# Bit-Reduced Automaton Inspection for Cloud Security

Haiqiang Wang[1] Kuo-Kun Tseng[1*] Shu-Chuan Chu[2] John F. Roddick[2] Dachao Li[1]

[1] Department of Computer Science and Technology, Harbin Institute of Technology, Shenzhen Graduate School
Shenzhen, Guangdong, China
haoyu1987@163.com, kktseng@hitsz.edu.cn, jengshyangpan@gmail.com

[2] School of Computer Science, Engineering and Mathematics, Flinders University
Adelaide, SA 5042, Australia
jan.chu@infoeng.flinders.edu.au, john.roddick@flinders.edu.au

*Received 13 September 2012; Revised 5 July 2013; Accepted 20 July 2013*

**Abstract.** With the development of the cloud computing, its security issues have got more and more attention. There is a great demand for the examining the content of data or packets in order to improve cloud security. In this paper, we propose a new algorithm about pattern matching for cloud security named Bit-Reduced automaton, First it performs inexact matching to filter out the part of nonattack information and then do exact matching to get the final attack information. Finally, Bit-Reduced automaton is feasible through a preliminary evaluation.

**Keywords:** pattern matching; deep packet inspection; cloud security; automaton

## 1 Introduction

For decades, the string matching algorithms have been proven to be useful for deep packet inspection (DPI) of network intrusion intrusions; it scans contents for viruses, and filter them out. However, the algorithms are still some hurdles, including the issues for large volumes of signatures and complex signature. The intrusion detection, virus scanning, content filtering, instant-messenger management, and peer-to-peer identification are all using string matching for inspection. Many works have been done in both algorithm design and hardware implementation to accelerate the inspection, reduce pattern storage space, and efficiently handle complex signature by regular expressions [1].

Especially, with the development of cloud computing, there is a great demand for cloud security to exam content of the data packets in order to improve cloud security and provide the application-specific services. In this paper, we propose a new algorithm for deep packet inspection. For cloud security, we can use this algorithm in the three aspects:

1) For deep packet inspection between the virtual machines of a cloud system.
2) For deep packet inspection of the gateway for cloud system.
3) For deep packet inspection of the cloud storage system.

The new architecture is shown as Figure 1. Firstly, we convert each pattern to generate a new reduced binary string by a reduced function *f*. The string matching module also does the same conversion for the input text. Secondly, a new pattern set does inexact matching with the input text. After inexact matching of the nonattack information is filtered out, the unfiltered information is output to do the exact matching by a suitable algorithm with the original pattern. There are three advantages of new architecture.

1) Because pattern set and input string are all converted by reduced function *f*, they are doing the inexact matching.
2) Because the nonattack information is filtered out, it reduces the time complexity while reduces the information for exact matching in the next step.
3) During the filtering process, unfiltered information is divided into different groups so they can perform the matching by a simple exact matching algorithm which can reduce space and time to a certain degree.

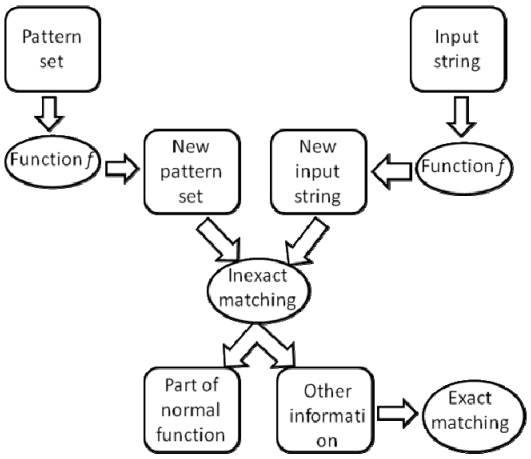


**Fig. 1.** The concept of Bit-Reduced Automaton.

With the 1 and 2 points, we assume that time of the filtering process is $t_f$ and because of information is being filtered out, the saving time of exact matching is $t_e$ . If $t_f$ is far less than $t_e$, there is an improved time efficiency for the overall intrusion detection.

