Multi-hash based Pattern Matching Mechanism for High-Performance Intrusion Detection

Byoungkoo Kim, Seungyong Yoon, and Jintae Oh

***Abstract*—**Many Network-based Intrusion Detection Systems (NIDSs) are developed till now to respond these network attacks. As network technology presses forward, Gigabit Ethernet has become the actual standard for large network installations. Therefore, software solutions in developing high-speed NIDSs are increasingly impractical. It thus appears well motivated to investigate the hardware-based solutions. Although several solutions have been proposed recently, finding an efficient solution is considered as a difficult problem due to the limitations in resources such as a small memory size, as well as the growing link speed. Therefore, we propose the FPGA-based intrusion detection technique to detect and respond variant attacks on high-speed links. It was designed to fully exploit hardware parallelism to achieve real-time packet inspection, to require a small memory for storing signature. The technique is a part of our system, called ATPS (Adaptive Threat Prevention System) recently developed. Most of all, the proposed system has a novel content filtering technique called Table-driven Bottom-up Tree (TBT) for exact string matching. However, as the number of signatures to be compared is growing rapidly, the improved mechanism is required. In this paper, we present the multi-hash based TBT technique with memory-efficiency. Simulation based performance evaluations showed that the proposed technique used on-chip SRAM less than 20% of the one-hash based TBT technique. Finally, experimental results about our system show a consistent performance in traffic level and had nothing to do with increasing number of signatures applied.

***Keywords*—**Intrusion Detection, Pattern Matching, Heuristic Analysis, Memory-efficiency

I. Introduction

THE fast extension of inexpensive computer networks also has increased the problem of unauthorized access and tampering with data. As a response to increased threats, many NIDSs have been developed to serve as a last line of defense in the overall protection scheme of a computer system. These NIDSs have two major approaches; misuse intrusion detection and anomaly intrusion detection, but most of existing NIDSs, such as Snort, NFR, and NetSTAT, only employs the misuse detection approach for reducing a lowering of performance to the minimum. Also, most NIDS has concentrated on catching and analyzing only the audit source collected on Fast Ethernet links. However, with the advancement of network technology, Gigabit Ethernet has become the actual standard for large network installations. Therefore, the effort of performing NIDS on high-speed links has been the focus of much debate in the intrusion detection community, and several NIDSs that are run on high-speed links actually have been developed [3]. But, these NIDSs are still not practical because of technical difficulties in keeping pace with the increasing network speed, and real-world performance also will likely be less. Therefore, there is an emerging need for security analysis techniques that can keep up with the increased network throughput. Most of all, content filtering problems have been extensively studied, and many of the proposed solutions are based on the general purpose pattern matching algorithms such as Boyer-Moore (BM)[10], Aho-Corasick (AC)[11], and Wu-Manber (WM)[12]. Although these algorithms greatly improve pattern matching speed compared to naive string matching techniques, providing real-time content filtering in high-speed networks is still challenging. In the previous work, we proposed our Gigabit IDS, called ATPS, to detect and respond attacks on the high-speed network [1]. The proposed system has a novel content filtering technique called TBT [2]. This paper presents the improved mechanism, which used on-chip SRAM less than 20% of the previous TBT technique.

The remainder of the paper is structured as follows. The next section presents related works about early studies of NIDS. Then, section 3 presents the architecture of our system, and describes the components. Section 4 shows the multi-hash based TBT technique with memory-efficiency. Section 5 shows the simulation based performance evaluations about the improved TBT technique, which is compared with the previous technique. Section 6 briefly introduces the prototype that we have developed, and shows the experimental result of our prototype system. Finally, we conclude and suggest directions for further research in section 7.

II. Related Work

Basically, IDS is classified into host-based IDS and network-based IDS. Audit sources discriminate the type of IDSs based on the input information they analyze. Host-based IDS analyzes host audit source, and detects intrusion on a single host. However, NIDS uses the network as the source of security-relevant information. Consequently, NIDS moves security concerns from the hosts and their operating systems to the network and its protocols. Besides, IDS is classified into two major approaches based on the detection method they operate; misuse intrusion detection and anomaly intrusion detection. The first, misuse intrusion detection is based on the

Manuscript received December 12, 2008: This work was supported in part by Korea Ministry of Information and Communication under "Next Generation Security System Development" Project.

Byoungkoo Kim is with Electronics and Telecommunications Research Institute, 161 Gajeong-dong, Yuseong-gu, Daejeon, Korea (phone: +82-42-860-4888; fax: +82-42-860-; e-mail: bkkim05@etri.re.kr).

Seungyong Yoon is with Electronics and Telecommunications Research Institute, 161 Gajeong-dong, Yuseong-gu, Daejeon, Korea (e-mail: syyoon@etri.re.kr).

Jintae Oh is with Electronics and Telecommunications Research Institute, 161 Gajeong-dong, Yuseong-gu, Daejeon, Korea (e-mail: showme@etri.re.kr).

detection of intrusions that follow well-defined patterns of attack exploiting known vulnerabilities. Therefore, this approach is based on the pattern of known misuse or abnormal behavior. This approach is very efficient, but this is hard to detect new intrusion patterns. Also, this approach is possible to draw false negative detection. The second is based on the detection of anomalous behavior or the abnormal use of the computer resource. This approach is based on the database of normal behavior. Therefore, it costs a great deal, but this approach is capable of detecting unknown intrusions. Also, It is possible to draw false positive detection, but hard to set a threshold value. Like this, these approaches all have each advantages and disadvantages. However, anomaly intrusion detection approach does not apply to operate real-time intrusion detection, since it tends to be computationally expensive because of several maintained metrics that are updated after every system activity. Therefore, most IDSs employed misuse detection approach only because of performance and availability consideration. Primary approaches to misuse detection might be implemented by on the following techniques: Expert Systems, State Transition Analysis, Model based Approach and so forth. But, these techniques do not present the definite mechanism, and sometimes contain complex and ambiguous concepts. Also, these approaches are not suited as a speedy detection mechanism in high-speed network environments. Therefore, most IDSs focus on more speedy and exact pattern matching algorithm and detection mechanism about Denial of Service (DoS) attacks and Port Scan attacks.

