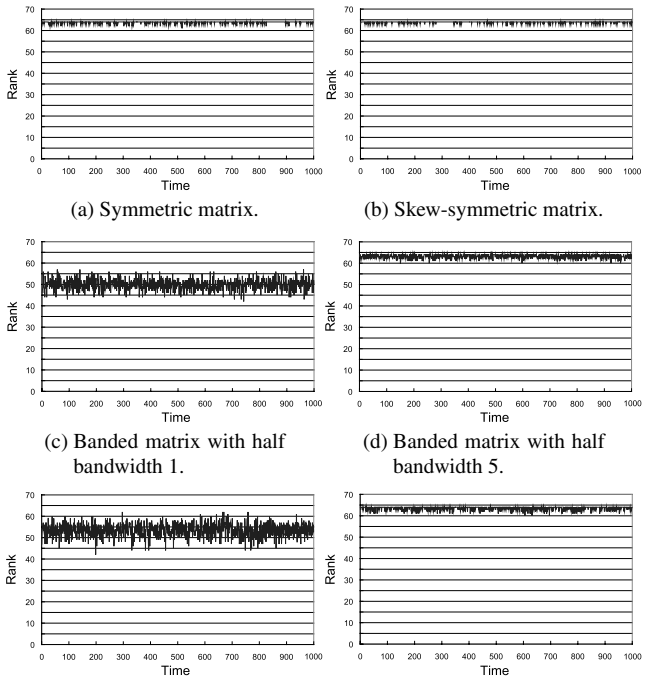
Many approaches have been suggested for testing the randomness, and a cost-effective approach is checking the linear-dependency among fixed-length substrings of its original sequence. In order to check the linear-dependence among rows or columns of a matrix, the rank of a matrix can be used [21]. Diehard [22] battery of tests can be an example, which is widely used for testing the quality of a random number generator.

A straightforward method to compute the rank of a matrix is by counting the number of non-zero rows after applying the Gaussian elimination method to the matrix. In other words, the rank of the matrix is equal to the number of leading 1's on the matrix [20]. In the case of a random $m \times n$ binary matrix, the rank of the matrix has the following probability:

$$2^{r<n+m-r>-nm} \prod_{i=0}^{r-1} \frac{\left(1-2^{i-n}\right)\left(1-2^{i-m}\right)}{\left(1-2^{i-r}\right)} \tag{2}$$

where rank $r = 1, 2, \ldots, \min(m, n)$ [21]. From Eq. (2), the distribution of probability for a given random matrix can be obtained. In the case of 64×64 random binary matrices, the distribution of probabilities for the rank of a random matrix is shown in Fig. 2.

For further discussion, it is assumed that the traffic matrix is $64 \times 64$ without loss of generality. Larger networks can use a larger matrix. If one 64×64 random matrix is provided, the probability that the rank exceeds 60 is over




(a) Symmetric matrix. (b) Skew-symmetric matrix.

(c) Banded matrix with half bandwidth 1. (d) Banded matrix with half bandwidth 5.

(e) Symmetric banded matrix with half bandwidth 1. (f) Symmetric banded matrix with half bandwidth 5.

**Fig. 3** Rank distributions for various types of matrix.

99.995% from Eq. (2). This implies that, if the 64×64 binary matrix is a random matrix, the rank has a high probability of being greater than 60. In this way, the rank of a matrix can be used to determine the randomness of element distribution.

Fig. 3 shows the rank distributions of various types of random matrices. The values are inserted into the elements on matrices randomly and the characteristics of each matrix are guaranteed at the same time. In Fig. 3, a symmetric matrix, a skew-symmetric matrix, a banded matrix with a half bandwidth 1 and 5 and a symmetric banded matrix with a half bandwidth 1 and 5 have a high rank value. The orthogonal elements of each matrix contain the randomness property. As well, the rank of a banded matrix becomes greater than 60 when the half bandwidth is greater than 5. This result demonstrates that not only a fully random matrix for all entries but also a matrix with random elements in the designated area contains the property of randomness. For example, if a matrix is a symmetric random binary matrix, the rank becomes greater than 60.

## 5. Anomaly Detection Using Randomness Check

We propose an anomaly-based worm detection algorithm, which is called ADUR (Anomaly Detection Using Randomness check). This section describes the ADUR mechanism, which includes matrix representation of network traffic and the XOR operation of two consecutive matrices. It is demonstrated how to express network traffic on a matrix and how to use the XOR operation in order to diminish the effect of normal traffic and magnify the effect of worm traffic.

