International Journal of Psychosocial Rehabilitation, Vol. 22, Issue 04, 2018
ISSN: 1475-7192

NETWORK INTRUSION DETECTION SYSTEM USING MODIFIED BLOOM FILTER

K.Saravanan[1], A.Yasmin[2], A.Anitha[3] Shreedharan M.D[4] and Jeganathan M[5]
[1-3] Assistant Professor, Department of Electronics and Communication Engineering, Nehru Institute of Technology, Coimbatore 641 105.
[4]Associate Professor, Excel College of Architecture & Planning, Komarapalayam, Tamil Nadu.
[5]Assistant Professor, Department of Environment and Herbal Sciences, Tamil University, Thanjavur, Tamil Nadu.
anithaayiru@gmail.com jegann1978@gmail.com

ABSTRACT

This paper presents an efficient implementation of the commonly used hash algorithm self-postulated in a modified bloom filter along with the details of standard bloom filter, with the counter bloom filter implementation. We have extended the report with a way of improvising the bloom filter in the network layer so that the threats are blocked in the network as well as to make the network safer.

A Bloom filter is a simple space-efficient randomized data structure for representing a set in order to support membership queries. Bloom filters allow false positives but the space savings often outweigh this drawback when the probability of an error is controlled. In this paper, we the architectural benefits of hybridizing the filter is clearly given, along with the way to construct it in an efficient manner. By future applications we mainly mean Network based intrusion detectors, which can help in controlling the advanced forms of intrusions threat detection. The aim of this report is to survey the ways in which Bloom filters have been used and modified in a variety of network problems, with the aim of providing a unified mathematical and practical framework for understanding them and stimulating their use in future applications.

Keywords: *Bloom Filter, Network Intrusion Detection System.*

INTRODUCTION

An intrusion detection system (IDS) is a device or software application that monitors network or system activities for malicious activities or policy violations and produces reports to a Management Station. Some systems may attempt to stop an intrusion attempt but this is neither required nor expected of a monitoring system. Intrusion detection and prevention systems (IDPS) are primarily focused on identifying possible incidents, logging Information about them, and reporting attempts.

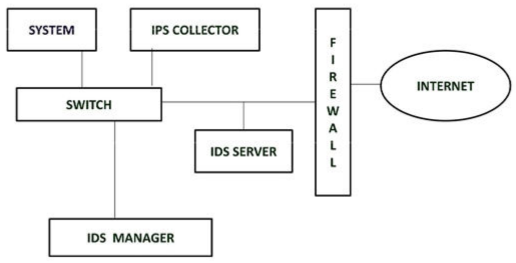


Fig 1: Intrusion Detection System

Comparison of NIDS with Firewalls: Though they both relate to network security, a network intrusion detection system (NIDS) differs from a firewall in that a firewall looks outwardly for intrusions in order to stop them from happening. Firewalls limit access between networks to prevent intrusion and do not signal an attack from inside the network. An NIDS evaluates a suspected intrusion once it has taken place and signals an alarm. An NIDS also watches for attacks that originate from within a system. This is traditionally achieved by examining network communications, identifying heuristics and patterns (often known as signatures) of common computer attacks, and taking action to alert operators. A system that terminates connections is called an intrusion prevention system, and is another form of an application layer firewall. (Vasanthy and Jeganathan 2007, Vasanthy et.al., 2008, Raajasubramanian et.al., 2011, Jeganathan et.al., 2012, 2014, Sridhar et.al., 2012, Gunaselvi et.al., 2014, Premalatha et.al., 2015, Seshadri et.al., 2015, Shakila et.al., 2015, Ashok et.al., 2016, Satheesh Kumar et.al., 2016).

LITERATURE SURVEY

It has proposed a system here, which is a hardware-implementable pattern matching algorithm for content filtering applications, which is scalable in terms of speed, the number of patterns and the pattern length [1]. The algorithm spoken about in this paper is based on a memory efficient multi-hashing data structure called Bloom filter. Embedded on-chip memory blocks in FPGA/VLSI chips are used to construct Bloom filters which can suppress a large fraction of memory accesses and speed up string matching. It has presented hardware architecture for highly efficient intrusion detection systems. By moving both the string matching and the linking of multi-part rules to hardware, their architecture leaves the host system free for higher-level analysis [2]. The tool automates the creation of efficient Field Programmable Gate Array architectures (FPGA). The generated hardware allows an FPGA based system to perform deep-packet inspection of streams at up to 10 Gb/s line rates at a high level of area efficiency. It have introduced a technique, based on a tree-based content addressable memory structure, for a pattern matching engine for use in a hardware-based network intrusion detection system. This technique involves hardware sharing at bit level in order to exploit powerful logic optimizations for multiple strings represented as a Boolean expression [3]. Their approach has been used to implement the entire SNORT rule set with around 12% of the area on a Xilinx XC2V8000