With the widespread use of the Internet, IDSs have become focused on network attacks. Therefore, most IDSs employed network-based IDS [14]. NIDS uses the network as the source of security-relevant information. Essential to almost every NIDS is the ability to search through packets and identify content that matches known attacks. As network technology presses forward, space and time efficient string matching techniques have been important for identifying these packets at line rate. Therefore, software solutions in developing high-speed NIDSs are increasingly impractical. It thus appears well motivated to investigate the hardware-based solutions. With the advance of hardware technologies, there have been several attempts to catch up the line speed using Field Programmable Gate Arrays (FPGAs) [9]. Cho et al. [6] proposed to use parallelized patterns (rule units) to find matches in a four-byte input stream on every clock cycle. In addition, Sidhu et al. [5] and Moscola et al. [13] mapped regular expressions into a FPGA using Nondeterministic Finite Automation (NFA) to minimize the space required for pattern matching. Most FPGA-based approaches are successful in keeping pace with line speed by exploiting the high degree of parallelism and reconfigurable hardware characteristics. However, hardware resource consumption linearly increases as the number of patterns and the number of characters to be stored increase. Another approach to achieve the high-speed content filtering is to use a fast parallel pattern matching circuit, called a Content Addressable Memory (CAM). Although the CAM-based solutions are relatively easy to implement, and can exploit fine grain pipelining in general, high hardware cost and power consumption, as well as high area cost are the main concerns. On the other hand, Dharmapurikar et al. [19] addressed the high-speed content filtering problem with parallel Bloom filters. Here, hardware-based Bloom filters, i.e., multiple independent hash functions, are programmed with each possible length of all signatures, and then a detection process is performed by testing membership of a string extracted from packet payload for the stored signature set. The main concern of this technique is that hardware parallelism can be limited by the wide variation in the signature length as noted in [20]. This technique was extended using the Extended Bloom Filter (EBF) [21] recently. Finally, Tuck et al. [22] proposed to use a bit-mapping and path compression technique to reduce the size of the memory required to store the SNORT rule set. Although the authors showed that the compressed version of the AC algorithm exhibits the deterministic worst-case running time, and thus is suitable for hardware implementation, the implementation details were not thoroughly examined.

As appeared in the related works, the content filtering at gigabit line speeds is still a challenging problem due to the limitations in resources such as memory size and the growing line speed. In addition, the number of signatures to be compared is growing rapidly and thus giving the signature explosion problem. With current processor and hardware technologies, neither increasing the speed of network processor nor providing more network processors to keep up with the multi-giga bits links is practical and cost-efficient. In addition, the number of signatures to be compared with the payload of each packet has been doubled during the last two years. For example, the number of signatures in SNORT [7] has increased from 1,500 in 2003 to 2,800 in 2005. Therefore, software-only-solutions in developing high-speed NIDS are increasingly impractical. It thus appears well motivated to investigate the hardware-based solutions. In particular, several content filtering techniques have been proposed to used FPGAs for taking advantage of the hardware parallelism and the configurable nature of the device [4], [5], [6]. However, the FPGA has to be reprogrammed to modify the signature database, which could lead the network infrastructure at stake in the presence of the zero-day attack [23]. Another concern in FPGA-based solutions is that an on-chip SRAM is generally favored for constructing signature database primarily due to the faster access time compared to an off-chip SRAM. However, the size of an on-chip SRAM is generally hundreds of kilobytes, whereas the space required for storing only all the characters in the latest SNORT signatures is almost 512 Kilobytes. Therefore, finding efficient techniques, which can minimize the memory consumption, is becoming a key success factor in the development of high-speed NIDSs.

This paper presents a novel FPGA-based content filtering technique, called TBT, which is a multiple hash implementation of a bottom-up tree. The hashing scheme helps in reducing the data access time, while the bottom-up tree helps in minimizing the memory consumption in TBT. The TBT was evaluated through extensive simulations, and implemented in a

XILINX FPGA, XC2VP70 [8].

III. System Description

In this section, we briefly introduce the architecture of our system and components of the architecture called ATPS. And then, we present efficient detection techniques for high-speed intrusion detection and response. Through these techniques, our proposed system can perform the real-time operations as inline-mode.

A. System architecture and components

Our system is aimed at real-time network-based intrusion detection based on misuse detection approach [15]. As shown in the figure 1, the proposed system consists of two parts; Application Task for policy management, alert management and system management, and Security Engine Board for wire-speed packet forwarding, packet preprocessing, high-performance intrusion detection and response.

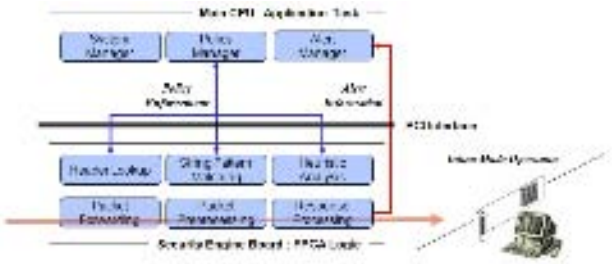

Fig. 1 The system architecture of ATPS

Most of all, for detecting network intrusions more efficiently on high-speed links, Security Engine Board is composed of several sub FPGA Logics. Here, communication between Security Engine Board and Main CPU is run through PCI interface. Communication through PCI interface is divided into two channels. One is a channel for policy enforcement. The other is a channel for alert information transmission. Through the interoperability of these components, our system analyzes data packets as they travel across the network for signs of external or internal attack. That is, the major functionality of our system is to perform the real-time traffic analysis and intrusion detection on high-speed links. Therefore, we focus on effective detection strategies applied FPGA Logics.

As shown in the figure 2, Security Engine Board is composed of three FPGA chips and one FPGA chip for PCI interfacing. First, ATIE (Anomaly Traffic Inspection Engine) FPGA chip uses the XILINX XC2VP70 FPGA chip. And, it is connected to the PM3386 for incoming packet forwarding. Also, it uses external TCAM and SRAM for incoming packet scheduling and management. Briefly, the main function of ATIE is the wire-speed packet forwarding and response coordinating such as alert message generation and packet filtering. Basically, incoming packets from PM3386 is sent to PPE FPGA chip, and if it is determined as attack according to the analysis result from other FPGA chips, alert information is sent to the main CPU through FPGA chip for PCI interfacing. Second, PPE (Pre-Processing Engine) FPGA chip uses the XILINX XC2VP50 FPGA chip. And, it uses two external SRAM for session management, IP de-fragmentation and TCP reassembly. Briefly, the main function of PPE FPGA chip is to process the before steps for intrusion detection. The preprocessing function supports the SPI (Stateful Packet Inspection) based intrusion detection and IDS evasion attack detection. Finally, IDE (Intrusion Detection Engine) FPGA chip uses the XILINX XC2VP70 FPGA chip. It uses three mechanisms for high-performance intrusion detection; Flexible Header Combination Lookup Algorithm for packet header pattern matching, Linked Word based Store-less Running Search Algorithm for string pattern matching about packet payload, and Traffic Volume based Heuristic Analysis Algorithm for DoS and Port-scan attack detection.