**Table 2** Four states depending on the rank of traffic matrix.

| State | Description |
|---|---|
| Calm | Both $R(M_I)$ and $R(M_O)$ remains in a small range. |
| Flowing | $R(M_I)$ suddenly increases but $R(M_O)$ remains steady in a small range. |
| Ebbing | $R(M_I)$ remains steady in a small range, however $R(M_O)$ suddenly increases. |
| Flooding | Both $R(M_I)$ and $R(M_O)$ suddenly increase. |

### 5.1 System Design

The ADUR mechanism is to detect the spreading of Internet worms through checking the randomness of traffic. Traffic data can be classified into two categories based on their direction: incoming and outgoing. ADUR checks two directions respectively, in order to obtain more accurate attack information such as either entering or departing the network. Checking the randomness of traffic can be accomplished by measuring the rank of the matrix representing network traffic for a given period of time.

Let $M_I$ denote the matrix marked with incoming traffic. Let $M_O$ represent the matrix marked with outgoing traffic. $R(M_I)$ and $R(M_O)$ represent the rank of matrix $M_I$ and $M_O$, respectively. Then, the value of $R(M_I)$ and $R(M_O)$ can be used to determine whether the worm is active. There are four states depending on the ranks $R(M_I)$ and $R(M_O)$ of traffic as shown in Table 2: calm, flowing, ebbing and flooding. In the calm state, the ranks of incoming and outgoing traffic matrices remain in a small range. Calm state implies that there is no suspicious activity so that the network is in normal state. In the flowing state, the rank of incoming traffic matrix increases, while the rank of outgoing traffic matrix remains in a small range. The flowing state implies that the network is under attack from external networks, which are infected by an Internet worm. In the ebbing state, only the rank of outgoing traffic matrix increases, which implies that the network is already infected by an Internet worm. The flooding state is the combination of both flowing and ebbing states.

### 5.2 Traffic Matrix Design

Matrix representation is described as follows, which can be used for both incoming and outgoing traffic. Each IP address is divided into four octets such as

$$\mathrm{IP}_1.\mathrm{IP}_2.\mathrm{IP}_3.\mathrm{IP}_4 \tag{3}$$

where the length of each octet is one byte. Matrix expression consists of two operations. The first one is placement. The second is storing information. When a packet is captured in a monitoring network, the packet is mapped into a specific location of a matrix, which is determined by the destination address of the packet. A placement function is described in Eq. (4).

$$\begin{aligned} i &= (\mathrm{IP}_4/16) \times 4 \\ j &= (\mathrm{IP}_4 \pmod{16}) \times 4 \end{aligned} \tag{4}$$

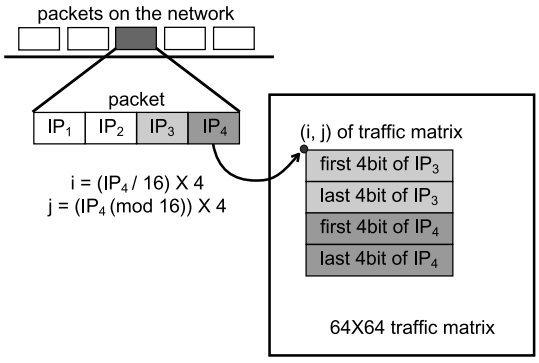



**Fig. 4** Constructing matrix $M$ by mapping a packet to submatrix $m$.

Next, a part of information in the packet is stored onto the matrix. Since Internet worms are likely to change the last two octets more frequently than the first two octets, the last two octets can be stored to capture the dynamics of worm traffic. In case of outgoing traffic, the destination address of a packet can be used for storing such information onto the matrix. In case of incoming traffic, the source address is used instead of the destination address. Figure 4 illustrates the information stored on a 4×4 submatrix, where the traffic matrix is 64×64. Furthermore, the 4×4 submatrix consists of four 1×4 submatrices, i.e. $m_1$, $m_2$, $m_3$, $m_4$. The contents of 1×4 submatrices are described in Eq. (5).

$$\begin{aligned} m_1 &= \text{first 4bit of IP}_3 \\ m_2 &= \text{last 4bit of IP}_3 \\ m_3 &= \text{first 4bit of IP}_4 \\ m_4 &= \text{last 4bit of IP}_4 \end{aligned} \tag{5}$$

There are other ways to generate traffic matrix by using a different placement function and storing different parts of information. However, they should have two conditions to form an effective traffic matrix. First, the traffic matrix must involve the characteristic of network traffic generated by a random generator. Namely, both source and destination IP address must be incorporated on a method of matrix generation. Second, the traffic matrix, e.g., $m \times n$ matrix, must be a square matrix, i.e., $m = n$, and large enough, i.e., $m \geq 10$ [21].

