IOSR Journal of Electronics and Communication Engineering (IOSR-JECE)
e-ISSN: 2278-2834,p- ISSN: 2278-8735
PP 54-61
www.iosrjournals.org

FPGA Implementation of Token-Based Clam AV Regex Virus Signatures with Early Detection

Rajalakshmi.R[1], Anbarasan.A[2]
[1]*PG Scholar, Department of ECE, Surya Group of Institutions; mailrajilakshmi@gmail.com*
[2]*Asst. Prof, Department of ECE, Surya Group of Institutions; apporvan@gmail.com*

Abstract: *This paper describes hardware architecture for high speed regex pattern matching with ClamAV virus database. The proposed method is incorporated with a new technique namely early detection in matching the regex patterns. With the information reduction approach, instead of viewing the pattern matching as byte sequences, the input byte stream is transformed into a token stream. The processing architecture will then process the token stream with bit-split string matching method to determine if any virus signatures can be found. Furthermore, this paper proposes a novel reconfigurable clock divider model to avoid synchronization problems during page-enabled parallel processing. The detection engine is memory-based and therefore the FPGA need not be reconfigured to support dynamic updates. The proposed approach can provide worst-case performance guarantee also for variable rate pattern match and efficient in terms of power with gated latch control circuit.*

Index Terms: *non-deterministic finite automaton, parallel processing, network intrusion detection, gated latch, memory-based.*

I. INTRODUCTION

Every system connected to the Internet is susceptible to different kinds of attack such as virus and worm inflections, junk mail (spam messages) and email spoofing. Therefore there exists an increasing demand for network devices capable of inspecting the content of data packets in order to enhance network security and provide application-specific services. Firewalls were used considerably to prevent access to systems from intrusions but they cannot get rid of all security threats, nor can they identify attacks when they occur. Hence, next generation firewalls should provide deep packet inspection [3] capabilities, in order to provide prevention from these attacks. Network Intrusion Detection Systems (NIDS) performs the function of deep packet inspection. Matching engines inspect packet’s payload searching for patterns that would alert security threats. Matching every incoming byte against thousands of pattern characters at wire rates such as for 3G, 4G high mobility communications is a complicated task. Most of the intrusion detection systems that accomplish deep packet inspection perform simple string matching algorithms to match packets against a large, but finite set of strings. Therefore, modern matching engines uses regular expression-based pattern matching, since regular expressions present higher expressive power and flexibility. Performance analysis of SNORT [2] open source IDS rule-set reveals that 31% of total processing time is due to string matching and it may goes up to 80% in case of Web-intensive traffic. Therefore, string matching of patterns is the most computationally intensive part of anti-virus system. So in this paper we mainly concentrate on optimized matching algorithms.

FPGA-based detection systems provide higher flexibility and high throughput comparable to systems based on ASIC platform. FPGA-based systems can exploit parallelism to achieve acceptable processing throughput. Several implementations for FPGA-based intrusion detection using regular expressions (NFAs/DFAs), ternary CAM and filtering techniques (Bloom and Bloomier filters) for deep packet inspection have been proposed.

The rest of the paper is laid out as follows. In section 2, we shall summarize the related work. In section 3, we shall present the novel architectures namely early detection method and gated latch control circuitry. In section 4, we explain the token detection methods and the aggregation unit. In section 5, we shall present a new methodology namely page-enabled parallel processing (PEPP) to raise the throughput rate of the pattern matching engine. Performance analysis of the revised architecture will be presented in section 6. Section 7 is the conclusion.

II. RELATED WORK

Implementation of NIDS signatures as deterministic finite-state automata (DFAs) may leads to state explosion problem. Although the overall memory has been reduced by Yu and Chen [3] with the proposal of multiple DFAs, state-space explosion arises for complex signature sets. Another limitation is the increased scanning time because multiple DFAs should inspect payloads now.

State explosion problems has not been resolved completely even after the implementations of DFA compression techniques based on transition rule reduction [6] and data encoding [7]. For the HFA method [8], hardware implementation details have not been revealed clearly. Bloom filters techniques proposed by Ho and Lemieux [9] pose upper bound limit on the length of string segments to be handled.

The major limitation of hardwired circuit approach [10] to exploit parallelism is that they require FPGA to be reconfigured for every database update. State explosion problem has not been addressed in the approach presented by Lee [4] to detect virus signatures with regex features based on generalization of Aho-Corasick string matching algorithm.