FPGA. The design can run at a rate of approximately 2.5 Gigabits per second, and is approximately 30% smaller in area when compared with published results. In this paper introduces the concept of using internal sensors to perform intrusion detection in computer systems.At a practical level, direct monitoring can be implemented using external or internal sensors. Internal sensors provide advantages with respect to reliability, completeness, timeliness and volume of data, in addition to efficiency and resistance against attacks. This paper introduces a concept of embedded detectors as a mechanism for localized data reduction [4]. Their implementation shows that it is possible to build both specific and generic detectors. Detection testing shows that embedded detectors have the capability of detecting a significant percentage of new attacks. It has proposed a Network Intrusion Detection System (NIDS) embedded in a Smart Sensor inspired device, under a Service Oriented Architecture (SOA) approach, able to operate independently as an anomaly-based NIDS or integrated, transparently, in a Distributed Intrusion Detection System (DIDS). The main goal of the work is to reduce the huge volume of management tasks inherent to this type of network services, as well as facilitating the design of DIDS whose managing complexity could be restricted within well-defined margins [5]. It has coined an FPGA-based implementation of the Bloom filter virus detection code that is compiled from the native C to VHDL and mapped onto a Virtex XC2V8000 FPGA is described in this paper [9]. Their results show that a single engine tailored for handling virus signatures of length eight bytes can achieve a throughput of 18.6 Gbps while occupying only 8% of the FPGA area. It has introduced A k-stage pipelined Bloom filter architecture to decrease power consumption. Pipelined Bloom filter architecture decreases power consumption in comparison to the standard Bloom filter [10]. The 4-stage pipelined Bloom filter is more appropriate than standard Bloom filter when the power consumption is critical. (Manikandan et.al., 2016, Sethuraman et.al., 2016, Senthil Thambi et.al., 2016, Ashok et.al., 2018, Senthilkumar et.al., 2018,).

EXISTING SYSTEM

Standard Bloom Filter: Unlike sets based on hash tables, any Bloom filter can represent the entire universe of elements. In this case, all bits are 1. Another consequence of this property is that add never fails due to the data structure "filling up." However, the false positive rate increases steadily as elements are added until all bits in the filter are set to 1, so a negative value is never returned. At this point, the Bloom filter completely ceases to differentiate between differing inputs, and is functionally useless.

Parallel Bloom Filter: The parallel bloom filters carry out the whole operation of the bloom filter in a parallel fashion in order in perform faster and efficiently. The parallel bloom filter is the basis of many other bloom filters.

Pipelined Bloom Filter: By fully pipelined, it is meant that each stage has only one hash function unlike the two-stage version where there are many hash functions per stage. The fully pipelined Bloom filter has the same number of hash functions as the regular Bloom filter. Hence, the false positive probability is the same. In the query phase, the first hash function, h1, is fed by a new query string every cycle. A query string is progressed to the next stage only when the previous hash function produces a match (i.e., lookup value, LVi=1). The programming phase is the same as the regular Bloom filter.

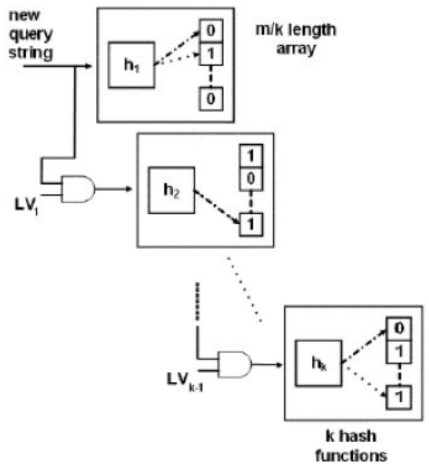


Fig 1: Hardware implementation for the fully pipelined Bloom filter Architecture

PROPOSED SYSTEM

Modified Bloom Filter: Deleting elements from a Bloom filter cannot be done simply by changing ones back to zeros, as a single bit may correspond to multiple elements. To allow for deletions, a counting Bloom filter (CBF) uses an array of n counters instead of bits; the counters track the number of elements currently hashed to that location. Deletions can now be safely done by decrementing the relevant counters. A standard Bloom filter can be derived from a counting Bloom filter by setting all non-zero counts to 1. Counters must be chosen large enough to avoid overflow; for most applications, four bits suffice. We generally use the rule of four bits per counter when comparing results of our data structure with a standard CBF, although we do note that this could be reduced somewhat with some additional complexity. A CBF obtains space savings by allowing false positives.