In Eq. (5), both $m_3$ and $m_4$ are meaningful in the traffic matrix. There are three reasons to construct a traffic matrix with both $m_3$ and $m_4$. The first reason is that we can not construct a square traffic matrix efficiently without both $m_3$ and $m_4$. In a /24 network, there is 256 hosts, so that 256 submatrices are required in a 64×64 traffic matrix. If the size of a submatrix is 2×4, 256 submatrices can be located in a 32×64 matrix, which is not a square matrix. But if the size of a submatrix is 4×4, we can construct a 64×64 square matrix. The second reason is that we can not detect the Internet worm infected at a host or a /24 network. It is not always true that there is no way to construct a square traffic matrix without both $m_3$ and $m_4$, if we have enough hosts ($> 256$). For example, we construct a square matrix with the traffic of two /24 networks. A upper half of a traffic matrix can be used for one network and the rest can be used for the other network. In this case, if a /24 network or a host is infected by a worm, the ADUR can not detect the worm since only a half of the matrix has a randomness. It implies that the rank value of the matrix will not become over 60. The third reason is that the attack location cannot be provided when we made a matrix without $m_3$ and $m_4$. For example, in case of constructing a square matrix with two networks, the ADUR can not localize one network as the attack location. When the rank of a matrix is over 60, it means that both networks were infected by a worm and ADUR cannot decide one network as an attack location. It implies the loss of one benefit of ADUR, which will be discussed at Sect. 6.5, thus we need to use both $m_3$ and $m_4$ for constructing a traffic matrix.

### 5.3 XOR Operator on Matrix Sequence

Not only suspicious traffic but also legitimate traffic must be considered properly. Simple XOR operation is greatly effective for this purpose. Let $M_t$ denote the matrix at time $t$. Equation (6) presents the XOR operation on a sequence of matrices, which dramatically reduces the effect of legitimate traffic on the rank of traffic matrix.

$$R(M'_t) = R(M_t \oplus M_{t-1}) \tag{6}$$

Let $M'_t$ denote the result of the XOR operation of two consecutive matrices, $M_t$ and $M_{t-1}$. Then, the XOR operation eventually removes most portions of legitimate traffic in the matrix $M'_t$ because legitimate traffic lives longer than one time unit so the portion of legitimate traffic is eliminated by the XOR operation. Experimental results in the next section show that the XOR operation on the consecutive matrices is a key factor for amplifying the spreading behavior of an Internet worm.

## 6. Evaluation of ADUR

In order to evaluate the effectiveness of ADUR, the experimental results are presented with real traffic. The traffic data is used the packet trace captured at a university network in July 29, 2004. For the purpose of creating a worm epidemic, various types of random scanning packets have been injected into the trace.

### 6.1 Effect of XOR Operation

The ADUR mechanism eliminates the effect of legitimate connections on the rank of traffic matrix, by the use of the XOR operation on consecutive matrices. Figure 5 presents the values of the rank "before" and "after" XOR operation on two consecutive matrices. Since the matrix after the XOR operation holds only the information of new connections on the network, which includes suspicious traffic but excludes legitimate traffic, the rank greatly reduces compared with the matrix before the operation. In other words, the rank will increase sharply when a new worm spreads over the network.

