Circuits and Systems, 2016, 7, 1324-1333
Published Online June 2016 in SciRes. http://www.scirp.org/journal/cs
http://dx.doi.org/10.4236/cs.2016.78115

High Speed and Low Power Architecture for Network Intrusion Detection System

Palanisamy Brindha[1*], Athappan Senthilkumar[2*]

[1]Department of Information and Communication Engineering, Anna University, Tamil Nadu, India
[2]Department of Electrical and Electronics Engineering, Dr. Mahalingam College of Engineering and Technology, Tamil Nadu, India
Email: *brindhavlsi@gmail.com

Received 19 March 2016; accepted 20 April 2016; published 8 June 2016

Abstract

The tremendous growth in the field of modern communication and network systems places demands on the security. As the network complexity grows, the need for the automated detection and timely alert is required to detect the abnormal activities in the network. To diagnose the system against the malicious signatures, a high speed Network Intrusion Detection System is required against the attacks. In the network security applications, Bloom Filters are the key building block. The packets from the high speed link can be easily processed by Bloom Filter using state-of-art hardware based systems. As Bloom Filter and its variant Counting Bloom Filter suffer from False Positive Rate, Multi Hash Counting Bloom Filter architecture is proposed. The proposed work, constitute parallel signature detection improves the False Positive Rate, but the throughput and hardware complexity suffer. To resolve this, a Multi-Level Ranking Scheme is introduced which deduces the 13% - 16% of the power and increases the throughput to 23% - 30%. This work is best suited for signature detection in high speed network.

Keywords

Intrusion Detection, Bloom Filter, Counting Bloom Filter, False Positive

1. Introduction

The technology development and innovation have become part of our life. An ever-increasing dependence on technology consolidates itself as a powerful platform that has revolutionized the way we do business and communicate with people, leaving us in the open to threats of cyber crime. KPMG in India is one of the chief pro-

*Corresponding author.

viders of risk, tax & regulatory services, financial & business advisory, internal audit, and corporate governance. KPMG was established in September 1993 in India, and they conducted the KPMG: Cybercrime survey report 2014 [1]. Software-based pattern matcher systems have been used for several years, but with the evolution of Giga byte networks these systems have become bottlenecks. So the best solution is to use a hardware based pattern matching system to trade off with the Gigabyte Ethernet.

Many algorithms were proposed earlier which was not suited for the growing line rate of Gbps. Aho and Corasick algorithm has been widely adapted for string matching [2]. By reducing the state transition, it minimizes the memory size. The Boyer-Moore (BM) algorithm is based on heuristic algorithm. The heuristic matches the characters from the suffix of the search window. The heuristic refers the lookup table whenever a mismatch is found [3]. The distance is shifted, based on the pattern matching.

An algorithm matches a block of character instead of matching a single character at a time. Based on the scanning of the right most block of the character in the window, a shift process is followed [4]. The time complexity is O (mn).

To enhance the through researches are being done in hardware based Intrusion Detection System (IDS). For hardware-based solutions, FPGA (Field Programmable Array Gates) and TCAM (Ternary Content Addressable Memory) are implemented for their parallelism capabilities and find that 87% of Snort rules have patterns. So he proposed a hardware accelerator for the pattern matching [5]. An Extended TCAM was proposed to reduce data structures [6]. Each signature-based and anomaly-based IDS has its advantages and disadvantages. Today, it is being observed that numerous antivirus packages include both signature-based and anomaly-based detection features, while only a handful of IDSs effect an incorporation of both approaches.

Even though intrusion detection has progressed rapidly in the past few decades, many issues remain unsolved. The first and foremost criteria are that it should be effective and should detect a wide range of attacks with minimal false positives. Second, the intrusion detection should keep up with the increasing Giga bit per second (Gbps) speed. The system should be dynamic to handle new attacks in real future. The detection system takes care of the host, but additionally it should protect the network too.

The rest of the paper is organized as follows: Section 2 describes the background and related work of Bloom Filter. Section 3 illustrates the existing Counting Bloom Filter architecture. Multi Hash Counting Bloom Filter is proposed in Section 4. Section 5 enumerates the Single and Multi-Level Ranking Scheme and finally Section 6 gives the Validation for the proposed work. Section 7 deals with conclusion.

2. Literature Review

A Set-Wise Boyer-Moore-Horspool (SBMH) proposed the first hybrid SB-NIDS that adapts the specific pattern matching Horspool variant of the Boyer-Moore algorithm (BM) and AC algorithm [7]. This hybrid technique works on the multi pattern matching search approach of AC algorithm to match the pattern either in single or multiple loops. The pattern matching is done by creating a suffix pattern tree and Horspool bad-character heuristic is employed with bad-character shift table. The bad character shift tree and suffix tree is built by pre-processing the patterns.