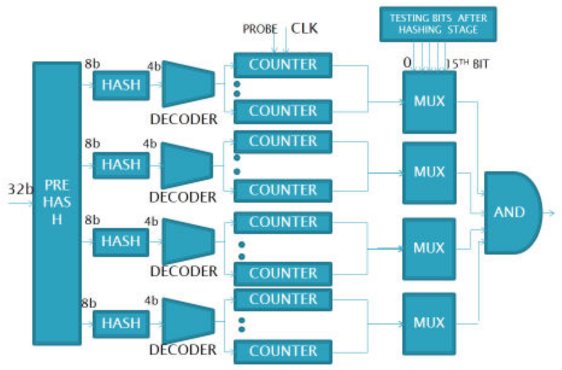


Fig 2: Architecture for Single Pattern Matching Algorithm

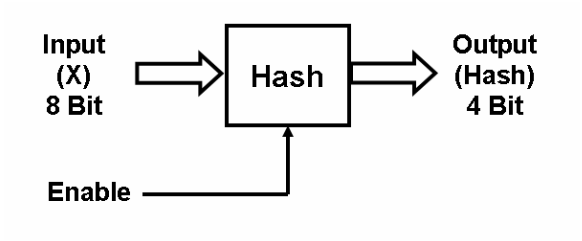
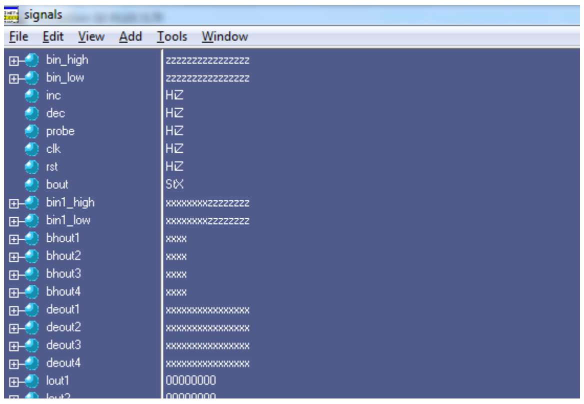


Fig 3: Hash Function Block

Software Analysis: There are two softwares used to the extents for analyzing the data and output in terms of different simulation. ModelSim V5.7 has been mainly used to simulate the output of the software and its output shows a rather good consistency. The modelsim simulations are shown in the result screen shots. The next software used to a larger brief extent is XilinxISE V9.2, Xilinx is a consistent software to perform design implementation and power analysis in such cases of Verilog or VHDL simulation. Xilinx is also a code generating and forging software for the VLSI design kits such as SPARTAN, VERTEX etc.



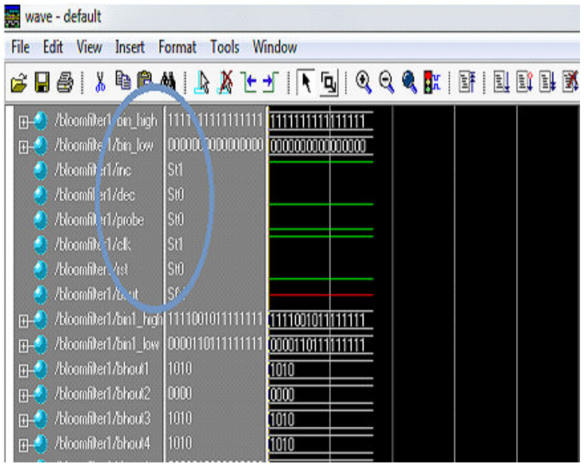
```
bin_high   zzzzzzzzzzzzzzzz
bin_low    zzzzzzzzzzzzzzzz
inc        HiZ
dec        HiZ
probe      HiZ
clk        HiZ
rst        HiZ
bout       StX
bin1_high  xxxxxxxxzzzzzzzz
bin1_low   xxxxxxxxzzzzzzzz
bhout1     xxxx
bhout2     xxxx
bhout3     xxxx
bhout4     xxxx
deout1     xxxxxxxxxxxxxxxx
deout2     xxxxxxxxxxxxxxxx
deout3     xxxxxxxxxxxxxxxx
deout4     xxxxxxxxxxxxxxxx
lout1      00000000
```

Fig 4: Signal Input for the functions

RESULTS AND DISCUSSION

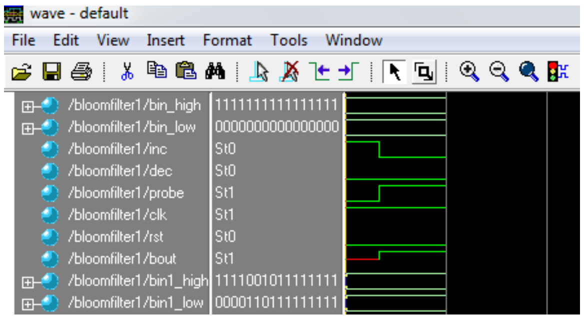
Behavioral Simulation: The above shown figure is a Signal feed dock of ModelSim V5.7 for the Modified bloom filter. The inputs in the above are "bin_low, bin_high", along with the inputs, We use four control triggers Probe, Increment, decrement and clock. Based on the above changes we control the dataflow of the bloom filter. This Bloom filter is a database with reduced string size. The Increment and decrement value is for the control of Up/Down counter whereas the clock is the cyclic inputs, and the value of Probe is to control between the two phases of testing and programming. This would be clearly explained in the below shown simulation and tabulations. There are two phases in the bloom filters output. The first part of the result can be said to as to insertion phase, where the database is created by adding n-number of enteries to the up counters.