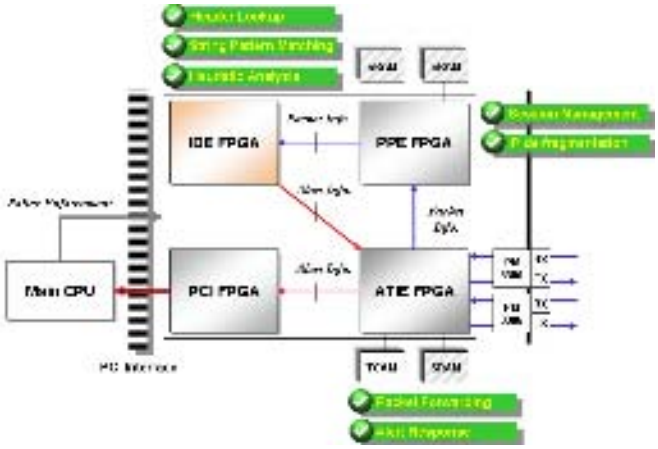

Fig. 2 the FPGA Chips Arrangement of Security Engine Board

Through these mechanisms, it executes the high-performance intrusion detection without packet loss.

B. Detection Mechanisms

The detection mechanism of our system is mainly performed on IDE FPGA Chip. For effective high-performance intrusion detection, our system has three detection mechanisms. One is the header lookup mechanism for flexible header combination lookup, another is the payload matching mechanism for packet payload matching, and the other is heuristic analysis mechanism for DoS and Port-scan attack detection. Besides, session management mechanism on PPE FPGA Chip supports SPI based intrusion detection.

1) Header Lookup Mechanism

Header lookup mechanism is performed by flexible header combination lookup algorithm. This algorithm compares pre-defined header related rule-sets with header information of incoming packets. If the incoming packet is matched with existing header patterns, 256 bits match result is sent to payload matching logic and traffic volume based heuristic analysis logic. As shown in Figure 3, this mechanism uses three memory maps: TCAM Lookup Map for each header field matching, Rule Combination Check Map for multiple header field matching, and Sequence Check Map for don't care field matching.

Fig. 3 Header Lookup Mechanism

First, TCAM Lookup Map is composed of three internal TCAM: 8bits lookup map for 8bits header fields such as ICMP type, TCP flags, and so forth, 16bits lookup map for 16bits header fields such as service port value, IP identification, and so forth, and 32bits lookup map for 32bits header fields such as IP address field. These maps have each 128entries, 64entries. In other words, our system supposes that header values of all rule-sets are within the entry number of each map. The match result of these lookup maps is used by Rule Combination Check Map and Sequence Check Map. Second, Rule Combination Check Map is composed of 256*256 block select RAM memories. Match address from TCAM Lookup Map is used for 256bits result of this memory map. This 256bits result presents the rule-set matching result about current matching field. If match result of ICMP type field is "{255{2'b0}, 2'b1}", the first rule-set is to be matched. Therefore, it is possible to support the multiple matching. In other words, our system supposes that the combination of all rule-sets is within the 256 entry numbers. Finally, Sequence Check Map is composed of 32*256 block select RAM memories, and includes don't care information about current matching field. If don't care information of ICMP type field is "{255{2'b0}, 2'b1}", the first rule-set is always to be matched. In other words, our system supposes that the kinds of packet header fields are within the 32 entry numbers. The result of this map is combined with result of Rule Combination Check Map.

Basically, the above operations are performed recursively about all packet fields of incoming packet. Finally, updated 256bits match result is referred by logics for packet payload matching and traffic volume based analysis.

2) Payload Matching Mechanism

Payload matching mechanism is performed by linked word based store-less running search algorithm. This algorithm compares pre-defined packet payload related rule-sets with packet payload information of incoming packets. If the incoming packet is matched with existing payload patterns, alert message is generated according to the 256bits header lookup result. For this operation, this algorithm uses the signature tokenizing technique. Each token size has boundary of size5 or 7 because of the limit of block memory in FPGA Chip. Here, same tokens are stored in the same memory space. Through signature tokenizing like this, our system can have about 2,000~3,000 rule-sets in the limited memory storage on FPGA Chip.

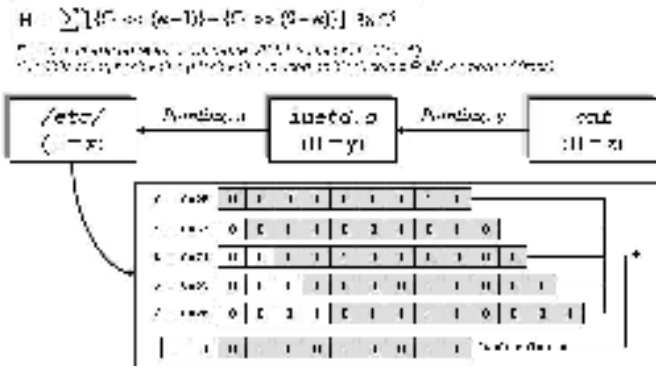

Fig. 4 Linked Word based Store-less Running Search Algorithm

Most of all, our system has the linked word based store-less running search algorithm for performing the payload pattern matching. This algorithm uses the spectrum dispersion technique as shown in the figure 4. The spectrum dispersion technique is method to calculate unique hash values about each signature token. For example, 5bytes "/etc/" pattern has the 9bits hash value "011010011" by sum about shifted values of each character. These hash values are used as the rule memory address for each token. The memory construction like this is performed by Embedded CPU when the system booting is run in advance. After system booting, IDE FPGA Logic performs the hash value calculation about the incoming packet to the unit of byte. If the payload in incoming packet is matched with the pattern in memory pointed by the calculated hash value, then it is checked out which the related reconstructed patterns is matched or not. If all reconstructed patterns are matched with incoming packet, alert message is generated according to the header lookup results. Through these operations, our system performs the pattern matching operation without lowing of performance and packet loss.