AC and SBMH algorithm is verified for various snort attack rules. The number of rules selected in the evaluation process will decide the choice of algorithm for implementation. SBMH is selected if there is only 1 rule, SBMH is selected if there are 2 to 100 rules and AC algorithm is selected when the rule is above 100. SBMH algorithm is much better and improves the overall Snort performance by a factor of 5 for web traffic applications.

An algorithm named Aho-Corasick-Boyer-Moore (AC-BM) which combines the multi-pattern search of Aho-Corasick and character skip feature of Boyer-Moore [8]. AC-BM algorithm uses the heuristic information for building a shift-table and utilizes the pre-process patterns to construct the pattern tree for multi-pattern search support. AC-BM search is held by two shift tables instead of one. These shift tables are good-character shift table and bad-prefix shift table.

BM algorithm achieved better performance than AC-BM algorithm. However, AC-BM algorithm shows outstanding performance when the test was skewed by the elimination of the non-content attack rules. When tested by Snort content attack rules of about 200 AC-BM found 1.31 times faster than Boyer Moore and 3.32 times faster with 786 rules.

Further HIDS can be classified based on Memory, State Machine, Hashing and Brute Force search. A hybrid hardware model is where Content addressable Memory (CAM) is employed for checking the packet header and payload [9]. The hardware provided an effective throughput of 2.0 Gbps. Further (Ioannis Sourdis and Dionisios Pnevmatikatos 2004) designed a pre-decoder for CAM for pattern matching. A different combination of techniques was adopted to deduce the area cost. This CAM based design achieved a throughput of 3 Gbps on a Virtex2 board. The author also introduced quad parallelism to increase the throughput for pattern matching of almost 10 Gbps. Like CAM, TCAM also plays a vital role in the field of networking. As (Rajendra Shinde *et al* 2010) presented a TCAM based approach, that allow wildcard searches which makes them a very powerful algorithm primitive. They experimented for an open source Snort of 5000 and found that the throughput is approximately 10 Gbps using TCAM.

Lin Tan and Timothy Sherwood developed an approach that solves the difficulties in transferring a bulk database of strings into a small state machine. These state machines are employed in searching the rule set [10]. They found that this new approach is 10 times more efficient than earlier approaches. A new technique has been introduced to have a memory efficient architecture. In this the state machine is converted into state transition tables which lead to parallel search with packets [11]. Still state machine based techniques (Finite Automata and Non-Finite Automata) are growing at a fast rate to have more and more efficient search with optimal parameters

A function that takes up any function of random length and produces a digital data of fixed length, then it is termed as the hash function. Normally a value that is obtained as the output from hash function is named a hash value, hash code, sometimes simply hashes. It is often used in data structures for querying the data. One such technique is a Bloom filter. After the introduction of Bloom Filter by Burton Howard Bloom in 1970, the hardware based IDS thrived a lot [12]. Bloom filter is a space-efficient probabilistic structure that is used to test whether an element is a member of a set.

So, IDS has to detect the attacks, in addition it should be space/time efficient. As it has been proven that the Bloom Filter provides a better solution for designing IDS, this review focus mainly on Bloom Filter based Hardware based IDS.

Basically a Bloom Filter allows false positive, but never False Negative. This feature tracks the Intrusion Detection System designers to use Bloom Filter for their designs. Bloom gave an example of a hyphenation algorithm for a dictionary of 500,000 words; out of which 90% follow simple hyphenation rules, but the residual 10% require expensive disk accesses to repossess specific hyphenation patterns. With sufficient core memory, an error-free hash could be used to purge all unnecessary disk accesses. Conversely, Bloom technique uses little hash area, but still eliminates most unnecessary accesses. For example, a hash requires only 15% of the size needed by an ideal error-free hash thereby it reduces the 85% of the disk accesses [13].

Burton H. Bloom introduced a new hash coding. This method is suggested for application in which the great majority of messages to be tested will not belong to the large set. First, the average time required for classifying the element as a non-member of large set is high. Second, the probability of error should be minimized (*i.e.*) the false identification of the member to be in the set will create a small error. Third, computation time and space should be efficient to meet the practical applications [14] [15].