The bloom filter takes all these enteries in a encrypted form. The database is a set of hashed hexadecimal bits which are later converged into a very low memory consuming structure.



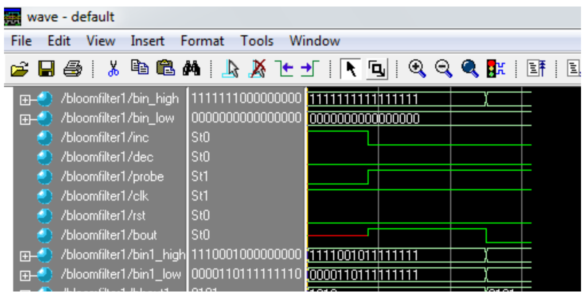
```
/bloomfilter1/bin_high   1111111111111111
/bloomfilter1/bin_low    0000000000000000
/bloomfilter1/inc        St1
/bloomfilter1/dec        St0
/bloomfilter1/probe      St0
/bloomfilter1/clk        St1
/bloomfilter1/rst        St0
/bloomfilter1/bout       [illegible]
/bloomfilter1/bin1_high  1111001011111111
/bloomfilter1/bin1_low   0000110111111111
/bloomfilter1/bhout1     1010
/bloomfilter1/bhout2     0000
/bloomfilter1/bhout3     1010
/bloomfilter1/bhout4     1010
```

Fig 5: The Output for the insertion of the bit.



```
/bloomfilter1/bin_high   1111111111111111
/bloomfilter1/bin_low    0000000000000000
/bloomfilter1/inc        St0
/bloomfilter1/dec        St0
/bloomfilter1/probe      St1
/bloomfilter1/clk        St1
/bloomfilter1/rst        St0
/bloomfilter1/bout       St1
/bloomfilter1/bin1_high  1111001011111111
/bloomfilter1/bin1_low   0000110111111111
```

Fig 6: The output of the testing bits



```
/bloomfilter1/bin_high   1111111000000000  1111111111111111
/bloomfilter1/bin_low    0000000000000000  0000000000000000
/bloomfilter1/inc        St0
/bloomfilter1/dec        St0
/bloomfilter1/probe      St1
/bloomfilter1/clk        St1
/bloomfilter1/rst        St0
/bloomfilter1/bout       St0
/bloomfilter1/bin1_high  1110001000000000  1111001011111111
/bloomfilter1/bin1_low   0000110111111110  0000110111111111
```

Fig 7: The output for a mismatch in the Testing

CONCLUSION AND FUTURE WORK

The result above is for the mismatch, in the case when there is an input, which is not the member of the bloom filters database. The AND gate is the most vital parts of the bloom filter block. Its only because of this gate that we can determine the legitimacy of the entry. In certain cases due to the over filling of bits in the hash array table, Its likely to show some level of false positive and false negatives, which are in other terms caused due to mis-mapping.

REFERENCES

1. Sarang Dharmapurikar, and John Lockwood, Member IEEE. “Fast and Scalable Pattern Matching for Network Intrusion Detection Systems”
2. Zachary K. Baker and Viktor K. Prasanna University of Southern California “High-throughput Linked-Pattern Matching for Intrusion Detection Systems”
3. Sherif Yusuf and Wayne Luk.“Bitwise Optimised CAM For Network Intrusion Detection Systems.”
4. Florian Kerschbaum Eugene H. Spafford Diego Zamboni . “Using internal sensors and embedded detectors for intrusion detection”
5. Francisco Maciá-Pérez, Francisco J. Mora-Gimeno, Diego Marcos-Jorquera,1Juan A. Gil-Martínez-Abarca, HéctoRamos-Morillo and IrenLorenzoFonseca.“Network Intrusion Detection System Embedded on SmartSensor”
6. Antonis Papadogiannakis, Michalis Polychronakis, Evangelos P. Markatos. “Improving the Accuracy of Network Intrusion Detection Systems Under Load Using Selective Packet Discarding.”
7. Lambert Schaelicke, Thomas Slabach, Branden Moore, and Curt Freeland. “Characterizing the Performance of Network Intrusion Detection Sensors”
8. M. Ali Aydın , A. Halim Zaim, K. Gokhan Ceylan. “A hybrid intrusion detection system design for computer network security”
9. Jin-Tae Oh, Byoung-Koo Kim, Seung-Yong Yoon, Jong-Soo Jang, and Yong-Hee Jeon. “Architecture and Mechanisms for Implementing an FPGA-based Stateful Intrusion Detection System”