In this paper we present a memory-efficient parallel string matching scheme using the pattern dividing approach and its hardware architecture for identification of patterns. Initially the byte stream is transformed into token stream by dividing the long target patterns into sub-patterns with a fixed length and then processed with bit-based comparisons. In this approach, there is an increased number of shared common states due to reduced length and this approach is very efficient when compared with the cases of the string matching with byte-based comparisons.

Here, we present the results of our implementation with open-source IDS software ClamAV [1]. ClamAV virus database consists of basic, regular expression (or regex) and MD5 types. A basic signature is represented as a continuous byte string and a regex signature is an extension of the basic pattern with various wildcards. The amount of processing speed achieved with the optimizations of string matching algorithm and the improvement in throughput has been analyzed. The methodology that we developed here is efficient in terms of throughput and worst-case performance. The proposed method speeds up the processing rate of the string searching by three times.

III. PROPOSED EARLY DETECTION METHOD

To build many tiny state machines is the key to achieving high performance so that each of the state machines searches for only a portion of the bits of each rule. The new methodology that we proposed is specifically directed toward implementation in an architecture built up as an array of small memory blocks. Therefore, a system that maintains tight worst-case bounds on performance can be easily updated without interrupting the matching function. So the proposed method is considered to be efficient than existing best known approaches.

3.1 Bit-split Scheme

The hardware architecture is configured bit-split scheme that partitions and bit-splits a finite-state machine (FSM) into number of small state machines. And finally the prototype of the proposed bit-split matching engine is presented with Altera Kit to analyze the hardware modules.

Here the state machines have been partitioned into a set of new state machines, thus each of the state machines matches only some of the bits of the input stream. Furthermore, each new state machine is only passed when a given input data could be a match. Therefore a match will be announced only when all of the match vectors agree which is implemented using partial match vector module.

The partial match vector is a bit-vector which denotes the match for each rule. A full match vector can be declared by taking the AND of each of the partial match vectors, which accounts for the true match for a particular rule.

TABLE I Binary Encoding of Input Stream "**client**"

char	7	6	5	4	3	2	1	0
c	0	1	1	0	0	0	1	1
l	0	1	1	0	1	1	0	0
i	0	1	1	0	1	0	0	1
e	0	1	1	0	0	1	0	1
n	0	1	1	0	1	1	1	0
t	0	1	1	1	0	1	0	0

Each bit of the eight-bit ASCII code is extracted to construct its own binary state machine, a state machine whose alphabet contains only 0 and 1. Based on the ASCII code, bit-based matching is first made at the state machines of lowest edges. For example for the input stream "client", the ASCII codes are "63, 6C, 69, 65, 6E 74" and the binary digits are extracted as given in Table I.

Matching of consecutive state machines can be triggered only if the system receives the enable signal from the lowest edge state machine. Thus considerable amount of processing time can be efficiently reduced from the fact the probability of occurrences of intrusions in the input stream will be considerably low. Such that most of the matching problems can be skipped off by the architecture and therefore increased throughput rate can be achieved with our early detection approach.

3.2 Gated latch control circuit

Since the matching of virus signatures presented as sub-patterns, for example with the pattern "client" sub-pattern 1 is "cli" and "ent" as sub-pattern 2. The result of partial match vector of sub-pattern1 can be used as the next state determiner for the second partial match vector to start the matching process for the sub-pattern 2 as shown in Fig. 1. Matching process for the patterns triggered by early detection module can be skipped through Gated latch control circuit for low power consumption.

Therefore, skipping of the matching process with the proposed control circuitry can greatly contributes to the increase in processing speed of the hardware architecture in addition to efficient power saving by keeping the hardware resources idle for those which skipped by the early detection methodology. The proposed scheme can be easily implemented for both basic and regex signatures.

		F_7	F_6	F_5	F_4	F_3	F_2	F_1	F_0	
c	**63**	**0**	**1**	**1**	**0**	**0**	**0**	**1**	**1**	
l	**6C**	**0**	**1**	**1**	**0**	**1**	**1**	**0**	**0**	Sub-pattern 1
i	**69**	**0**	**1**	**1**	**0**	**1**	**0**	**0**	**1**	
e	**65**	**0**	**1**	**1**	**0**	**0**	**1**	**0**	**1**	
n	**6E**	**0**	**1**	**1**	**0**	**1**	**1**	**1**	**0**	Sub-pattern 2
t	**74**	**0**	**1**	**1**	**1**	**0**	**1**	**0**	**0**	

Fig. 1. Early Detection module

IV. TOKEN DETECTION METHOD

The increase in number of counting blocks poses difficulties to the design of the hardware matching engine. The hardware implementation using ad hoc approach is not feasible to incorporate large count of hardware circuits to execute the required counting. Also the hardware implementation using DFA suffers from state explosion problem in case of counting blocks get expanded into explicit states.