Bloom Filter (BF) is a compact space efficient algorithm. Consider a set $S=\{a_1,a_2,a_3,\cdots,a_n\}$ of "*n*" elements and a set $H=\{h_1,h_2,h_3,\cdots,h_k\}$ of "*k*" independent and uniformly distributed hash function. The individual hash, say h_1 will range from $\{0,1,2,\cdots,m-1\}$. Consider an example: If $S=\{\text{AA},\text{AB},\text{BA},\text{BB},\cdots,\text{ZZ}\}$ where n = 100 and assume k = 3, then, $H=\{h_1,h_2,h_3\}$. Using k hash function of H makes each element of S map to the vector whose size is m. An array is constructed whose length is "m". The initial value in the array is maintained at zero. As the hash function starts, depending on the hash output the corresponding locations are set to one. On the other hand, the query is done to check whether the element belongs to set. The problem with the standard BF is, the same location is accessed for multiple times based on the hash output.

To solve this problem Fan *et al.* introduced CBF [15]. In a CBF, each array is replaced with counter instead of a single entry. Once the element is inserted the corresponding counter is incremented and when an element has to be deleted, the corresponding counter is decremented by one.

Still the False positive remains the same as that of the BF. To overcome this drawback a Multi-Hash Technique is proposed. Next, to improve the speed and to reduce the power a Multi-level Ranking scheme is used in the proposed work.

3. Counting Bloom Filter

The Counting Bloom Filter (CBF) is organized as an array of counter indexed by the hash function. This is depicted in **Figure 1**. CBFs possess three basic operations: 1) increment count 2) decrement count and 3) test to check the count is zero. The first two operations increment or decrement the corresponding counter by one which is also named as update phase, and the third operation tracks the counter to ensure the output which is normally either 0 or 1 is termed as test phase. A single bit output is monitored, if the count is Zero it returns true else false. The working of CBF is originally characterized by two variants, one is the number of entries and the other is the width of count per entry.

3.1. Hash Function

Pattern matching using CBF begins with the hash function, where the incoming address or pattern of n-bit can be mapped to a fixed m-bit length. Simple hash function helps to index the array of large size, where the data has to be stored. While querying, the same hash index is used to check whether the pattern is present or not. Ramakrishna *et al.* described a class of universal hash functions that are suitable for this hardware implementation [16]. To generate the necessary k hash functions, the class H_3 class hash function is performed.

The class H_3 is defined as follows:

Let Q denotes the set of all $i \times j$ Boolean matrices. For a given $q \in Q$ and $x \in A$, let $q(k)$ be the row of the matrix q and x_k the k^{th} bit of x. The hashing function $h_q(x)$: $A \rightarrow B$ is defined (Ramakrishna *et al.* 1994) by Equation (1),

$$h_q(x) = x_1 \cdot q(1) \oplus x_2 \cdot q(2) \oplus x_3 \cdot q(3) \oplus \cdots \oplus x_i \cdot q(i) \tag{1}$$

Above universal hash function can be presented with different notations as follows. Consider an input address whose bits are $X = \{x_1, x_2, x_3, x_4, \cdots, x_n\}$ The address bits first pass through the universal hash function block based on the random value "d_{im}" is shown in Equation (2),

$$H_i(X) = x_{i1} \cdot d_{i1} \oplus x_{i2} \cdot d_{i2} \oplus x_{i3} \cdot d_{i3} \oplus \cdots \oplus x_{in} \cdot d_{im} \tag{2}$$

where "$\cdot$" is bit wise AND operator and "$\oplus$" is a bit wise XOR operator. d_{im} is a predetermined random number which is in the range of $(0, \cdots, m-1)$. Single hash function sometimes increases the False Positive Rate. The problem in CBF is one cannot differentiate the entries in the CBF for which each input items is done. Due to this, sometimes an element may be wrongly alarmed to be a virus to the administrator, when it is not. This is said to be False Positive. To overcome Multi Hash CBF is proposed.

3.2. False Positive Rate of Counting Bloom Filter

FPR of CBF is given by assuming n as the number of elements; m is the number of bit-vector; and k is the total number of hash functions.