10. Najmadin Wahid Abdulrahman Boskany. "Network Intrusion Detection System Based on Matching Logical Address and Port Number"
11. Z. Baker and V.K. Prasanna, "High Throughput Linked-Pattern Matching for Intrusion Detection Systems", In Proceedings of Symposium on Architectures for Networking and Communication Systems (ANCS' 05) , Princeton, New Jersey, October 2005
12. I. Sourdis and D. Pnevmatikatos, "Fast, large-scale string match for a 10Gbps FPGA-based network intrustion detection system",. In Proceedings of International Conference on Field Programmable Logic and Applications, 2003
13. Clark, C. R. and Schimmel, D. E., 2004, " Scalable Pattern Matching for High-Speed Networks". IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM), pp. 249-257
14. J. W. Lockwood, N. Naufel, J. S. Turner, and D. E.Taylor. "Reprogrammable Network Packet Process-ing on the Field Programmable Port Extender (FPX)". In ACM International Symposium on Field Programmable Gate Arrays (FPGA'2001), pages 87-93, Monterey, CA, USA, Feb. 2001.
15. M. Roesch. "SNORT - lightweight intrusion detection for networks". In Proceedings of the 13th Systems Ad-ministration Conference, 1999
16. A. Broder and M. Mitzenmacher, "Network applications of Bloomfilters: A survey, "Internet Mathematics, vol. 1, no. 4, pp. 485-509, 2005
17. Sourcefire,Inc.,"Official Snort ruleset," Columbia, for NIDS and servers MD (web:http://www.snort.ort/pub-bin/downloads.cgi).
18. Sarang Dharmapurikar, Praveen Krishnamurthy, Todd Sproull, John Lockwood."Deep Packet Inspection using Parallel Bloom Filters"
19. Mahmood Ahmadi and Stephan Wong. "K-Stage Pipelined Bloom Filter for Packet Classification"
20. Michael Paynter and Taskin Kocak. "Fully Pipelined Bloom Filter Architecture"
21. Michael J. Lyons and David Brooks. "The Design of a Bloom Filter Hardware Accelerator for Ultra Low Power Systems"
22. M. Attig, S. Dharmapurikar, J. Lockwood. "Implementation Results of Bloom Filters for String Matching,", In proceedings of the 12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'04), pages. 322-323, 2004
23. The webhost DaBlooms is an open library helping in learning of the scalable bloom variants http://word.bitly.com/post/28558800777/dablooms-an-open-source-scalable-counting-bloom
24. The Kellabyte host gives a brief view on inexpensive bloom filter preparation processhttp://kellabyte.com/2013/01/24/using-a-bloom-filter-to-reduce-expensive-operations-like-disk-io/
25. The Website Stack overflow is a giant database for bloom filter, in a report it shows the efficiency of bloom filter, http://stackoverflow.com/questions/7049027/how-do-bloom-filter-implementations-keep-clean.

26. The Hindawi programmers developed a bloom filter built only on cache memory .http://www.hindawi.com/journals/jece/2011/475865/
27. A clipboard on i-prog's where a new concept of invertible bloom filter is quoted and elaborated, http://www.i-programmer.info/programming/theory/4641-the-invertible-bloom-filter.html
28. linux platform has developed a linux based database for the bloom filter, http://xlinux.nist.gov/dads/HTML/bloomFilter.html
29. Deng, Fan; Rafiei, Davood (2006), "Approximately Detecting Duplicates for Streaming Data using Stable Bloom Filters", Proceedings of the ACM SIGMOD Conference, pp. 25–36
30. Broder, Andrei; Mitzenmacher, Michael (2005), "Network Applications of Bloom Filters: A Survey", Internet Mathematics 1 (4): 485–509.
31. Charles, Denis; Chellapilla, Kumar (2008), "Bloomier Filters: A second look", The Computing Research Repository (CoRR), arXiv:0807.0928
32. Chazelle, Bernard; Kilian, Joe; Rubinfeld, Ronitt; Tal, Ayellet (2004), "The Bloomier filter: an efficient data structure for static support lookup tables", Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 30–39
33. Cohen, Saar; Matias, Yossi (2003), "Spectral Bloom Filters", Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, pp. 241–252.
34. Dharmapurikar, Sarang; Song, Haoyu; Turner, Jonathan; Lockwood, John (2006), "Fast packet classification using Bloom filters", Proceedings of the 2006 ACM/IEEE Symposium on Architecture for Networking and Communications Systems
35. Dillinger, Peter C.; Manolios, Panagiotis (2004b), "Bloom Filters in Probabilistic Verification", Proceedings of the 5th Internation Conference on Formal Methods in Computer-Aided Design, Springer-Verlag, Lecture Notes in Computer Science 3312
36. Donnet, Benoit; Baynat, Bruno; Friedman, Timur (2006), "Retouched Bloom Filters: Allowing Networked Applications to Flexibly Trade Off False Positives Against False Negatives", CoNEXT 06 – 2nd Conference on Future Networking Technologies
37. Eppstein, David; Goodrich, Michael T. (2007), "Space-efficient straggler identification in round-trip data streams via Newton's identities and invertible Bloom filters", Algorithms and Data Structures, 10th International Workshop, WADS 2007, Springer-Verlag, Lecture Notes in Computer Science 4619, pp. 637–648.
38. Goel, Ashish; Gupta, Pankaj (2010), "Small subset queries and bloom filters using ternary associative memories, with applications", ACM Sigmetrics 2010 38: 143.
39. Kirsch, Adam; Mitzenmacher, Michael (2006), "Less Hashing, Same Performance: Building a Better Bloom Filter", in Azar, Yossi; Erlebach, Thomas, Algorithms – ESA 2006, 14th Annual European Symposium, Lecture Notes in Computer Science 4168, Springer-Verlag, Lecture Notes in Computer Science 4168, pp. 456–467.