In a summary, the main contributions of this paper are:

1) A new algorithm – Bit-Reduced Automaton in deep packet inspection, which can be very useful for cloud security.
2) A new architecture – which is applied in deep packet inspection of cloud computing, such as between the virtual machines of a cloud system, the gateway for cloud system and the cloud storage system.

The rest of paper is organized as follows. The related work is presented in Section 2. Section 3 proposes the new algorithm - Bit-Reduced Automaton and defines its reduce function. We give a demonstration of Bit-Reduced Automaton for a statistical analysis in Section 4. Finally, the paper is ended with the concluding remarks in Section 5.

## 2 II. Related Work

There are many approaches to achieve data security over the cloud system, such intelligent classifier [24], authentication scheme [25] or biometric recognition [26]. However, this paper focuses on deep data inspection over cloud system. According to it, we first survey the related algorithm of pattern matching as the deep data inspection approaches.

Pattern matching can be an exact or approximate manner. An exact matching algorithm assumes that the pattern and the matched text should be the same exactly. While an approximate matching algorithm allows some error between the pattern and the matched text.

Conventionally, the exact string matching algorithms can be categorized as in the various ways. According to the survey literature [2] and the other research papers, We categorize the algorithms into three types. These are automaton-based, heuristics-based and filtering-based algorithms.

### 2.1 Automaton-based algorithms

The famous Aho-Corasick (AC) algorithm [3] was proposed for multi-pattern matching. Norton [4] presents the Optimized-AC algorithm to reduce the memory requirement in the well-known IDS package which is named Snort. It compresses the transition table into a compressed sparse vector format or a banded-row format. Tuck et al. [5] uses both bitmap and path compression to compress the transition table. Tan and Sherwood [6] literature splits the finite automaton in the AC algorithm into the several bit level. Becchi and Crowley [7] proposed a hybrid automaton which addresses this issue by combining the benefits of the deterministic and non-deterministic finite automata. In paper [8], Kumar et al. proposed a Delayed Input DFAs ( $D^2$ FAs), which provides a trade-off between the memory requirements of the compressed DFA and the number of visited states for each processed character. [9] proposes a compression technique for DFAs which ensures at most 2N state tra-

versals when processing a string of length *N*. [10] performs evaluation of the workload for the regular expression matching architectures.

### 2.2 Heuristics-based algorithms

The Boyer-Moore (BM) algorithm is a single string matching algorithm [11]. Horspool simplified the BM algorithm [12] by resorting to one single bad-character, which is useful for a reasonable large alphabet, such as the ASCII table. Because the probability of a character mismatch with the characters of the pattern within the search window is high. The set-wise Horspool algorithm [13] extends the Horspool algorithm to search for multiple patterns simultaneously. The Wu-Manber (WM) algorithm [14] improves the set-wise Horspool algorithm by reading a block of *B* characters rather than only one character for the shift value. The Shift-Or (SOR) algorithm [15] and the Backward Nondeterministic DAWG Matching (BNDM) algorithm [16] were proposed for single string matching with bit-parallelism to simulate the tracking of a non-deterministic finite automaton (NFA).

### 2.3 Heuristics-based algorithms

The Rabin-Karp (RK) algorithm [17] is designed to handle single string matching. Muth and Manber [18] use two-level hashing to accelerate the binary search. Taskin Kocak and Ilhan Kaya [19] propose a new Bloom filter architecture that exploits the well-known pipelining technique, which leads to the energy-efficient implementation of intrusion detection systems. Ozgun Erdogan and Pei Cao [20] propose Hash-AV filtering with a set of hash functions sequentially.

It can be observed from the above researches as shown in Figure 2. In recent years, more and more researches are focused on automaton for deep packet inspection. Because of deterministic automatons' high-speed and flexible for in pattern length. Therefore, we design a specific algorithm which is called Bit-Reduced Automaton. Its main difference with the previous automaton will be described in a more detail as in the next section.