3) Traffic Volume based Heuristic Analysis Mechanism

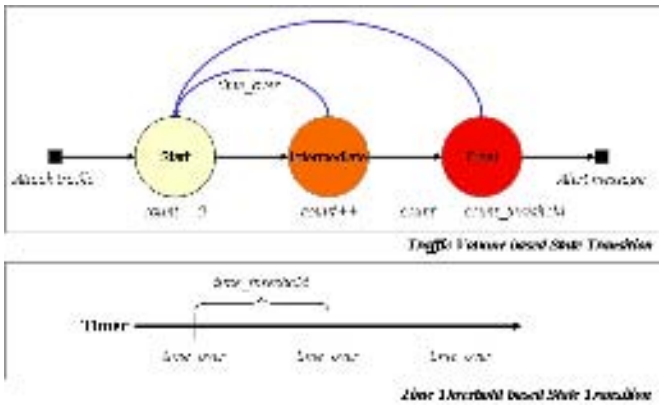

Fig. 5 Traffic Volume based Heuristic Analysis Algorithm

Traffic volume based analysis mechanism is performed by traffic volume based heuristic analysis algorithm. Like pattern matching mechanism, this algorithm also compares pre-defined rule-sets with packet information of incoming packets. But, this mechanism is based on the traffic volume. In other words, this mechanism generates alert message by traffic volume within

time threshold. As shown in the figure 5, if the incoming packet is matched with existing rule-set, then count value of the rule-set is increased, and count threshold and time threshold is checked out. If the count threshold is exceeded by the incoming packet within time threshold, alert message is generated. Through these operations, our system is capable of detecting the DoS and Port-scan attacks such as TCP syn flooding attack, UDP Bomb, SYN/ACK/XMAS Port-scans, and so forth.

4) Session Management Mechanism

Session Management Mechanism supports the SPI based intrusion detection. Because stateless IDS only look at one packet at a time, a lot of false positive alerts generate during attempt to attack using IDS evasion tool, for example, “stick” or “snot”. To prevent this problem, SPI was employed in our system.

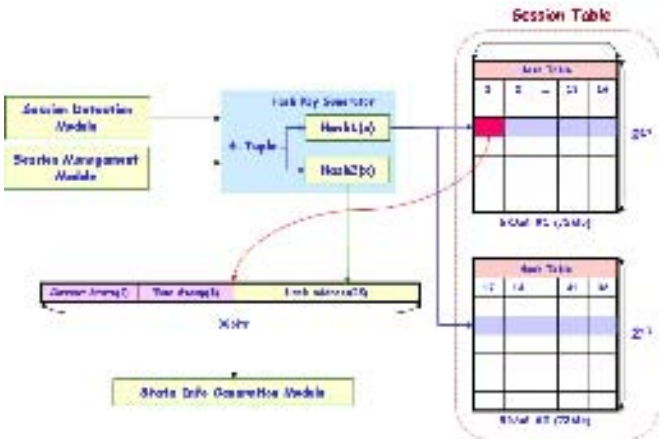

Fig. 6 Session Management Mechanism

For this, the session table stores session entries that are indexed and managed by the hash key generator. 4-tuple information including a source IP address, a destination IP address, a source port, and a destination port is input, as information used to hash a newly received packet, to the hash key generator. Once the packet is inputted, a packet parser extracts this information from the packet. The hash key generator indexes and manages a session entry corresponding to the received packet based on the input 4-tuple information. Hash key generator has a dual hash structure with two different hash functions Hash1(x) and Hash2(x). The hash functions Hash1(x) and Hash2(x) are well-known functions that are used to hash packets. For example, XOR or CRC functions can be used as these hash functions. One hash function “Hash1(x)” is used to generate indices that point to hash sets permitting hash collisions in order to achieve faster session table search. The other hash function “Hash2(x)” is used to generate hash addresses that are used to identify session entries in a hash set pointed by the hash function “Hash1(x)”. Session table may be designed and implemented using two or more SRAM devices, if necessary. In our system, the session table is constructed using two SRAMs (SRAM#1 and SRAM#2), which can be accessed simultaneously or in parallel using a hash set index that is generated by the Hash1(x) to achieve faster session table search.

The session table stores session data of packets inputted from an external network. For efficient session table management, the session table has an N-way set associative session table structure in which each hash set in the session table can include N session entries. The session table shown in figure 6 is a 32-way set associative session table that is constructed using two 72-Megabit SRAMs with each session entry having a length of 36 bits. Each session entry stored in the session table includes current state, time stamp, and hash address parts. The current state part includes current connection state information of a corresponding session, the time stamp part is used to determine which session entry is to be deleted when the session table is full, and the hash address part is used to identify each session entry in the same hash set. The time stamp is updated by an internal timer each time a corresponding session is accessed. If any hash sets of the session table are full so that new session cannot be allocated to the hash sets, current time of internal timer is compared with the time stamp of each session entry to replace the oldest session with a new session.

Session state information is separately managed in embryonic state and established state for timeout mechanism. Embryonic state includes sessions that TCP 3-way handshaking does not finish, on the other hand established state includes completed sessions. It is necessary to manage separately states, because embryonic session needs to have shorter timeout value than that of established session. The SPI devices and computers have vulnerabilities to SYN flooding attack in nature. This mechanism helps to prevent against denial-of-service attack such as SYN flooding. Although a Transmission Control Protocol (TCP) session is terminated without sending an RST or FIN packet, a corresponding session entry is immediately removed if a time stamp in the session entry exceeds a timeout threshold predetermined by the administrator. Accordingly, a session which has been terminated without sending an RST or FIN packet is positively removed from the session table.

The session management mechanism searches the session table according to the received packet. Specifically, it obtains a hash set pointer from Hash1(x) calculated by the hash key generator and then searches the session table for a session entry corresponding to the hash value from Hash2(x). The session management mechanism performs a process for adding, deleting, and changing sessions of the session table in order to maintain the session table. The state info generation module generates state information regarding the direction of the packet and session establishment information and then transmits this information to intrusion detection engine (IDE FPGA) with packet data.

Finally, our work related protocol anomaly detection method is described by Dong-Ho Kang at al. [8] in detail. Through the interoperability of these mechanisms, our system analyzes data packets as they travel across the network for signs of external or internal attack. That is, the major functionality of our system is to perform the real-time traffic analysis and intrusion detection on high-speed links. In this paper, we focus on effective detection strategies applied FPGA Logics. Especially, content

filtering mechanism for payload pattern matching is very important because of hardware resource limitation.