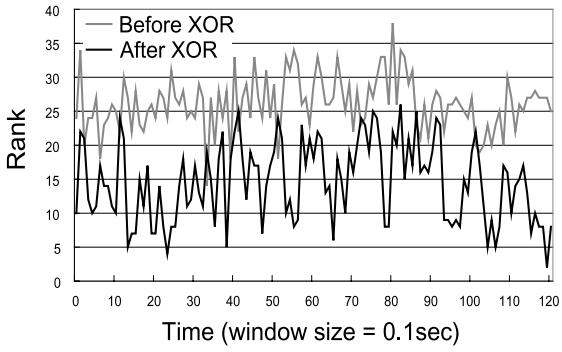



**Fig. 5** The rank of matrix before and after XOR operation.

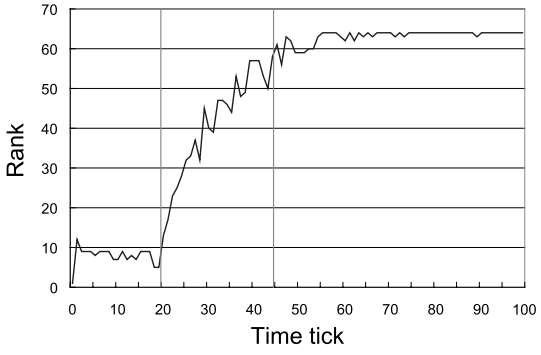



**Fig. 6** The rank of traffic matrix when randomly generated connections are added incrementally.

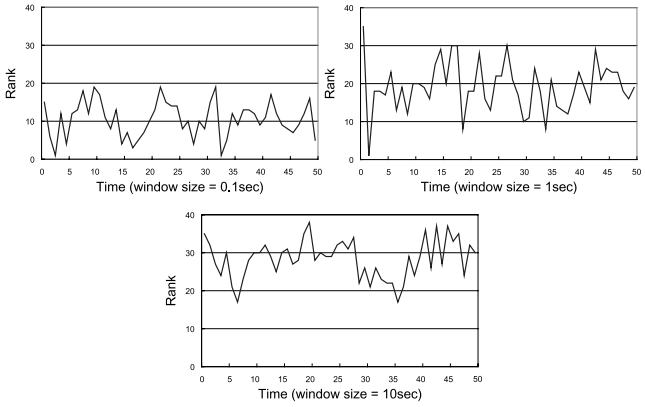



**Fig. 7** The relationship between the rank and the window size.

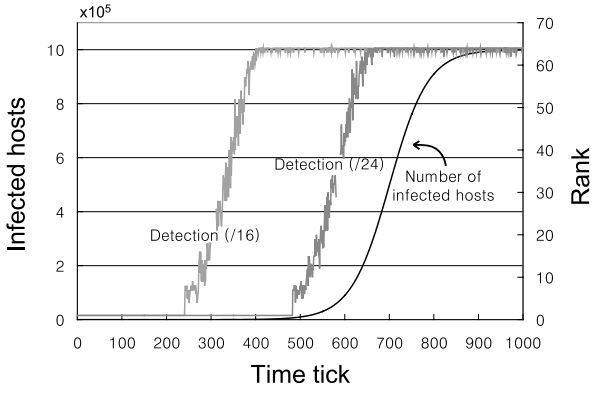



**Fig. 8** The faster response of the ADUR mechanism than the speed of worm infection.

### 6.2 Rank and Random Connection

Figure 6 shows the rank as a function of the number of random connections. In this experiment, we inject one additional connection per one time unit, after passing the 20-th time tick. When more than 25 random connections are added, the rank becomes greater than 60. This demonstrates the rapid transition from the "calm" state to the "ebbing" state, by initial connection attempts of an Internet worm. Since conventional worms generate thousands of new connections per second, ADUR can detect new worms in an early stage of worm propagation.

### 6.3 Effect of Window Size

Traffic matrix $M$ is constructed by traffic gathered for a given time period, called "window size," then the matrix rank is measured after passing every window size. This unit time is counted as one time tick in this paper. Here, an experiment is conducted for measuring the effect of window size.

Figure 7 presents the rank of traffic matrix with three different window sizes. Each graph represents the rank in a different window size using the same traffic data. As the window size increases, the amount of traffic for construction matrix $M$ also increases. It implies that the rank of $M$ will increase as the window size increases. However, this increment is quite limited in a certain boundary, e.g. 20 as presented in Fig. 7. Contrarily, adding random connections will greatly increase the range of the rank. It demonstrates that the ADUR mechanism is robust to the window size, which shows the possibility of applying ADUR to heavy traffic and high-speed networks.

### 6.4 The Dynamics of Rank

In order to evaluate the effectiveness of ADUR, the dynamics of ranks are measured as the worm proceeds. Figure 8 shows the number of infected hosts on the Internet and the dynamics of the rank of a traffic matrix. The number of packets monitored is determined by the size of a monitoring network. Namely, the number of incoming packets on a /24 network is determined by the ratio $2^8/2^{32}$ of the number of total packets generated by the AAWP model.

The rank is much sensitive so that it responds quicker than the speed of infection. Two different networks, i.e., /24 and /16 networks, were considered as a monitoring network. Monitoring larger networks can provide better looking glasses that result in earlier detection. By monitoring a /16 network, the worm can be detectable at 400 unit time, where less than 1% infection is achieved. Even in a small network such as a /24 network, the symptom can be caught

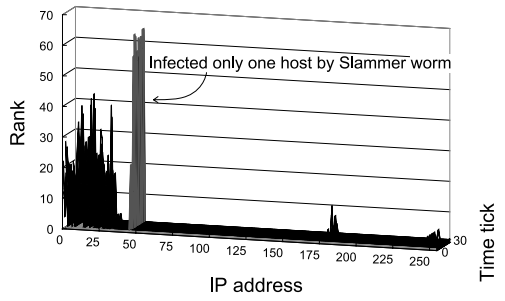



**Fig. 9** Rank distribution for a /16 network, where only one host is infected by Slammer. This graph exposes the location of the infected subnet.

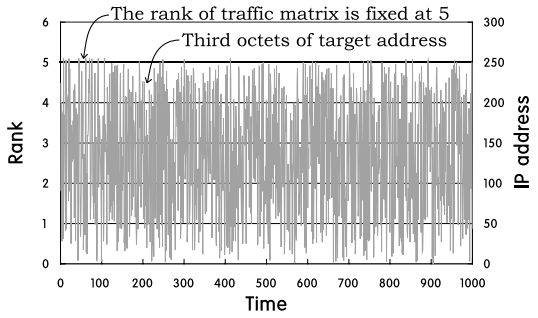



**Fig. 10** The rank of traffic matrix when infected by sequential scanning worm.

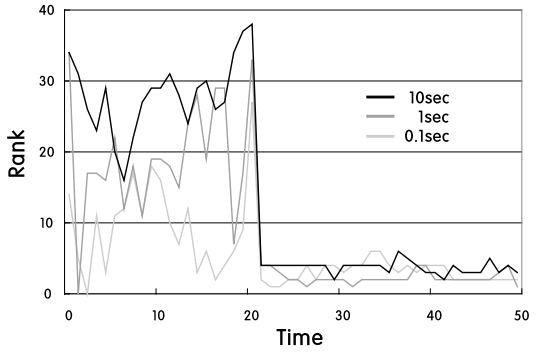



**Fig. 11** The rank of a traffic matrix when a network is infected by a sequential scanning worm.

rapidly such as 650 unit time, where only 20% of hosts in the network are infected, as presented in Fig. 8. This implies that the ADUR mechanism is effective even by monitoring a small network.