The probability that any particular bit $H_i = 0$ is

$$(1 - 1/m)^{kn} \tag{3}$$

The probability that a particular bit $H_i = 1$ is

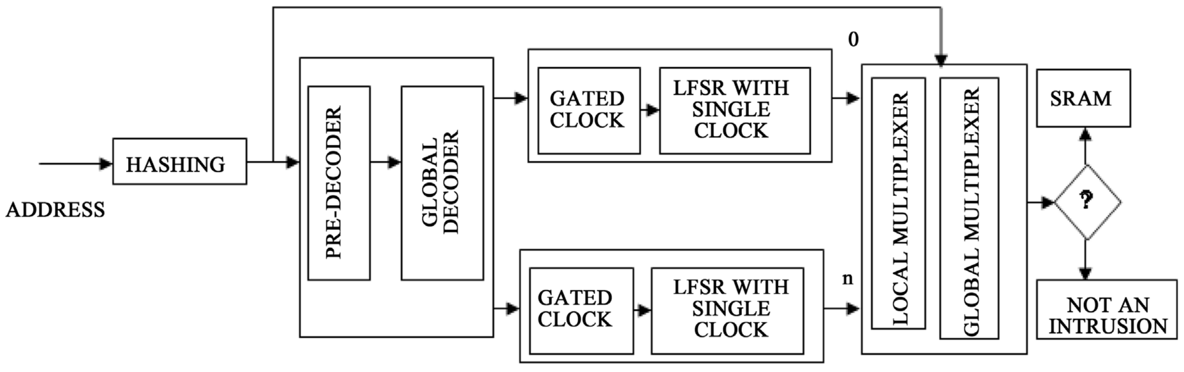

Figure 1. Internal architecture of counting bloom filter.

$$P = 1 - (1 - 1/m)^{kn} \tag{4}$$

For an element *x*, if the query result in a false positive which means that each of the *k* hash values must be the index of a bit that is set to 1. The probability that this happens is

$$P_{k,n,m} = \Pr\{H_{x1} = \text{and } H_{x2} \text{ and} \cdots H_{xk} = 1\}$$

which is claimed to be

$$P^k = \left(1 - (1 - 1/m)^{kn}\right)^k \tag{5}$$

Equation (5) found in many papers is incorrect. Consider an example, to prove the incorrect statement. Assume $n = 1, k = 2$ and $m = 2$ where two elements are involved. The element *x*is stored in the table, whereas some element *y* is not stored. The presence of false positive for element *y* is the condition among x_1, x_2, y_1, y_2. Among the 16 possibilities, the false positive is 5/8, given by Equation (6).

$$P^k = \left(1 - (1 - 1/2)^{2.1}\right)^2 = 9/16 \Rightarrow 4.5/8 \tag{6}$$

In almost all the applications the value of k chosen and hence *FPR* is constant and its value is nearly equal to 1/2 as shown in Equation (8).

$$FPR = \left(1 - (1 - 1/m)^{kn}\right)^k \tag{7}$$

If $k = \ln 2$ and $m = \infty$, then

$$FPR = \left(\frac{1}{2}\right)^k \tag{8}$$

4. Multi Hash Counting Bloom Filter

There are few applications that require multiple or simultaneous CBF accesses to memory locations. In the single hash implementation, there is a chance of collision if multiple accesses take place. To overcome this problem a Multi-Hash Function architecture is proposed as shown in **Figure 2**.

The proposed architecture takes the input signature and multiple hash functions are used to index the counter array. In traditional CBF only single hash function is employed. This single hash may index the same counter location for different inputs. While querying this leads to collision, multiple hash function is employed in the proposed architecture to have multiple checking which reduces the occurrence of false positive.

For example, if an element is hashed to $h_1(x) = 5$ and if another element is hashed to the same location $h_1(y) = 5$, then the CBF in the 5th location will be incremented to two. If a deletion operation has to be performed, the CBF will be decremented based on the hash output. So in this typical case the decrement may lead to false positive as the two element hash to the same location. If query is made on the CBF, the result may end up with false positives. To avoid multihash CBF can be employed as depicted in **Figure 2**.

For the traditional CBF, the decoder module, multiplexer and the zero detectors have to be replicated for performing multiple accesses. In case of multiple accesses in CBF, the multi-porting as in SRAM cells are not a straight forward solution. A simple and easiest solution is to detect the output from different hash function and then serializing them to access the CBF. An extra circuitry is added to determine the collective effect of all the accesses. For example, if the $h_1(x) = 5$ and $h_2(x) = 3$, then the CBF has to increment both the location by 1. On querying the element *x*, the membership test is carried out by checking 5th and 3rd locations. On the other hand, if $h_1(x) = 5$ and $h_2(x) = 5$, then the CBF increments the 5th location and query operation on this will provide an answer "Is a Member". In the query process, the input signature is compared with the stored signature. The ouput is either logic 1 (is present or is a member) or logic 0 (is absent or not a member). The capability of the circuit to access multiple hash function in a single clock cycle is required. Further, the false positive rate improves but the hardware complexity grows with the increase in the number of hash function and in turn power consumption also increases. To resolve this, a Multi-Level Ranking scheme is introduced in the multi hash architecture to reduce the power and to improve the throughput.