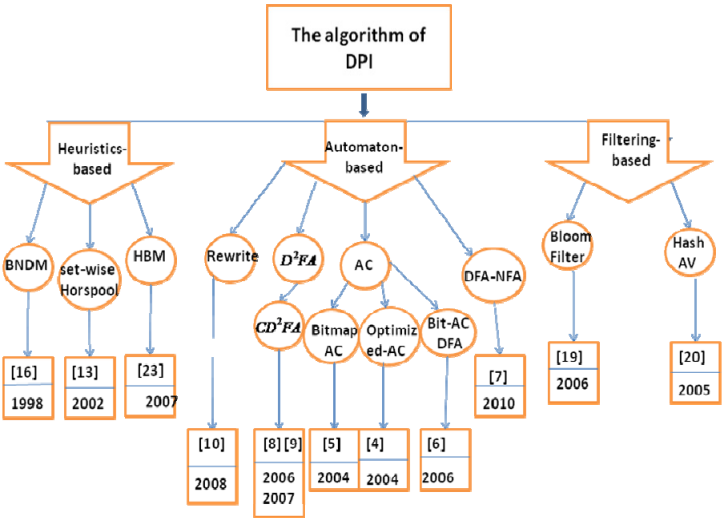


**Fig. 2.** The algorithms of deep packet inspection

## III. Proposed Bit-Reduced Automaton

The proposed architecture of the Bit-Reduced Automaton is shown as Figure 3. We divide the architecture into two modules. The module 1 is a preprocessing module, it in charge of building Bit-Reduced Automaton with the patterns, and module 2 is a matching unit, it performs the pattern matching for the input string of network stream.

From the architecture of Bit-Reduced Automaton (DFA), we know that the reduce function plays a key role in fast inexact matching. They may be many appropriate reduce functions to achieve a good performance in the fast inexact matching. In the next section, we give an example of the reduce function.

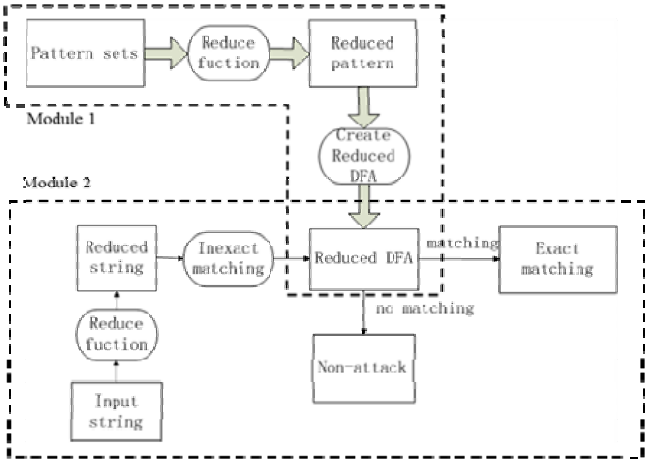


**Fig. 3.** The architecture of Bit-Reduced Automaton.

### 3.1 Reduced Function

In this section, we give a simple reduce function. The main idea of this reduced function is that the byte being converted to bit and vice versa. The detailed process is that. For instance, the reduce function convert single byte into a single bit. Then 8 bits are grouped into one byte for the automaton matching. So the reduced function plays a role for compressing patterns and providing inexact matching. In reduced function, the reduced degree can be the value *1, 2* and *4*.

If we define pattern set as $P\{P_1, P_2, P_3, ...P_{n-1}, P_n\}$. $P_i(i = 1,2,...n)$ represents a pattern. $P_{i1}P_{i2}P_{i3}...P_{im}$ represents *m* bytes of $P_i$. And $b_{i1}b_{i2}b_{i3}...b_{im}$ means a bit sequence that is converted by each $P_i$'s byte by using reduce function. The rule from byte to bit is defined as following.

$$\begin{cases} P_{ij} \geq P_{ij+1}, & b_{i1}=1 \\ P_{ij} < P_{ij+1}, & b_{i1}=0 \end{cases} \quad (1)$$

In the above formula $j = 1, 2, \ldots, m\text{-}1$.

By (1) we can get a bit sequence $b_{i1}b_{i2}b_{i3}...b_{im}$. According to reduce function, then the bit sequence is need to be converted into a byte sequence again. The rule from bit to byte is as following:

From the first bit of $b_{i1}b_{i2}b_{i3}...b_{im}$, for example if we use 8-bit as one group, it convert above 8 bits sequence into one byte again. However, if the last remaining bits are less than 8, we drop them.