Bit-split string detection method can be used to detect the string tokens. PACX method (Extended P-AC method) has been used to match fixed-length tokens with 4 to 15 bytes which consists of a basic string component at the front combined with a small number of wildcard bytes, nibbles and/or alternate bytes. A regex detection method called MX-NFA is used to detect more complex tokens which may consists of counting blocks and other regex features. The block diagram of signature matching engine is given in Fig. 2.

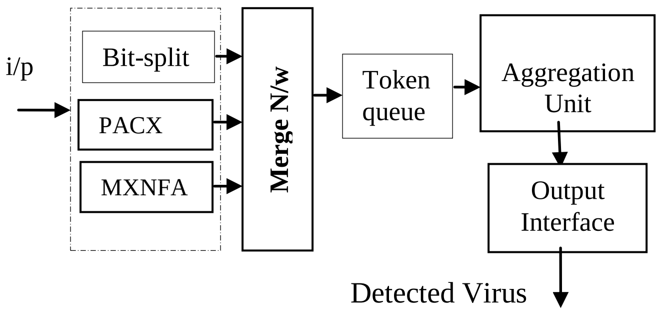


Fig. 2. Block diagram of the signature matching engine

4.1 PACX token detection unit

The PACX (extended P-AC string detection method) detection method differs from the P-AC method in the fact that PACX method does not contain any feedback path. Furthermore PACX method can support wildcard byte, nibble and alternate byte values.

The PACX detection method can process tokens with 4 to 7 bytes as given in Fig. 3 which can execute matching of tokens with 4 bytes. The lookup table and a processing unit are the important components of PACX method. There are about 6 fields to account for an entry in the lookup table which are *char* (character), *cm* (character mask), *ns* (next state), *bs* (bit-select mask), *tid* (token ID) and control flags. The comparator does not required for LT_0 as it can be directly indexed by the input byte.

The detection method can be illustrated with the four example set of tokens. The character-trie is formed for the given set of tokens and consists of transition symbol (number next to an edge) and base address (number inside a state) as shown in Fig. 4. Detailed explanation for the set-up of the lookup tables of PACX method for the example set of tokens is given in [5].

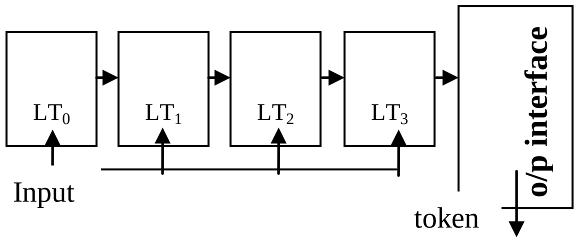


Fig. 3. PACX detection method

Token	Description
t1	28223266
t2	2822(33|34|35|36|37|38|39)66
t3	7638e?e9
t4	83??6a68

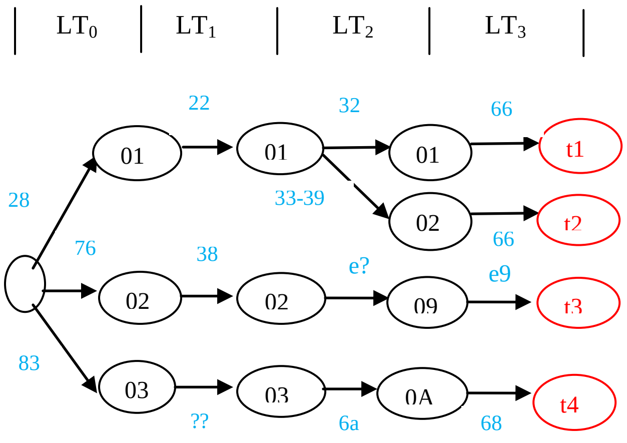


Fig. 4. Sample set of tokens and the character-trie.

4.2 MX-NFA token detection unit

The MX-NFA regex detection method can be used to process the tokens that are complex to be executed by any other detection methods. The implementation of this method is based on NFA. The active rule table can be used to store and keep track of the transition rules of the underlying NFA. In the table, each rule can be enabled or disabled automatically by the control logic circuits.