IV. Multi-hash based TBT Mechanism

In this section, we briefly introduce the previous content filtering mechanism for payload pattern matching, called TBT. Then, we present the multi-hash based TBT technique with more memory-efficiency than the previous one hash based TBT technique.

A. Previous TBT Mechanism

In the previous work, we present our novel content filtering mechanism, so called TBT, which is one hash implementation of a bottom-up tree [2]. Figure 7 shows the basic architecture of the previous work.

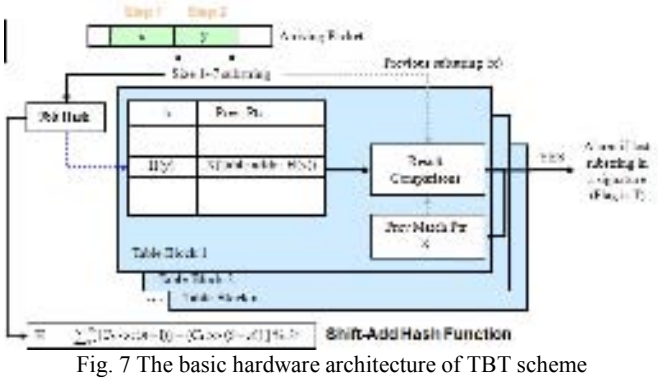

Fig. 7 The basic hardware architecture of TBT scheme

When a packet arrives, the first k-byte substring is extracted from the beginning of the payload. For the successive substrings, the k-byte window shifts toward the end of the payload one byte at a time. Therefore, p - k + 1 substrings are extracted for a p bytes packet. For each substring, the hash function generates an index, and the corresponding slots in all the hash tables are accessed simultaneously. In Fig. 1, the simultaneous accesses for the multiple hash tables are represented with dashed-lines. If the key values in the accessed slots match the substring (of the packet) and the addresses of the accessed slots match the values in the previous slot address registers (PSA registers), then the registers are updated with the matched slot addresses. Otherwise, the PSA registers are cleared. Here, the PSA register is cross-referred by each Table Blocks (TBs), and PSA register comparison is performed only when the matched slot is not the first substring, called the head substring, in a signature. An attack is detected when the last substring (the tail substring) in a signature is matched.

The main advantage of the TBT architecture is its flexibility. Firstly, as discussed in the previous section, one of the main disadvantages in FPGA-based solutions is that the FPGA has to be re-programmed to modify signatures. In TBT, operations such as addition and deletion in the signature database do not involve any hardware modification. Therefore, the signature modification is simple so that any update to the database can be made interactively. Secondly, TBT is flexible in expressing various types of signatures. Currently, 40% of SNORT signatures contain special characters such as * and ?, which represent an arbitrary length of a string and any one byte character, respectively. In general, hardware-based content filtering techniques are targeted to reduce the overhead of exhaustive pattern matching algorithms by exploiting simplicity and regularity in hardware. Therefore, the ability to represent various signatures is limited. However, the current design of TBT can support most of regular expressions easily. For example, content filtering, which involves the signature with a variable length wildcard character, *, is implemented in the PSA register comparison block by simply delaying the register clearing. Thirdly, TBT supports real-time packet content inspection by exploiting hardware parallelism. Although the prototype currently offers 2 Gbps throughput, it can be improved up to 10 Gbps when the clock speed is optimized. Finally, TBT requires a small memory for storing signatures. In fact, only 350 Kilobytes of on-chip memory are used for storing the latest version of SNORT rule, consisting of 2770 signatures. This is less than 30% of memory used in the other techniques for storing 1533 signatures.

B. Multi-hash based TBT Mechanism

To detect certain patterns from packets, the attack patterns (signatures) have to be effectively stored in the memory to minimize memory consumption and to facilitate data retrievals. Therefore, we proposed the design concepts of the TBT mechanism in terms of the data structure and the algorithm. The solution that we propose takes advantage of a bottom-up tree to maximize the memory efficiency, and uses multiple hash tables to improve the run-time performance of the content filtering algorithm. All hash tables used in TBT share the same structure, and a slot in the hash tables consists of four fields: substring, previous node pointer (Prev_Ptr), flags to indicate the head node and the tail node, and a field to store signature specific information such as signature description. For example, figure 8 and figure 9 illustrate how a bottom-up tree is constructed when k is 5 or 7. As shown in the figure 8, the first five-byte substring of "/bin/echo" pattern is equal to the first five-byte substring of "/bin/kill" pattern. Therefore, "/bin/" substring is stored in the same memory space. Through this tokenizing, other patterns are also constructed. Here, a way to improve memory utilization is to use a bottom-up tree, where each child node points to the address of its parents.

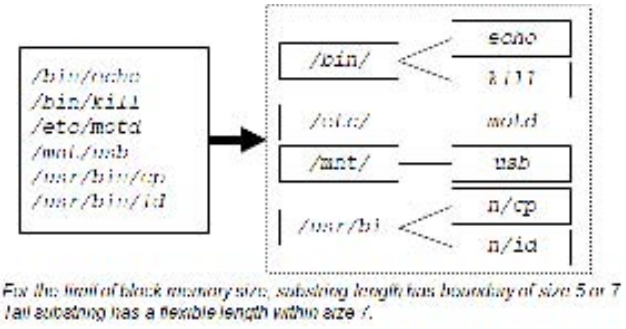

Fig. 8 Signature Tokenizing

Given the signatures, each signature is divided into 5 or 7 byte substrings, and each substring is hashed into a 9bit value,

called a token. A bottom-up tree, then, is constructed using the tokens as showed in figure 9. Here, the sequence of tokens computed from a signature forms a unique path in the tree and, thereby, completely identifies the signature. First, a signature, "/bin/echo", is divided into five-byte substrings, "/bin/" and "echo", and then the substrings are hashed to find the tokens, 30 and 10 respectively. Finally, "/bin/" and "echo" are inserted at slots 30 of the hash table in Table Block 1 (T1) and slots 10 of the hash table in Table Block 4 (T4). Then, the link between the adjacent tokens is represented by Prev_Ptr. Similarly, other signatures are stored in each Table Blocks.