40. Pagh, Anna; Pagh, Rasmus; Rao, S. Srinivasa (2005), "An optimal Bloom filter replacement", Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 823–829
41. Putze, F.; Sanders, P.; Singler, J. (2007), "Cache-, Hash- and Space-Efficient Bloom Filters", in Demetrescu, Camil, Experimental Algorithms, 6th International Workshop, WEA 2007, Lecture Notes in Computer Science 4525, Springer-Verlag, Lecture Notes in Computer Science 4525.
42. “Space/time tradeoffs in hash coding with allowable errors”, Communications of the ACM, 13(7): pages 422-426, July 1976.
43. M. Ramakrishna, E. Fu, and E. Bahcekapili--A performance study of hashing functions for hardware applications. InProc. of Int. Conf. on Computing and Information, pages 1621-1636, 1994.
44. Kaya and T. Kocak, “A low power lookup technique for multi-hashing network applications,” inProc. IEEE Annual Symp. on VLSI (ISVLSI), Karlsruhe, Germany, 2006.
45. The hash process of bloom filter was best narrated in the powerpoint from the www.eecs.harvard.edu/~michaelm/TALKS/HashingTalk.ppt
46. Bret Mulvey, Evaluation of CRC32 for Hash Tables, in Hash Functions. Accessed April 10, 2009.
47. Bret Mulvey, Evaluation of SHA-1 for Hash Tables, in Hash Functions. Accessed April 10, 2009
48. The Intial stage of understandings were done purely based on Web Search and study from the “billmill.org/bloomfilter-tutorial/”
49. The basics of counting bloom filter is clearly explained in an effective seminar reported by the website http://www.allthingsdistributed.com/2012/09/counting-bloom-filters.html
50. Bonomi, Flavio; Mitzenmacher, Michael; Panigrahy, Rina; Singh, Sushil; Varghese, George (2006), "An Improved Construction for Counting Bloom Filters", Algorithms – ESA 2006, 14th Annual European Symposium, Lecture Notes in Computer Science 4168, pp. 684–695
51. Ahmadi, Mahmood; Wong, Stephan (2007), "A Cache Architecture for Counting Bloom Filters", 15th international Conference on Networks (ICON-2007), p. 218.
52. Mano, M. Morris and Charles R. Kime. Logic and Computer Design Fundamentals, ThirdEdition.PrenticeHall,2004.p.73.,pp.518–523,Sep./oct.1981.
53. Vasanthy M and M. Jeganathan. 2007. Ambient air quality in terms of NOx in and around Ariyalur, Perambalur DT, Tamil Nadu. Jr. of Industrial pollution Control., 23(1):141-144.
54. Vasanthy. M ,A.Geetha, M. Jeganathan,and A.Anitha. 2007. A study on drinking water quality in Ariyalur area. J.Nature Environment and Pollution Technology. 8(1):253-256.