Now we will give an example of reduce function shown as Figure 4. In this example, the byte sequence *shbea23cdd27dkgl9nv* is converted into new byte sequence by reduce function. The new byte sequence is *0xD8 0xC5* which is represented by hexadecimal. Comparing the original byte sequence with the new byte sequence, we found that the length of new byte sequence is much shorter than the original byte sequence.

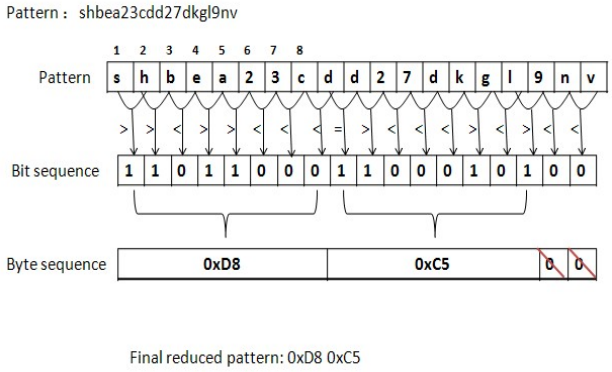


**Fig. 4.** An example of reduce function.

### 3.2 The definition of Bit-Reduced Automaton

Bit-Reduced Automaton is a 5-tuple $(Q, \Sigma, \delta, q_0, A(\alpha, \beta))$ [22, 23], consisting of

- $Q$: a finite set of states.
- $\Sigma$: a finite set of input symbols called the alphabet.
- $\delta(Q \times \Sigma \rightarrow Q)$: a transition function.
- $q_0(q_0 \in Q)$: a start state.
- $A(\alpha, \beta)$: a set of accept states.

where $\alpha$ and $\beta$ are two parameters. $\alpha$ represents the depth that is the number of edges from root to an accept state. $\beta$ represents the maximum length of the output pattern in an accept state.

Bit-Reduced Automaton and formal automaton are similar. The only one difference between them is that the accept state of Bit-Reduced Automaton has two parameters $\alpha$ and $\beta$. We will introduce these two parameters during the demonstration of this algorithm as in the next section.

In this paper, we use AC to build a Bit-Reduced Automaton. AC algorithm is a good multi-pattern matching algorithm. It is widely used in Deep Packet Inspection nowadays.

### 3.3 Demonstration of Bit-Reduced Automaton

Figure 5 is a demonstration of Bit-Reduced Automaton algorithm. The input string from the first character is converted to new input string by reduce function. Converted input string does matching in Bit-Reduced Automaton starting state 0 and then it matches in states 1, 2 and 3. However, when from state 3 to state 4, it fails to match. According to fail function of AC, the next matching state is state 6. In figure 4, P is the pointer pointing the character which is being processed now in the original input string. Because state 6 is a accept state in Bit-Reduced Automaton, there are matching string in original input string. The algorithm must do an exact matching at this time. However, if all input string should do exact matching?

We know that each accept state has two parameters $\alpha$ and $\beta$. In figure 4, the two parameters' values of accept state 6 are 2 and 22. Because of $\alpha$, the depth of accept state 6, is equal to 2, there are only 16 (multiplied by 2 and 8) characters before the pointer $P$. And because $\beta$, the maximum length of accept 6, is equal to 22, there are only 6 (subtraction of 22 and 16) characters after pointer P including the character which $P$ points now. Therefore, only 22 characters in input sting from $r$ to $h$ expressed in the braces in figure 4 do exact matching in accept state 6.

Because the number of pattern contained in each accept-state is different, when doing exact matching, we can choose appropriate algorithm depending on the circumstance. In the evaluation part, we will introduce how to choose an appropriate algorithm.

Here is some basic steps to illustrate our proposed architecture using Figure 5 as an example, which assumed the Reduced pattern and Reduced DFA are already constructed:

1) The input string is converted to new input string by reduce function.
2) Converted input string does matching in Bit-Reduced Automaton starting from state 0 as shown in figure 5.
3) During the inexact matching process, we compared the final state with the reduced pattern sets.
4) Finally, if the result is matching then we do exact matching; otherwise, we consider it as nonattack.