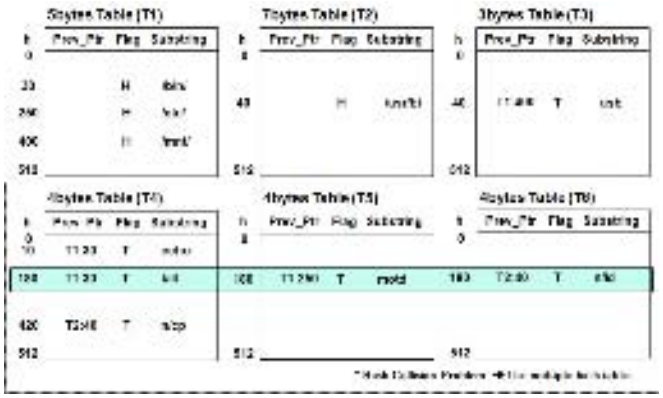

Fig. 9 Hash Collision Problem

Here, the value of slot 180 induces the hash collision problem. For example, "motd" collides with the previously inserted key of "kill". In TBT, collisions can be resolved simply by searching all the hash tables simultaneously and using the first table with unoccupied slots for the corresponding token. Therefore, the new key is inserted in other same copying Table Block. Although the sequential search on multiple tables is an expensive operation when implemented with software, it can be easily converted into a parallel search with hardware. However, the more it does if the hash collision like figure 9 frequently happen, the waste of the hardware resource becomes severe.

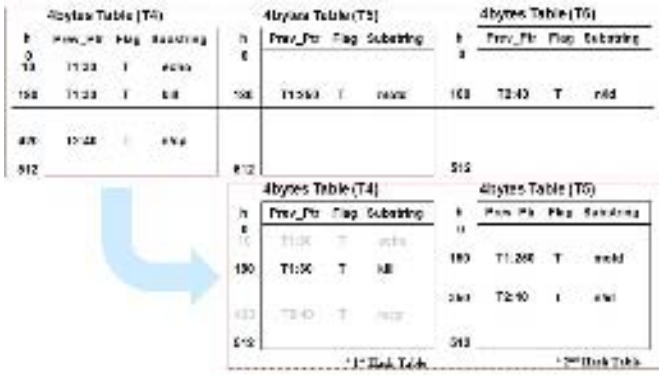

Fig. 10 Multi-hash based Signature Table Configuration

Therefore, we propose the multi-hash based TBT mechanism. If hash collision happens frequently by 1st hash function, signature table configuration by multi-hash based TBT can be performed by 2nd hash function as shown in the figure 10. That is, although the key by 1st hash function collides with the previous inserted key, the key by 2nd hash function can be used. Here, the 1st hash function and the 2nd hash function are the hash function which is suitable to the hardware implementation.

After above signature table configuration, linked word based store-less running search algorithm is performed as string pattern matching mechanism. This algorithm uses the hardware architecture of multi-hash based TBT scheme as shown in the figure 11. When a packet arrives, the first k-byte substring is extracted from the beginning of the payload. For the successive substrings, the k-byte window shifts toward the end of the payload one byte at a time. For each substring, the multiple hash function generates an index, and the corresponding slots in all the hash tables are accessed simultaneously. In figure 11, the simultaneous accesses for the multiple hash tables are represented with dashed-lines. If the key values in the accessed slots match the substring (of the packet) and the addresses of the accessed slots match the values in the previous slot address registers (PSA registers), then the registers are updated with the matched slot addresses. Otherwise, the PSA registers are cleared. Here, the PSA register is cross-referred by each Table Blocks, and PSA register comparison is performed only when the matched slot is not the first substring, called the head substring, in a signature. An attack is detected when the last substring (the tail substring) in a signature is matched.

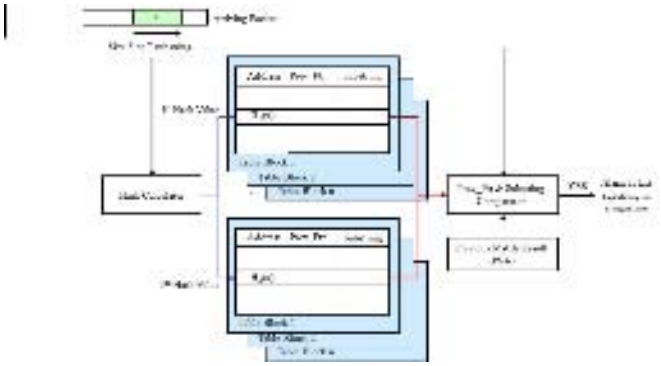

Fig. 11 The hardware architecture of Multi-hash based TBT

Through these operations, our system performs the string pattern matching operation without lowing of performance and packet loss.

V. Simulation based Performance Evaluation

The approach that we presented here is a part of an Intrusion Detection System, called ATPS recently developed. A prototype of multi-hash based TBT was implemented in a XILINX Virtex-II Pro platform FPGA. The FPGA device, XC2VP70, has 74,448 logic cells and 5.9Mbits of an on-chip SRAM, which is a configurable block select memory. There are various ways to configure the block memory in the XC2VP70 FPGA to implement TBT. As shown in the previous work, 512 entries per hash table h with the substring length k of 1~7 bytes showed the best performance in terms of the memory consumption and utilization. However, Implementing 4~7

bytes wide hash tables requires two memory blocks in parallel, since the single block width of XC2VP70 with 512 entries is 4.5 bytes.

Table 1 One-hash based TBT Implementation Result

	Token size (bytes)	Tables	BlockRAMs (18Kbits)
Head/Body Tokens	7	9	18
	5	16	32
Tail Tokens	7	4	8
	6	3	6
	5	3	6
	4	4	8
	3	4	4
	2	3	3
	1	1	1
SUM		47	86

Table 2 Multi-hash based TBT Implementation Result

		Token size (bytes)	Tables	BlockRAMs (18Kbits)
1st Hash	Head/Body Tokens	7	4	8
		5	8	16
	Tail Tokens	7	2	4
		6	1	2
		5	1	2
		4	2	4
		3	2	2
		2	1	1
		1	1	1
2nd Hash	Head/Body Tokens	7	3	6
		5	6	12
	Tail Tokens	7	1	2
		6	1	2
		5	1	2
		4	1	2
		3	1	1
		2	1	1
		1	0	0
SUM			37	68

The current implementation of our prototype is capable of performing only single-content filtering. Therefore, 1660 signatures except signatures with header only and multiple contents are loaded in the prototype system. Here, Table 1 summarizes the amount of the block memory used by the one-hash based TBT implementation. Next, Table 2 shows it by the multi-hash based TBT implementation. As shown in Table 1 and 2, multi-hash based TBT used 68 block memories, which is less than 20% of the previous one-hash based TBT.