Control flags can be used to represent the corresponding behavior of a transition rule. For example a token aabb{-2}ccdd{-10}eeff is considered and MX-NFA rule table can be constructed as given in Fig. 5. The wildcard entries (*e*3 and *e*4) can be used to support the at-most count {-2}. Using activation rule *e2,* the sequential rules e3, e4 and e5 can be fired. The counter module can be used to handle the at-most count {-10} in MX-NFA regex detection method.

The *E*-bit of *e*1 which is set to 1 after system reset can remain active with H=1. The at-most count can be checked by the count module which consists of an initial count register and a count-down counter. The counter starts the count-down if the count signal *AC* is passed to the count module. The count-event signal *CE* will be generated to stop the counting when the counter reaches to 0. Detailed explanation for the implementation of MX-NFA rule table can be seen from [5].

Rule	**Symbol**	**Control flags**					
		I	**H**	**O**	**S_2S_1**	**R**	**C**
e1	aa	1	1	0	00	0	0
e2	bb	0	0	0	10	0	0
e3	??	0	0	0	00	0	0
e4	??	0	0	0	00	0	0
e5	cc	0	0	0	00	0	0
e6	dd	0	0	0	00	0	1
e7	ee	0	1	0	00	1	0
e8	ff	0	0	1	00	0	0
e9	null	0	0	0	00	0	0

Fig. 5.Setup of the MX-NFA rule table for the token aabb{-2}ccdd{-8}eeff

4.3 Token refinements and Aggregation Unit

The detected tokens are reassembled by the aggregation unit to check for any multi-token segment. Hence the lower-bound displacement can be checked by the aggregation unit despite of the token detection unit. There may be the possibility for the occurrence of two or more tokens with only variation in at-most count. For example in case of token 5b4424**{-20}**33d2 and the token 5b4424**{-30}**33d2, MX-NFA detection method takes account of the token with the larger at-most count only, i.e. token 5b4424**{-30}**33d2.

The AU can be implemented as a conventional NFA. *Current state list* (CSL) and *next state list* (NSL) are the two lists of states which can be maintained as FIFO queues. The aggregation unit can be used to perform the approximated checking instead of exact checking of displacement counts in the case of multiple at-most counts.

V. IMPLEMENTATION OF PAGE-ENABLED PARALLEL PROCESSING (PEPP)

To catch up with the throughput requirement, we need to devise new system architecture that can process multiple bytes of input data per cycle. With the presented parallel approach for hardware implementation of string matching, system throughput can be doubled with simple logic structure. By allowing **page enabled parallel processing (PEPP)** as presented in Fig. 6, throughput rate is highly increased. In Asynchronous circuits, handshaking protocol is used for time reference. In synchronous circuits clock will be used for time reference. The main limitation of this approach is the occurrences of synchronization errors in synchronous circuits.

There are three cases of synchronization mismatches called mismatch during ASCII encoding, mismatch occurs during string matching conversion and mismatches may occurs between the pages. This limitation can be avoided using PLL (Phase locked loop) clock technique. Here highly reconfigurable clock divider will be used along with reconfigurable delay lines based phase match schemes. By using this model single source clock can be used for multi rate clock domains.

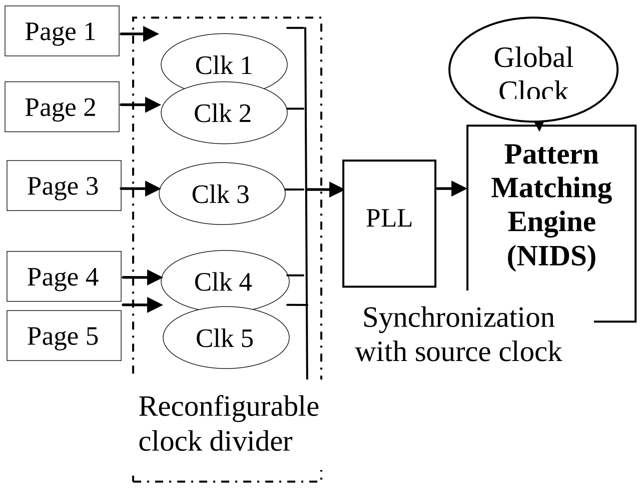

Page 1
Page 2
Page 3
Page 4
Page 5
Clk 1
Clk 2
Clk 3
Clk 4
Clk 5
Reconfigurable clock divider
PLL
Global Clock
Pattern Matching Engine (NIDS)
Synchronization with source clock

Fig. 6. Page-enabled parallel processing (PEPP)

Here in this study, five pages with various operating clocks are implemented based on single source clock by using RECONFIGURABLE CLOCK DIVIDER. And phase signals will be successfully matched with PLL.