### 6.5 Detecting the Slammer Worm Using ADUR in a /16 Network

The effectiveness of ADUR is measured in real network traffic. The traffic is gathered in a university network and worm traffic is injected into the network, by one host being infected with the Slammer worm. The rank distribution is presented in Fig. 9. This is the situation where one host located in an unused network is infected. Figure 9 demonstrates the case where the infection of the subnet 48, which means the third octet is 48 in a /16 network. In Fig. 9, the size of a traffic matrix is 256×64×64. There are several ways to construct a traffic matrix and a 256×64×64 matrix is an example. The rank of normal traffic is less than forty. However, the rank of the traffic with a single host infection becomes greater than sixty. It shows that, even though only one host is infected by a worm, the ADUR mechanism can be useful to detect such infection and provide additional information such as infected subnet locations.

### 6.6 Detecting Non-random Scanning Worms

One can doubt whether ADUR can work properly for detecting worms which do not scan randomly. One way against random scanning is sequential scanning, which selects the target IP addresses sequentially.

Figure 10 shows the rank of the traffic matrix when a worm scans a /24 network sequentially and ADUR do not acquit XOR operation. In this case, the rank is fixed at 5. The worm generates a /24 IP addresses sequentially for next targets, of which the third octet is unchanged but the forth octet is changing sequentially. When expressing these IP addresses on the traffic matrix, two identical rows are expressed 16 times repeatedly due to the fixed third octet. If Gaussian elimination is executed on the traffic matrix, 59 rows will become zero. This means that only five rows among 64 rows remain after Gaussian elimination. The first four elements of the five rows are 0000, 0001, 0010, 0100 and 1000, respectively. Since the forth octet is changing sequentially, all rows excepting the five rows will become zero by Gaussian elimination. Therefore, only the five rows among 64 rows contributes the rank, which results in the fixed rank of five. Note that the row whose first four elements are 0000 is not always all zero's. Since the rest of 60 elements in the row are from 0001 to 1111, the row starting with 0000 not always eliminated but contributes the rank. The rows starting with 0001, 0010, 0100 and 1000 can express other 59 rows since they are the basis of the rows. Therefore, the rank of the traffic matrix by a sequential worm is fixed at 5, and further decreases when applying the XOR operation.

Figure 11 shows the rank of a traffic matrix when a sequential scanning traffic is injected into the normal traffic at a /24 network in a university campus. In Fig. 11, the sequential scanning traffic is injected after 20th time tick. And values of rank before 20th tick are the state of a /24 normal network. In Fig. 11, the rank is measured at three different time intervals, which are 0.1, 1 and 10 seconds. The sequential scanning traffic is stored on the traffic matrix sequentially. And the normal traffic is injected on a traffic matrix per a time tick uniformly. For example, if there are 100 normal connections per a time unit, a normal connection is injected on a traffic matrix at one over 100 of a time unit uniformly. Connections in the normal traffic can be overwritten by the sequential scanning traffic, and vice versa. As a result, the rank under a sequential worm attack is sustained at less than or equal to 5. Even though the amount of traffic stored in

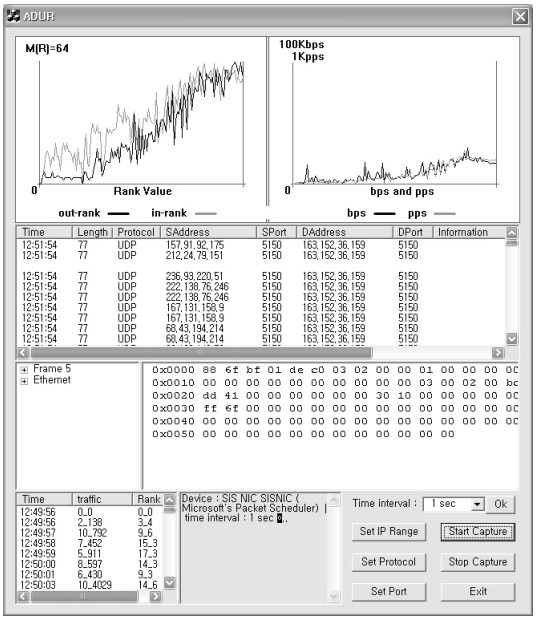



**Fig. 12** Application program of the ADUR mechanism.

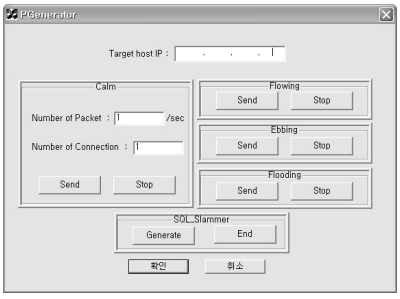



**Fig. 13** Packet generator using the AAWP model.

the matrix increases, the rank is sustained at the low value, as shown in Fig. 11. Thus, the sudden decrement of the rank into a small range, i.e. 5, can be regarded as a metric for sequential worm detection.