VI. Implementation and Experimental Results

We have developed Gigabit IDS based on our architecture, called ATPS. Our system is implemented in programming languages that is best suited for the task it has to perform. Basically, application tasks of our system are implemented in C programming language. FPGA Logic of our system is implemented in verilog HDL (Hardware Description Language) that is best suited for high-speed packet processing. That is, our system has developed in the side of improvement in performance for packet processing. As shown in the figure 12 (a), our system was implemented in a XILINX Virtex-II Pro platform FPGAs. In the above figures, it is marked with the red square. Also, the screen shots were captured during experiments to validate the performance of the prototype. The screen shot (b) shows that web-related attacks were detected. The next screen shot (c) shows that rule-sets for intrusion detection and response were applied.

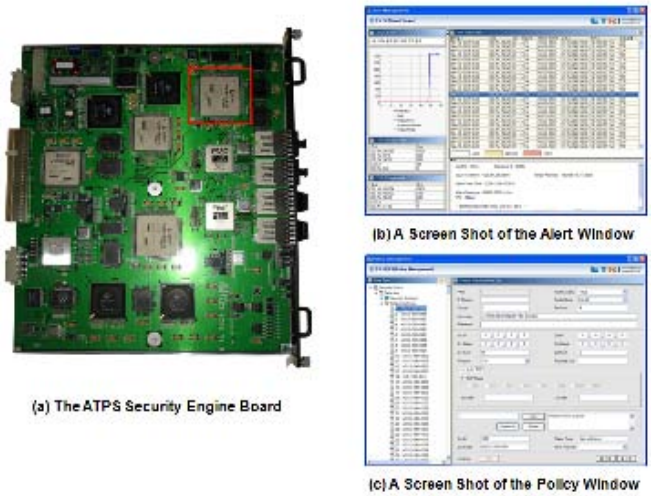

Fig. 12 The Prototype of ATPS

Figure 13 shows our test-bed environment. We applied the snort rule-sets for the whole performance evaluation of our system. And we used IXIA Traffic Generator [25] for background traffic generation, IDS Informer Attack Tool [26], Nessus Vulnerability Scanner [27] for attack traffic generation, and so forth.

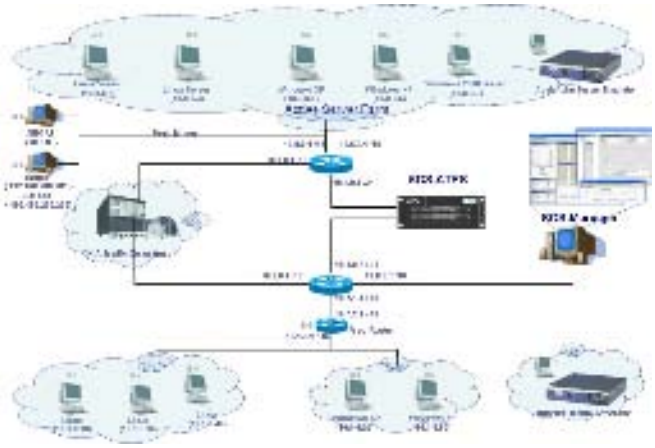

Fig. 13 Test-bed for Experiments

Most of all, detection rate is most important factor in SPI-based intrusion detection system. Generally, both traffic

rate and the number of signatures have an effect on detection rate. For performance evaluation of our prototype system, we observed the rate of alert generation when background traffic generated by IXIA increase gradually. That is, we measured the decrease in effectiveness of the detection when the traffic rate increases. As shown in the figure 14 (a), increasing traffic rate hasn't an effect on detection rate. The second, our experiment was run with a constant traffic rate of 100Mbps and an increasing number of signatures. The experiment starts with only the 200 rule-sets that are needed to achieve maximum detection for the given attacks. As shown in the figure 14 (b), increasing number of signatures also hasn't an effect on detection rate of our system. The previous two experiments using Snort sensors are performed by Kruegel et al. [24]. Compared with Snort sensor, our system showed a consistent performance in traffic level and had nothing to do with increasing number of signatures applied.

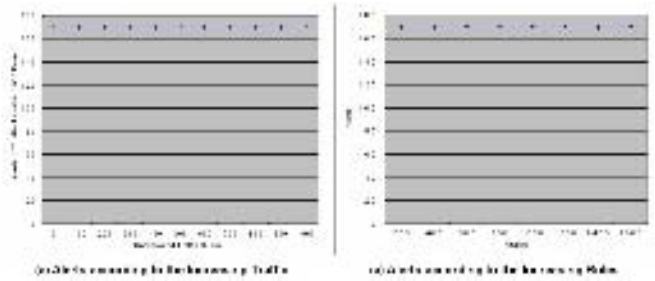

Fig. 14 Performance Evaluation

VII. Conclusion

Providing seamless protection for secure network service is becoming difficult primary because of the increasing link speed and the number of attack patterns, signatures to be maintained. In this paper, we designed the architecture of our system, called ATPS that performs the real-time traffic analysis and intrusion detection on high-speed links, and proposed the novel detection mechanism in FPGA-based reconfiguring hardware that supports more efficient intrusion detection. Also, we have developed the prototype of our system. Most of all, our system focus on reducing a lowering of performance caused by high-speed traffic analysis to the minimum. Therefore, we present an improved content filtering techniques, called multi-hash based TBT. The hashing scheme helps to reduce the data access time, while the multi-hash based TBT technique helps to minimize the memory consumption. Also, it has the advantage that is capable of supporting the effective response by using inline mode monitoring technique on four Gigabit links. However, the current prototype is very preliminary and a thorough evaluation will require experimentation in a real-world environment. In future, for resolving the problem derived from the verification of implemented system, we will go and consider on system performance, availability, faults-tolerance test with prototype. Also, we will keep up our efforts for improvement in performance of detection mechanism on high-speed links. Finally, we will implement and expand our designed system and give more effort to demonstrate effectiveness of our system.

Simulation-based performance evaluation showed that the proposed technique is memory efficient, thereby outperforming the previous one-hash based technique. A prototype of the proposed approach implemented on a FPGA, used the minimum amount of memory for storing 1661 signatures, whereas it can support a 2Gbps link regardless of the number and length of the signatures. Although the prototype can only perform single pattern comparison as of now, we are currently working toward the complete implementation of the TBT architecture to support multi-pattern signatures. Our future work includes optimizing the circuit timing to achieve higher throughput, and extending current TBT to harmonize with the high-speed signature generation approaches that employ packet content filtering.