55. Ramanathan R ,M. Jeganathan, and T. Jeyakavitha. 2006. Impact of cement dust on azadirachtain dicaleaves – ameasure of air pollution in and Around Ariyalur. J. Industrial Pollution Control. 22 (2): 273-276.
56. Vasanthy M and M. Jeganathan. 2007. Ambient air quality in terms of NOx in and around Ariyalur, Perambalur DT, Tamil Nadu. Pollution Research., 27(1):165-167.
57. Vasanthy M and M. Jeganathan. 2008.Monitoring of air quality in terms of respirable particulate matter – A case study. Jr. of Industrial pollution Control.,24(1):53 - 55.
58. Vasanthy M, A.Geetha, M. Jeganathan, and M. Buvaneswari. 2008. Phytoremediation of aqueous dye solution using blue devil (Eichhornia crassipes). J. Current Science. 9 (2): 903-906.
59. Raajasubramanian D, P. Sundaramoorthy, L. Baskaran, K. Sankar Ganesh, AL.A. Chidambaram and M. Jeganathan. 2011. Effect of cement dust pollution on germination and growth of groundnut (*Arachis hypogaea* L.). IRMJ-Ecology. International Multidisciplinary Research Journal 2011, 1/1:25-30 : ISSN: 2231-6302: Available Online: http://irjs.info/.
60. Raajasubramanian D, P. Sundaramoorthy, L. Baskaran, K. Sankar Ganesh, AL.A. Chidambaram and M. Jeganathan. 2011. Cement dust pollution on growth and yield attributes of groundnut. (*Arachis hypogaea* L.). IRMJ-Ecology. International Multidisciplinary Research Journal 2011, 1/1:31-36.ISSN: 2231-6302. Available Online: http://irjs.info/
61. Jeganathan M, K. Sridhar and J.Abbas Mohaideen. 2012. Analysis of meterological conditions of Ariyalur and construction of wind roses for the period of 5 years from January 2002. J.Ecotoxicol.Environ.Monit., 22(4): 375-384.
62. Sridhar K, J.Abbas Mohaideen M. Jeganathan and P Jayakumar. 2012. Monitoring of air quality in terms of respirable particulate matter at Ariyalur, Tamilnadu. J.Ecotoxicol.Environ.Monit., 22(5): 401-406.
63. Jeganathan M, K Maharajan C Sivasubramaniyan and A Manisekar. 2014. Impact of cement dust pollution on floral morphology and chlorophyll of *healianthus annus* plant – a case study. J.Ecotoxicol.Environ.Monit., 24(1): 29-34.
64. Jeganathan M, C Sivasubramaniyan A Manisekar and M Vasanthy. 2014. Determination of cement kiln exhaust on air quality of ariyalur in terms of suspended particulate matter – a case study. IJPBA. 5(3): 1235-1243. ISSN:0976-3333.
65. Jeganathan M, S Gunaselvi K C Pazhani and M Vasanthy. 2014. Impact of cement dust pollution on floral morphology and chlorophyll of *healianthus annus.*plant a case study. IJPBA. 5(3): 1231-1234. ISSN:0976-3333.
66. Gunaselvi S, K C Pazhani and M. Jeganathan. 2014. Energy conservation and environmental management on uncertainty reduction in pollution by combustion of swirl burners. J. Ecotoxicol. Environ.Monit., 24(1): 1-11.
67. Jeganathan M, G Nageswari and M Vasanthy. 2014. A Survey of traditional medicinal plant of Ariyalur District in Tamilnadu. IJPBA. 5(3): 1244-1248. ISSN:0976-3333.

68. Premalatha P, C. Sivasubramanian, P Satheeshkumar, M. Jeganathan and M. Balakumari.2015. Effect of cement dust pollution on certain physical and biochemical parameters of castor plant (*ricinus communis*). IAJMR.1(2): 181-185.ISSN: 2454-1370.
69. Premalatha P, C. Sivasubramanian, P Satheeshkumar, M. Jeganathan and M. Balakumari.2015. Estimation of physico-chemical parameters on silver beach marine water of cuddalore district. Life Science Archives. 1(2): 196-199.ISSN: 2454-1354.
70. Seshadri V, C. Sivasubramanian P. Satheeshkumar M. Jeganathan and Balakumari.2015. Comparative macronutrient, micronutrient and biochemical constituents analysis of *arachis hypogaea.* IAJMR.1(2): 186-190.ISSN: 2454-1370.
71. Seshadri V, C. Sivasubramanian P. Satheeshkumar M. Jeganathan and Balakumari.2015. A detailed study on the effect of air pollution on certain physical and bio chemical parameters of *mangifera indica* plant.Life Science Archives. 1(2): 200-203.ISSN: 2454-1354.
72. Shakila N, C. Sivasubramanian, P. Satheeshkumar, M. Jeganathan and Balakumari.2015. Effect of municipal sewage water on soil chemical composition- A executive summary. IAJMR.1(2): 191-195.ISSN: 2454-1370.
73. Shakila N, C. Sivasubramanian, P. Satheeshkumar, M. Jeganathan and Balakumari.2015. Bacterial enumeration in surface and bottom waters of two different fresh water aquatic eco systems in Ariyalur, Tamillnadu. Life Science Archives. 1(2): 204-207.ISSN: 2454-1354.
74. Ashok J, S. Senthamil kumar, P. Satheesh kumar and M. Jeganathan. 2016. Analysis of meteorological conditions of ariyalur district. Life Science Archives. 2(3): 579-585.ISSN: 2454-1354. DOI: 10.21276/lsa.2016.2.3.9.
75. Ashok J, S. Senthamil Kumar, P. Satheesh Kumar and M. Jeganathan. 2016. Analysis of meteorological conditions of cuddalore district. IAJMR.2 (3): 603-608.ISSN: 2454-1370. DOI: 10.21276/iajmr.2016.2.3.3.
76. Satheesh Kumar P, C. Sivasubramanian, M. Jeganathan and J. Ashok. 2016. South Indian vernacular architecture -A executive summary. IAJMR.2 (4): 655-661.ISSN: 2454-1370. DOI: 10.21276/iajmr.2016.2.3.3.
77. Satheesh Kumar P, C. Sivasubramanian, M. Jeganathan and J. Ashok. 2016. Green buildings - A review. Life Science Archives. 2(3): 586-590.ISSN: 2454-1354. DOI: 10.21276/lsa.2016.2.3.9.
78. Satheesh Kumar P, C. Sivasubramanian, M. Jeganathan and J. Ashok. 2016. Indoor outdoor green plantation in buildings - A case study. IAJMR.2 (3): 649-654.ISSN: 2454-1370. DOI: 10.21276/iajmr.2016.2.3.3.
79. Manikandan R, M. Jeganathan, P. Satheesh Kumar and J. Ashok. 2016. Assessment of ground water quality in Cuddalore district, Tamilnadu, India. Life Science Archives. 2(4): 628-636.ISSN: 2454-1354. DOI: 10.21276/lsa.2016.2.3.9.