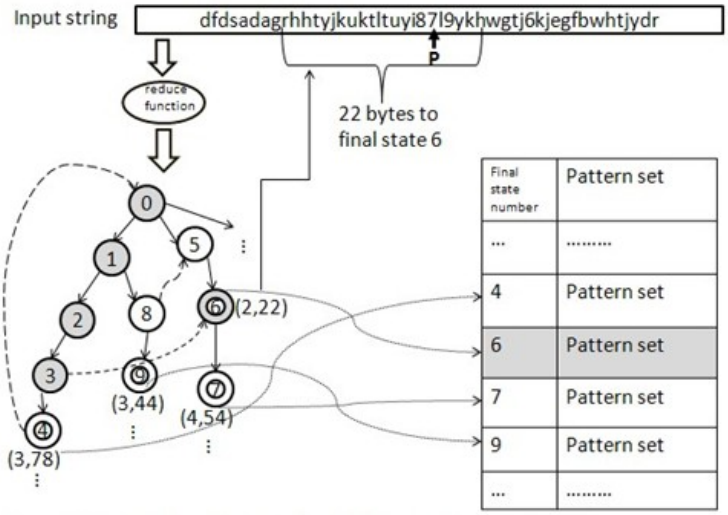


**Fig. 5.** Demonstration of Bit-Reduced Automaton.

## IV. Evaluation

To evaluate this algorithm, we randomly choose 4932 non-repetitive patterns from all patterns of snort version 2.9 [21]. All these patterns must meet a condition, that the lengths of their bit sequences converted by reduced function which is longer than 8 bits. As we use these patterns to build Bit-Reduced Automaton according to AC algorithm, we have the related statistical result as Figure 6.

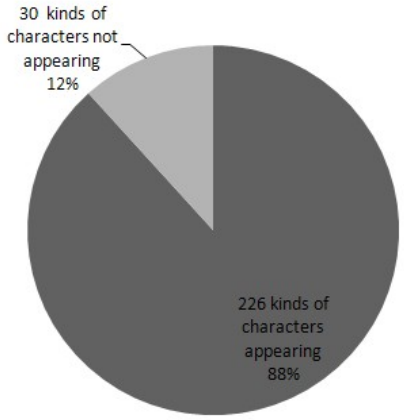


**Fig. 6.** The statistics of edges information from the root to one-depth states of Bit-Reduced Automaton.

In Figure 6, it appears 226 kinds of characters of all edges' characters from root to one-depth states in the Bit-Reduced Automaton. One byte can represent the type of 256 characters, so it has no 30 type of characters appearing from the root to one-depth states. Therefore, when input string does inexact matching in Bit-Reduced Automaton from the root to the one-depth state, the byte converted by input string is filtered which belongs one of these 30 type of characters. Thus in theory, the percentage of filtered is at least 12%.

We know that the smaller the depth value of each accept-state and the maximum length of pattern, the shorter the string gets from input string, the less time consuming in the exact pattern matching. In Figure 7, the depth value of most accept-states is less than 3. In Figure 8, the maximum length of pattern in most accept-states is less than 30. From these two statistics, these patterns we choose from snort version 2.9 is satisfied by this condition. Therefore, this new algorithm and architecture designs should be reasonable.

In the experimental results, we can know how to choose an appropriate algorithm. Our Bit-Reduced Automaton is an inexact matching algorithm, which has a significant contribution, that it is still effective when input string or signatures are changed. Especially with the development of cloud computing, the data transferred between cloud and client may be changed very quickly, so our algorithm and architecture can be very useful for identifying content of the data packet. This is the fundamental part for improving cloud security and providing the application-specific services.