References

[1] Byoungkoo Kim, Seungyong Yoon, and Jintae Oh, "ATPS – Adaptive Threat Prevention System for High-Performance Intrusion Detection and Response," In Antonio Lagana et al., editors, *Proceedings of the 10th Asia-Pacific Network Operations and Management Symposium(APNOMS 2007)*, volume 4773 of LNCS, pp. 344~353, Sapporo, Japan, Oct., 2007. Springer-Verlag.

[2] Sungwon Yi, Byoung-koo Kim, Jintae Oh, Jongsoo Jang, George Kesidis and Chita R. Das, "Memory-Efficient Content Filtering Hardware for High-Speed Intrusion Detection Systems," *Proceedings of the 2007 ACM Symposium on Applied Computing*, pp. 264-269, Seoul, Korea, 11~15 March, 2007.

[3] Kruegel, C., Valeur, F., Vigna, G. and Kemmerer, R. "Stateful intrusion detection for high-speed networks," In *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 266-274, 2002.

[4] M. Aldwairi, T. Conte, and P. Franzon, "Configurable string matching hardware for speeding up intrusion detection," In *ACM SIGARCH Computer Architecture News*, March 2005, pp. 99-107.

[5] R. Sidhu and V. K. Prasanna, "Fast Regular Expression Matching using FPGAs," In *IEEE Symposium on Field Programmable Custom Computing Machines*, April 2001.

[6] Y. Cho, S. Navab, and W. Mangione-Smith, "Specialized Hardware for Deep Network Packet Filtering," In *Proceedings of International Conference on Field-Programmable Logic and Application*, September 2002.

[7] M. Roesch. "Snort-Lightweight Intrusion Detection for Networks," In *Proceedings of the USENIX LISA '99 Conference*, November, 1999.

[8] Xilinx inc., http;//www.xilinx.com.

[9] Thomas Ptacek and Timothy Newsham, "Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection," Secure Networks Inc., 1998.

[10] R. Boyer and J. Moore, "A Fast String Searching Algorithm," *Communications of the ACM*, vol. 20, no. 10, pp. 762-772, 1977.

[11] A. V. Aho and M. J. Corasick, "Efficient string matching: An aid to bibliographic search," *Communications of the ACM*, vol. 18, no. 6, pp. 333-340, 1975.

[12] S. Wu and U. Manber, "A Fast Algorithm for Multi-Pattern Searching," Technical Report TR-94-17, Department of Computer Science, University of Arizona, 1994.

[13] J. Moscola, J. Lockwood, R. p. Loui, and M. Pachos, "Implementation of a Content-Scanning Module for an Internet Firewall," In *IEEE Symposium on Field-Programmable Custom Computing Machines(FCCM)*, April 2003.

[14] H. Debar, M. Dacier and A. Wespi, "Research Report Towards a Taxonomy of Intrusion Detection Systems," Technical Report RZ 3030, IBM Research Division, Zurich Research Laboratory, Jun., 1998.

[15] S. Kumar and E. Spafford, "A pattern matching model for misuse intrusion detection," In *Proceedings of the 17th National Computer Security Conference*, pp. 11-21, Oct., 1994.

[16] Dong-Ho Kang, Byoung-Koo Kim, and Jin-Tae Oh, "Protocol Anomaly and Pattern Matching based Intrusion Detection System," *WSEAS Transaction on Communications*, Issue 10, Vol. 4, October 2005, pp.994-1001

[17] Slobodan Bojanic, Vladimir Milovanovic, Zorana Bankovic, Carlos Carerras, and Octavio Nieto-Taladriz, "Intrusion Detection Using New FPGA Architecture," *WSEAS Transaction on Communications*, Issue 10, Vol. 4, October 2005, pp.1077-1085
[18] M. Mehde, M. Bensebti, A. Anou, and M. Djebari, "Real Time Solution for Computer Network Intrusion Detection," *WSEAS Transaction on Computers*, Issue 1, Vol. 5, January 2006, pp.216-222
[19] S. Dharmapurikar, P. Krishnamurthy, T. Sproull, and J. Lockwood, "Deep Packet Inspection using Parallel Bloom Filters," in *IEEE Micro*, 2004, vol. 24.
[20] F. Yu, R. H. Katz, and T. V. Lakshman, "Gigabit Rate Packet Pattern-Matching Using TCAM," in *Proceedings of the International Conference on Network Protocols (ICNP)*, 2004.
[21] H. Song and J. W. Lockwood, "Multi-pattern Signature Matching for Hardware Network Intrusion Detection Systems," in *IEEE GLOBECOM*, November 2005.
[22] N. Tuck, T. Sherwood, B. Calder, and G. Varghese, "Deterministic Memory-Efficient String Matching Algorithms for Intrusion Detection," in *Proceedings of the IEEE INFOCOM*, March 2004.
[23] J. R. Crandall, Z. Su, and S. F. Wu, "On deriving unknown vulnerabilities from zero-day polymorphic and metamorphic worm exploits," in *CCS '05: Proceedings of the 12th ACM conference on Computer and Communications Security*, 2005, pp. 235-248.
[24] C. kruegel, F. Valeur, G. Vigna, and R. Kemmerer, "Stateful Intrusion Detection for High-Speed Networks," in *Proceedings of the IEEE Symposium on Research on Security and Privacy*, Oakland, CA, IEEE Press, May 2002.
[25] http://www.ixiacom.com
[26] http://www.bladesoftware.net
[27] http://www.nessus.org

Byoungkoo Kim received the B.S. and M.S. degrees in Information and Communication Engineering from Sungkyunkwan University in 1999 and 2001, respectively. Since 2001, he has stayed in Security Gateway System Team, Electronics and Telecommunications Research Institute (ETRI) of Korea to study Network Security related Topics.

Seungyong Yoon received the B.S. and M.S. degrees in Computer Engineering from Chungnam National University in 1999 and 2001, respectively. Since 2001, he has stayed in Security Gateway System Team, Electronics and Telecommunications Research Institute (ETRI) of Korea to study Network Security related Topics.

Jintae Oh received the B.S and M.S degrees in Electronics Engineering from Kyungpook National University in 1990 and 1992, respectively. He worked at ETRI (Electronics and Telecommunications Research Institute) from 1992 to 1998. During 1998-1999, he stayed in MinMax Tech, USA, as a Research staff. He served as a Director in Engedi Networks, USA, during 1999-2001. He was both Co-founder and CTO Vice President in Winnow Tech. USA during 2001-2003. From 2003, he works with the Security Gateway Team, ETRI, Daejeon, Korea.