80. Manikandan R, M. Jeganathan, P. Satheesh Kumar and J. Ashok. 2016. A study on water quality assessment of Ariyalur district, Tamilnadu, India. IAJMR.2 (4): 687-692.ISSN: 2454-1370. DOI: 10.21276/iajmr.2016.2.3.3.
81. Sethuraman G, M. Jeganathan, P. Satheesh Kumar and J. Ashok. 2016. Assessment of air quality in Ariyalur, Tamilnadu, India. Life Science Archives. 2(4): 637-640.ISSN: 2454-1354. DOI: 10.21276/lsa.2016.2.3.9.
82. Sethuraman G, M. Jeganathan, P. Satheesh Kumar and J. Ashok. 2016. A study on air quality assessment of Neyveli, Tamilnadu, India. IAJMR.2 (4): 693-697.ISSN: 2454-1370. DOI: 10.21276/iajmr.2016.2.3.3.
83. Senthil Thambi J, C. Sivasubramanian and M. Jeganathan. 2018. Ambient Air quality monitoring in terms of (Nitrogen di oxide in and around Ariyalur District, Tamilnadu, India. IAJMR.4 (3): 1414-1417.ISSN: 2454-1370. DOI: 10.22192/iajmr.2018.4.3.2.
84. Senthil Thambi J, C. Sivasubramanian and M. Jeganathan. 2018. Study of Air pollution due to vehicle emission in Ariyalur District, Tamilnadu, India. Life Science Archives. 4(4): 1409-1416.ISSN: 2454-1354. DOI: 10.22192/lsa.2018.4.4.3.
85. Ashok J, S.Senthamil kumar, P.Satheesh kumar and M.Jeganathan. 2018. Estimation of Cement kiln exhaust on Air quality of Ariyalur in terms of suspended particulate matter - A Case Study. International Journal Of Civil Engineering And Technology. 9 (12): Scopus Indexed Journal ISSN: 0976 – 6316.
86. Ashok J, S.Senthamil kumar, P.Satheesh kumar and M.Jeganathan.2018. Air quality assessment of Neyveli in Cuddalore District, Tamilnadu, India. International Journal Of Civil Engineering And Technology. 9 (12): Scopus Indexed Journal ISSN: 0976 – 6316.
87. Senthilkumar M, N. Nagarajan, M. Jeganathan and M. Santhiya. 2018. Survey of Medicinal Plants diversity on Bodha Hills in Salem District, Tamil Nadu, India. Indo – Asian Journal Of Multidisciplinary Research (IAJMR) ISSN: 2454-1370.
88. Senthilkumar M, N. Nagarajan, M. Jeganathan and M. Santhiya. 2018. Survey of Traditional Medicinal Plants in and around Ariyalur in TamilNadu, India. Life Science Archives (LSA) ISSN: 2454-1354. DOI: 10.22192/lsa.2018.4.6.5.
89. Malarvannan J, C. Sivasubramanian, R. Sivasankar, M. Jeganathan and M. Balakumari. 2016. Shading of building as a preventive measure for passive cooling and energy conservation – A case study. Indo – Asian Journal of Multidisciplinary Research (IAJMR): ISSN: 2454-1370. Volume – 2; Issue - 6; Year – 2016; Page: 906 – 910. DOI: 10.21276.iajmr.2016.2.6.10.
90. Malarvannan J, C. Sivasubramanian, R. Sivasankar, M. Jeganathan and M. Balakumari. 2016. Assessment of water resource consumption in building construction in tamilnadu, India. Life Science Archives (LSA) ISSN: 2454-1354 Volume – 2; Issue - 6; Year – 2016; Page: 827 – 831 DOI: 10.21276/lsa.2016.2.6.7.
91. Sivasankar R, C. Sivasubramanian, J. Malarvannan, M. Jeganathan and M. Balakumari. 2016. A Study on water conservation aspects of green buildings. Life Science

Archives (LSA),ISSN: 2454-1354. Volume – 2; Issue - 6; Year – 2016; Page: 832 – 836, DOI: 10.21276/lsa.2016.2.6.8.

92. Ashok J , S. Senthamil Kumar , P. Satheesh Kumar and M. Jeganathan. 2016. Analysis and design of heat resistant in building structures. Life Science Archives (LSA), ISSN: 2454-1354. Volume – 2; Issue - 6; Year – 2016; Page: 842 – 847. DOI: 10.21276/lsa.2016.2.6.10.