### 6.7 Evaluation on Real Networks

We have implemented the ADUR mechanism as an application program, which is shown in Fig. 12. Also, we have applied the ADUR program to real network environments where the size of a monitoring network is a /24 network on a university campus. The ADUR program expresses the rank of incoming traffic, shortly *in-rank*, and the rank of outgoing traffic matrix, shortly *out-rank*. As well, bits per second (bps) and packets per second (pps) are also shown in the program. The top-left graph of the program shows in-rank and out-rank, and the top-right graph shows bps and pps. The detailed information is located under the lower part of the ADUR program[†].

A packet generator is developed for injecting worm traffic into normal traffic, which is shown in Fig. 13. The generator produces the worm traffic following the AAWP model, and also has the function to replay the traffic of Slammer worm. The generator can realize the worm traffic according to the state of a network.

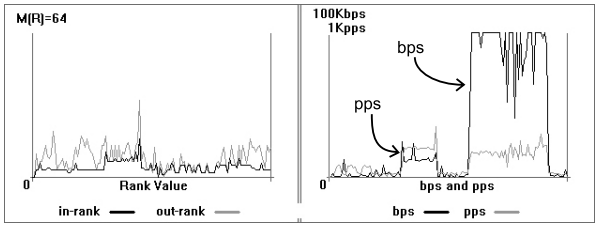



**Fig. 14** In the calm state, both in-rank and out-rank are not increased. If nmap scans ports of a target host, only pps increases. And only bps increases by P2P file transmission.

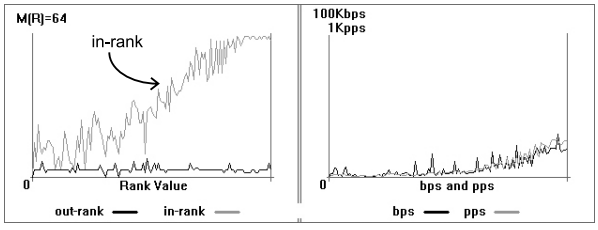



**Fig. 15** In the flowing state, only in-rank increases because the monitoring network is attacked by other network where is infected by the worm.

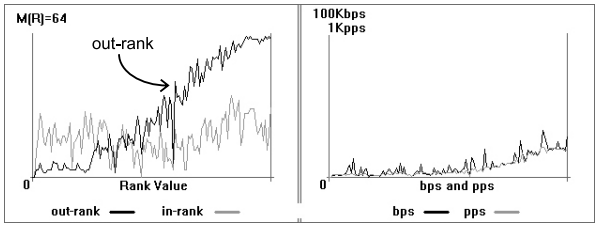



**Fig. 16** In the ebbing state, only out-rank increases because the monitoring network is infected by the worm.

Figure 14 presents the monitoring network in a calm state. The traffic is generated in order to evaluate under portscanning and P2P file transmission. In the portscanning state, only the pps increases. In the P2P file transmission state, only bps increases. However, two ranks are remaining unchanged. It shows that the ADUR has no effect where the traffic on the monitoring network has no randomness, even under heavy traffic such as portscan and P2P applications.

Figure 15 shows the flowing state. If the Internet is under worm epidemics but the local network is not infected yet, the infection attempts from external networks increase the rank of incoming traffic. In Fig. 15, it is shown that ADUR can detect the state of worm flowing properly.

The ebbing state is shown in Fig. 16. In the ebbing state, the out-rank increase because the local network is infected by the worm. Figure 17 presents the flooding state, where in-rank and out-rank increase drastically. This situation is also produced by the traffic generator we developed.

[†]The application program of ADUR can be obtainable at http://ccs.korea.ac.kr/ADUR.

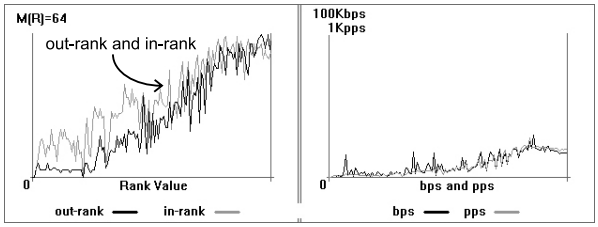



**Fig. 17** The flooding state, both in-rank and out-rank increase because the whole Internet including the monitoring network is infected by the worm.