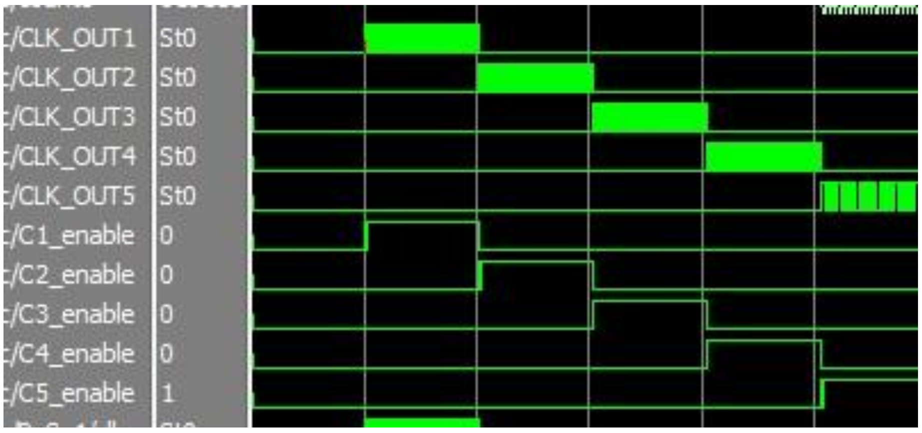
To cope up with the variable rate browsing of the users from different geographical locations such as 2G,3G or 4G speeds, the source clock will be divided into required range. And this divided clock will be matched with global clock using PLL. For every positive clock of match PAGE data will be read out and match with input stream.

The main advantage of this method is that this implementation can support the previously proposed early detection method. During matching time early detection will be carried out from any PAGES.

VI. EXPERIMENTAL RESULT AND EVALUATIONS

The simulation results for the page-enabled parallel implementation given with ModelSim-Altera 6.4a showed the efficiency of the approach. Here the input intrusions are partitioned into pages. Adoptive reconfigurable phase locked loop based clock divider has been used for variable rate pattern match. Bit-wise FSM based state transitions are performed between input bit stream & sub pattern from each pages. Final global match has been based on PMV vector.

SIMULATION OUTPUT WITH CLOCK MISMATCH



The partitioning of the source clock with the reconfigurable clock divider contributes to the improvement of clock speed in the new design. Also the hardware matching engine only uses a few percents of the LUTs available in the FPGA board.

If the method is proposed on ASIC, more chip area can be allocated to memory blocks and the matching engine can achieve even higher clock speed. From the evaluation results it is clear that our bit-split based matching approach effectively reduces the information that needs to be executed by the aggregation unit and the scoreboard.

Our pattern matching engine is presented on Altera DE2 Development board. FPGA chip in the DE2 board is Cyclone II EP2C70F896C6. We use Quartus II 12.0 Web Edition for hardware synthesis and mapping. QUARTUS II hardware synthesis report (shown in Table II) proved the efficiency of the proposed matching method.