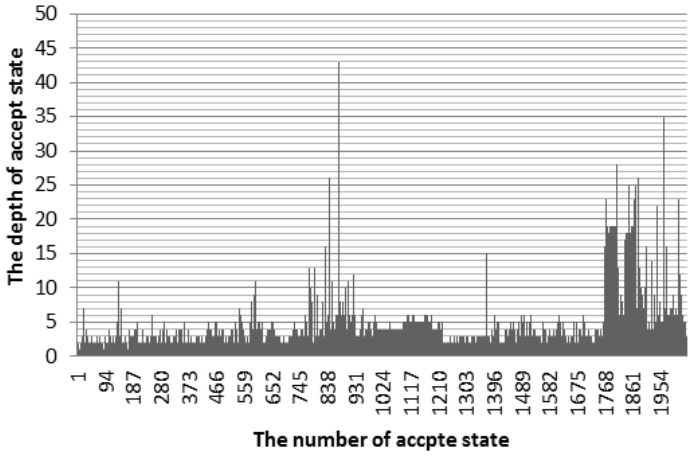


**Fig. 7.** The statistics of each accept state depth ( parameter $\alpha$ ).

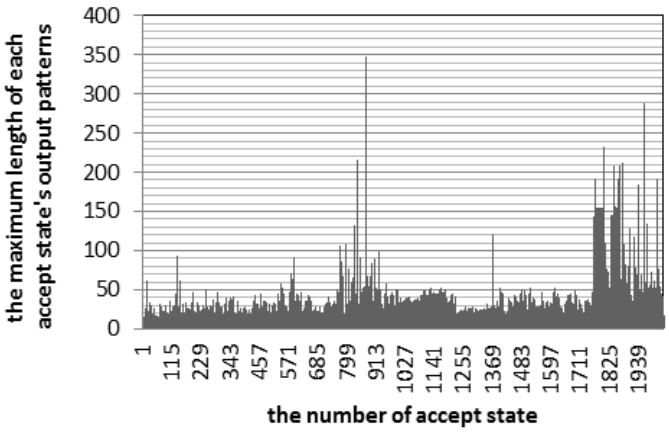


**Fig. 8.** In each accept state, the maximum pattern length of its patterns (parameter $\beta$).

## V. Conclusion

This paper presents a new architecture and based on it we proposed a specific Bit-Reduced Automaton. In addition, we defined a reduced function for the Bit-Reduced Automaton, and redefined 5-tuple DFA with adding two parameters, α and β for the accept state. For evaluation, we use snort rules to verify the feasibility of the new algorithm. The result shows the algorithm can achieve at least 12% of the information filtering in theory. According to the parameters α and β statistics and the number of pattern each acceptance state. They also show that this new algorithm design is reasonable.

## Acknowledgement

The authors would like to express their acknowledgement for the financial support from the cloud verification platform project of National Development and Reform Commission (NDRC), China.

## References

[1] P.-C. Lin, Y.-D. Lin, Y.-C. Lai, T.-H. Lee, "Using String Matching for Deep Packet Inspection," *Computer*, Vol. 41, No. 4, pp. 23-28, 2008.

[2] P.-C. Lin, Z.-X Li, Y.-D. Lin, Y.-C. Lai, F. C. Lin, "Profiling and Accelerating String Matching Algorithms in Three Network Content Security Applications," *IEEE Communications Surveys & Tutorials*, Vol. 8, No. 2, pp. 24-36, 2006.

[3] A. Aho and M. Corasick, "Efficient String Matching: An Aid to Bibliographic Search," *Communications of the ACM*, Vol. 18, No. 6, pp. 333-40, 1975.

[4] M. Norton, "Optimizing Pattern Matching for Intrusion Detection," Available: http://www.snort.org/docs/

[5] N. Tuck, T. Sherwood, B. Calder, G. Varghese, "Deterministic Memory–Efficient String Matching Algorithms for Intrusion Detection," in *Proceedings of IEEE INFOCOM 2004*, IEEE Press, Vol. 4, pp. 2628-2639, 2004.

[6] L. Tan and T. Sherwood, "A High Throughput String Matching Architecture for Intrusion Detection and Prevention," in *Proceedings of 32nd International Symposium on Computer Architecture (ISCA 2005)*, pp. 112-122, 2005.