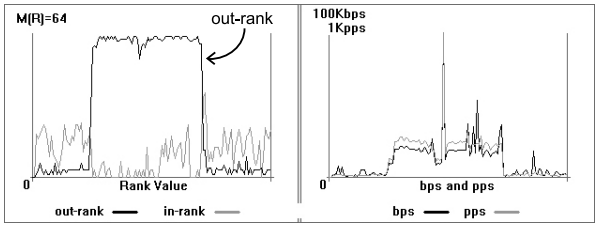



**Fig. 18** The infected state by Slammer worm, since the outgoing traffic to infect increases, out-rank increases.

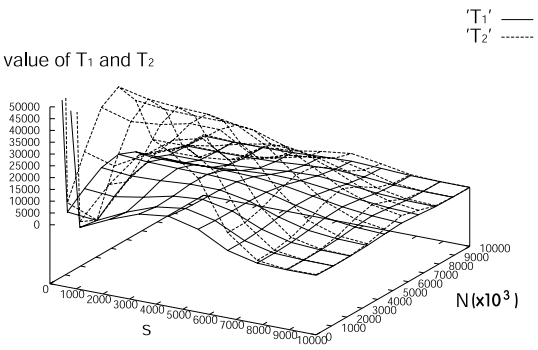



**Fig. 19** Comparison of $T_1$ with $T_2$.

We show that ADUR can detect the worm epidemic effectively. Also, the current spreading situation is clearly identified among four different states of worm infection.

Instead of artificial traffic using the AAWP model, a real worm traffic is captured and replayed for evaluating the ADUR mechanism. Figure 18 is the state where only one host on the monitoring network is infected by the Slammer worm. The host, infected by Slammer, generates a number of random IP addresses. As a result, the increased random outgoing traffic raises out-rank. It is shown that ADUR can detect the Internet worm even when only one host is infected in a monitoring network.

### 6.8 ADUR Sensitivity

We have measured the sensitivity of ADUR compared with the infection speed of a worm. In Fig. 19, the $T_1$ is the time when the rank is greater than 60 on the traffic matrix. The $T_2$ is the time when the number of infected hosts is greater than 90 percent of all vulnerable hosts. The $s$ is the scan rate per second of the worm. The $N$ is the number of vulnerable hosts on the Internet. In Fig. 19, it can be seen that the ADUR can rapidly detect worm propagation.

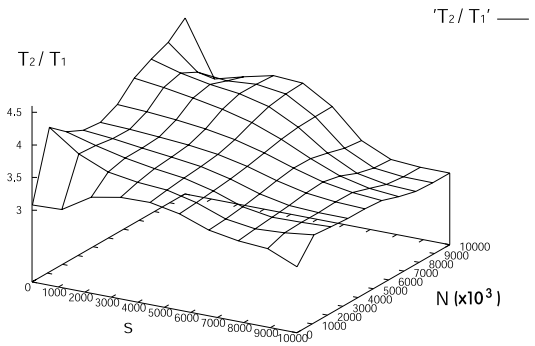



**Fig. 20** $T_2$ over $T_1$.

In Fig. 20, if the scan rate is low and the number of vulnerable hosts is high, the ADUR can detect the early stage. If the number of vulnerable hosts on the Internet is ten million and the scan rate is ten per second, the ADUR can detect worm propagation using the AAWP model after fifty three hosts on the Internet are infected by the worm.

From this evaluation, we show that ADUR is highly sensitive so that an Internet worm can be detectable three times faster than the wide spreading over the majority of vulnerable hosts (90% hosts).

## 7. Conclusion and Future work

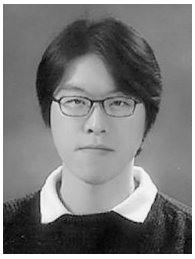
In this paper, we proposed an unknown worm detection algorithm, named ADUR. The proposed mechanism measures the degree of randomness on the distribution of destination addresses in order to detect a worm which scans target hosts randomly. Matrix expression of network traffic and simple XOR operation on two consecutive matrices provide a worm spreading indicator by measuring the rank of the matrix. This paper demonstrates that the ADUR mechanism can detect unknown worms in the early stages of worm spreading, robust to the size and speed of a network and the volume of traffic.

In the future, research will be conducted relating to various methods of traffic matrix generation and the proposed system will be extended to cope with many other attacks on the Internet, such as IP spoofing DoS attacks, DDoS attacks with distributed agents, mass-mailing virus and messenger spam senders. In addition, the method for calculation of the rank of the traffic matrix will be improved, in order to simplify the implementation on a real system. In addition, this mechanism will be executed in high speed networks and an attempt will be made to experiment with greater number of instances.

#### References

[1] R. Russell and A. Machie, "CodeRed II worm," Tech. Rep., Incident Analysis, Security Focus, Aug. 2001.
[2] A. Machie, J. Roculan, R. Russell, and M.V. Velsen, "Nimda worm analysis," Tech. Rep., Incident Analysis, Security Focus, Sept. 2001.