SIMULATION OUTPUT WITH PROPER SYNCHRONIZATION

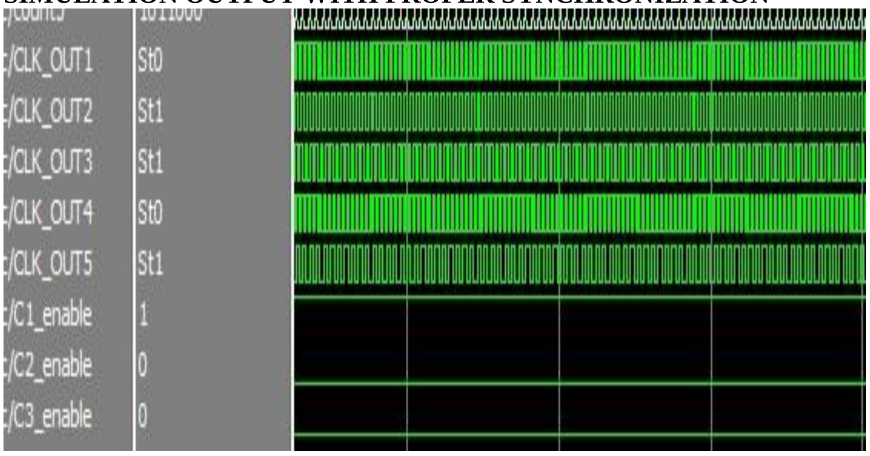


TABLE II Power and Throughput rate on CYCLONE IIEP2C70F9 86C6

	Power consumption	**Speed (char/cycle)**
Bit-split scheme	69.5 mW	1(max 250 MHz clk)
Gated latch control circuit	46.3 mW	1
PEPP	54.7 mW	multiple (char/cycle)

VII. CONCLUSION

Content inspection in intrusion detection system is a computation intensive task. To inspect web-intensive traffic in real-time, hardware accelerators are necessary to perform this action. However, the rate of increase in database is much faster than the rate of increase in the clock frequency of VLSI technology. Hence, there exists a demand to design new hardware architecture that can process multiple bytes of input data per cycle. In this paper, we proposed the bit-split matching approach and could support both basic and regular expression virus signatures.

Experimental results on ClamAV signature database show that our system could support 1Gbps throughput and is more efficient than previous approaches in term of processing speed. Our design mainly bases on memory and could be easily updated for new virus signatures.

The proposed architectural features namely bit-split matching scheme and gated latch control circuit contributes to the improved throughput rate. The page-enabled parallel matching system could exploit the high matching throughput FPGA-based engine while maintain the flexibility and low power consumption. The method of page-enabled parallel computation proposed in this study is only efficient to simple strings. Therefore, the future work will be focused to the design of hardware engine for matching the regexes in the virus database.

REFERENCES

[1]. ClamAV anti-virus system, http://www.clamav.net
[2]. Snort: http://www.Snort.org
[3]. F. Yu, Z. Chen, Y. Diao, T. V. Lakshman and R. H. Katz, "Fast and Memory-Efficient Regular Expression Matching for Deep Packet Inspection". In *Proc. of ANCS*, 2006.
[4]. T. H. Lee, "Generalized Aho-Corasick algorithm for signature based anti-virus applications", *IEEE Int. Conf. Computer, Communications and Networks*, pp. 792-797, 2007.
[5]. Nga Or, Xing Wang and Derek Pao, "Memory-Based Hardware Architectures to Detect ClamAV Virus Signatures with Restricted Regular Expression Features", *IEEE Transactions on Computers*, no. 1, pp. 1, 2015.2439274

[6]. T. Liu, Y. Yang, Y. Liu, Y. Sun and L. Guo, "An efficient regular expressions compression algorithm from a new perspective", *IEEE INFOCOM*, pp.2129-2137, 2011.

[7]. N. Tuck, T. Sherwood, B. Calder and G. Varghese, "Deterministic memory-efficient string matching algorithms for intrusion detection", *IEEE INFOCOM*, pp. 2628-2639, 2004.

[8]. M. Becchi and P. Crowley, "A hybrid finite automaton for practical deep packet inspection", *ACM Conf. on Emerging Network Experiment and Technology (CoNEXT)*, pp.1–12, 2007.

[9]. J. T. L. Ho and G. G. F. Lemieux, "PERG: A scalable FPGA-based pattern-matching engine with consolidated Bloomier filters", *Int. Conf. on Field-Programmable Technology,* pp.73–80, 2008.

[10]. I. Sourdis, S. Vassiliadis, J. Bispo and J.M.P. Cardoso, "Regular expression matching in reconfigurable hardware", *J.of Signal Processing Systems,* Vol.51, pp. 99 –121, 2008.

AUTHOR'S BIOGRAPHY

Rajalakshmi.R received the B.E. degree in electronics and communication engineering from the Thangavelu Engineering College, Anna University, Chennai in 2012. She is currently pursuing the M.E. degree in VLSI Design with the Surya Group of Institutions, Anna University. Her research interests are in Computer Security, Biometric System and Security Solutions.

Anbarasan.A received his B.E Degree in Annamalai University in the year 2009, M.E-VLSI Design in Anna University in the year 2012. He is currently working as an Assistant Professor in the Department of Electronics and Communication at Surya Group of Institutions. He has publications in International Journal of Applied Engineering Research. His area of interest includes VLSI Design, Cryptographic Engineering and Hardware Security.