[7] M. Becchi and P. Crowley, "A Hybrid Finite Automaton for Practical Deep Packet Inspection," in *Proceedings of ACM CoNEXT 2007*, ACM Press, pp. 1-12, 2007.

[8] S. Kumar, S. Dharmapurikar, F. Yu, P. Crowley, J. Turner, "Algorithms to Accelerate Multiple Regular Expressions Matching for Deep Packet Inspection," in *Proceedings of ACM SIGCOMM'06*, ACM Press, pp. 339-350, 2006.

[9] M. Becchi and P. Crowley, "An Improved Algorithm to Accelerate Regular Expression Evaluation," in *Proceedings of ANCS* 2007, pp. 145-154, ACM Press, 2007.

[10] M. Becchi, M. Franklin, P. Crowley, "A Workload for Evaluating Deep Packet Inspection Architectures," in *Proceedings of IISWC 2008*, IEEE Press, pp. 79-89, 2008.

[11] R. Boyer and S. Moore, "A Fast String Searching Algorithm," *Communications of the ACM*, Vol. 20, No. 10, pp. 762-772, 1977.

[12] N. Horspool, "Practical Fast Searching In Strings," *Software – Practice and Experience*, Vol. 10, pp. 501–06, 1980.

[13] G. Navarro and M. Raffinot, *Flexible Pattern Matching in Strings*, Cambridge University Press, 2002.

[14] S. Wu and U. Manber, *A Fast Algorithm For Multi–Pattern Searching*, Report TR–94–17, Department of Computer Science, University of Arizona, 1994.

[15] R. Baeza-Yates and G. Gonnet, "A New Approach to Text Searching," *Communications of the ACM*, Vol. 35, pp. 74-82, 1992.

[16] G. Navarro and M. Raffinot, "A Bit-Parallel Approach to Suffix Automata: Fast Extended String Matching," in *Proceedings of 9th Annual Symposium of Combinatorial Pattern Matching (CPM 1998)*, Volume 1448 of Lecture Notes in Computer Science, pp. 14-33, 1998.

[17] R. Karp and M. Rabin, "Efficient Randomized Pattern-Matching Algorithms," *IBM Journal of Research and Development*, Vol. 31, No. 2, pp. 249-60, 1987.

[18] R. Muth and U. Manber, "Approximate Multiple String Search," in *Proceedings of 7th Annual Symposium of Combinatorial Pattern Matching (CPM 1996)*, Volume 1075 of Lecture Notes in Computer Science, pp. 75–86, 1996.

[19] T. Kocak and I. Kaya, "Low-power Bloom Filter Architecture for Deep Packet Inspection," *IEEE/Communications Letters*, Vol. 10, No. 3, pp. 210-212, 2006.

[20] O. Erdogan and P. Cao, "Hash-AV: Fast Virus Signature Scanning by Cache-resident Filters," *International Journal of Security and Networks*, Vol. 2, No. 1/2, pp. 50-59, 2007.

[21] SNORT. Network Intrusion Detection System. Available at: http://www.snort.org/.

[22] DFA. Deterministic Finite Automaton. Available at: http://en.wikipedia.org/wiki/Deterministic_finite_automaton.

[23] J. Ni, C. Lin, Z. Chen, P. Ungsunan, "A Fast Multi-pattern Matching Algorithm for Deep Packet Inspection on a Network Processor," in *Proceedings of International Conference on Parallel Processing*, IEEE Press, 2007.

[24] I. Bouzouita and S. Elloumi, "Positive and Negative Generic Classification Rules-Based Classifier," *International Journal of Knowledge and Learning*, Vol. 7, No. 3/4, pp. 271-293, 2011.

[25] T.-Y. Wu, T.-T. Tsai, Y.-M. Tseng, "A Revocable ID-based Signcryption Scheme," *Journal of Information Hiding and Multimedia Signal Processing*, Vol. 3, No. 3, pp. 240-251, 2012.

[26] Q. Feng, J.-S. Pan, L. Yan, "Restricted Nearest Feature Line with Ellipse for Face Recognition," *Journal of Information Hiding and Multimedia Signal Processing*, Vol. 3, No. 3, pp. 297-305, 2012.