[3] CERT/CC: CERT Advisory CA-2001-26 Nimda Worm, http://www.cert.org/advisory/CA-2001-26.html, Sept. 2001.
[4] D. Song, R. Malan, and R. Stone, "A snapshot of global Internet worm activity," Tech. Rep., Arbor Network, Nov. 2001.
[5] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver, "Inside the slammer worm," IEEE Magazine of Security and Privacy, vol.1, pp.33–39, July/Aug. 2003.
[6] C.C. Zou, L. Gao, W. Gong, and D. Towsley, "Monitoring and early warning for Internet worms," Proc. ACM CCS, pp.190–194, Oct. 2003.
[7] J. Wu, S. Vangala, L. Gao, and K. Kwiat, "An efficient architecture and algorithm for detecting worms with various scan techniques," Proc. NDSS, Feb. 2004.
[8] V.H. Berk, R.S. Gray, and G. Bakos, "Flowscan: Using sensor networks and data fusion for early detection of active worms," SPIE AeroSense, vol.5071, pp.92–104, 2003.
[9] V.H. Berk, G. Bakos, and R. Morris, "Designing a framework for active worm detection on global networks," The IEEE International Workshop on Information Assurance, pp.13–24, March 2003.
[10] C. Leckie and R. Kotagiri, "A probabilistic approach to detecting network scans," The 8th IEEE Network Operations and Management Symposium (NOMS 2002), pp.359–372, April 2002.
[11] J. Jung, V. Paxon, A.W. Berger, and H. Balakrishnan, "Fast portscan detection using sequential hypothesis testing," Proc. IEEE Symp. Security and Privacy, pp.211–225, IEEE CS Press, 2004.
[12] D. Whyte, E. Kranakis, and P.van Oorschot, "DNS-based detection of scanning worms in an enterprise network," Proc. 12th Annual Network and Distributed System Security Symposium, 2004.
[13] D. Whyte, P.van Oorschot, and E. Kranakis, "ARP-based detection of scanning worms within an enterprise network," Proc. Annual Computer Security Applications Conference (ACSAC 2005), Tucson, AZ, Dec. 2005.
[14] B. Wiley, "Curious yellow: The first coordinate worm design," Oct. 2002. http://blanu.net/curious_yellow.html
[15] S. Staniford, V. Paxson, and N, Weaver, "How to own the Internet in your spare time," USENIX Security Symposium, pp.149–169, Aug. 2002.
[16] N. Weaver, "Warhol worms: The potential for very fast Internet plaques," 2001. http://www.cs.berkeley.edu/~nweaver/warhol.html
[17] M. Eichin and J. Rochlis, "With microscope and tweezers: An analysis of the Internet virus of November 1988," IEEE Symposium on Security and Privacy, pp.326–343, 1989.
[18] C.C. Zou, W. Gong, and D. Towsley, "CodeRed worm propagation modeling and analysis," Proc. ACM CCS, pp.138–147, Nov. 2002.
[19] Z. Chen, L. Gao, and K. Kwiat, "Modeling the spread of active worms," IEEE INFOCOM, 2003.
[20] H. Anton, Elementary linear algebra, 7th ed., John Wiley & Sons, 1994.
[21] G. Marsaglia and L.H. Tsay, "Matrices and the structure of random number sequences," Linear Algebra Appl., vol.67, pp.147–156, Elsevier Science, 1985.
[22] G. Marsaglia, "DIEHARD: A battery of tests of randomness," 1996. http://stat.fsu.edu/~geo/diehard.html



**Hyundo Park** received the BS, MS degrees from the Department of Computer Science and Engineering, and also received the BS degree form the Department of Math at Korea University, Seoul, Korea. He is currently working towards the PhD degree in the Department of Computer Science and Engineering at Korea University.

**Heejo Lee** received his BS, MS, PhD in Computer Science and Engineering from Pohang University of Science and Technology (POSTECH), Korea in 1993, 1995 and 2000, respectively. Since 2004, he has been an assistant professor at the Department of Computer Science and Engineering, Korea University, Seoul, Korea. From 2001 to 2003, he was at Ahnlab, Inc. as Chief Technology Officer (CTO) and Director of Technology Planning Department. From 2000 to 2001, he was a Post Doctoral Research Associate at the Network Systems Lab of the Department of Computer Sciences, and at the Center for Education and Research in Information Assurance and Security (CERIAS), Purdue University. His research interest includes computer and communication security, parallel scientific computing, and fault-tolerant computing.

**Hyogon Kim** is an associate professor at the Korea University. He got his PhD from the University of Pennsylvania in 1995. Prior to joining the Korea University, he was a research scientist at Bell Communications Research (Bellcore). His research interests include Internet protocols and applications, wireless networking, network security, and computational neuroscience.