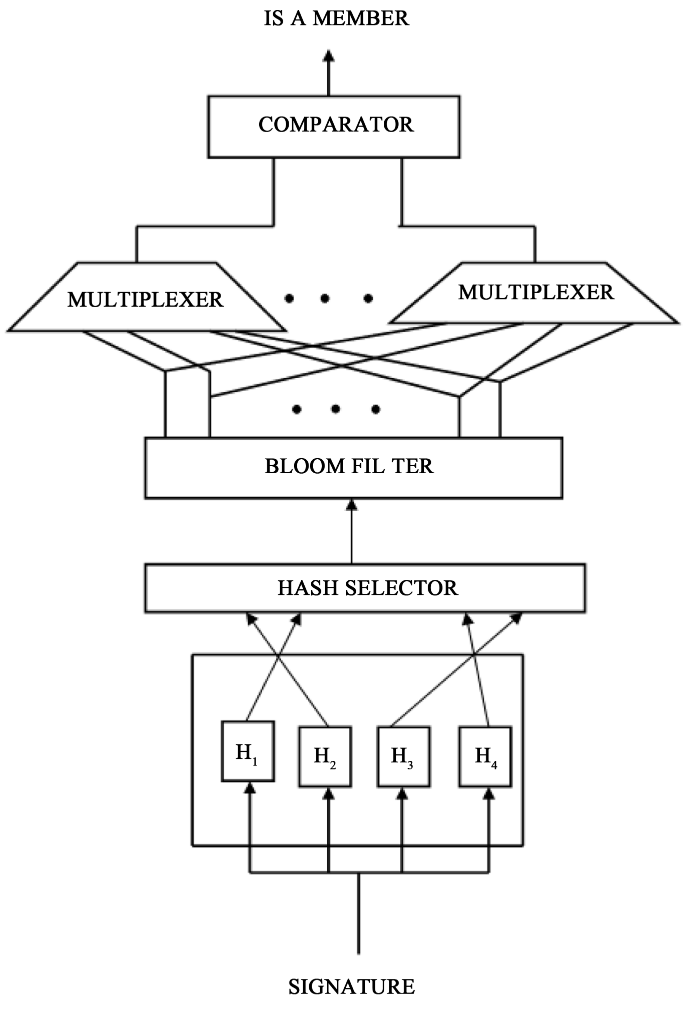

Figure 2. Multi hash function architecture.

5. Single and Multi-Level Ranking Scheme

5.1. Single Level Ranking Scheme for Bloom Filter and Counting Bloom Filter

The Single Level Ranking scheme is performed as shown above, using the single array of counter for maintaining the hash entries. The multi hash function is performed for the element X and its hash values are stored in Bloom Filter as shown in **Figure 3**. The Single Level hash function used in CBF is shown in **Figure 4**. The false positive probability in a single level counter structure is dominated by the m, n and k. The m indicates the number of counters, the n denotes the maximum number of elements and k implies the number of hash functions [17]. The false positive rate predominantly depends on the value of m, for a fixed n and k.

5.2. Multi-Level Ranking Scheme for Counting Bloom Filter

In multi level hash technique, the memory array (counter) or vector is arranged in a multilevel configuration to reduce the lookup. This partitioning technique increases the accuracy of the counting Bloom filter. A basic hierarchical structure for multi level scheme is shown in **Figure 5**. This idea is incorporated in the multi hash CBF as depicted in **Figure 6**. In this hierarchy, the first level is used to check the membership for each query. Other levels are used to calculate the total number of times the counters got incremented for each element for insertion or decremented for deletion. The update operation in this hierarchical structure may seem to be difficult, but in reality it minimizes the false positive probability than conventional Bloom Filter.

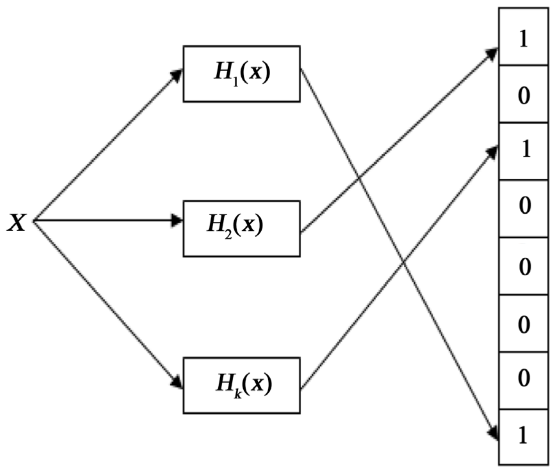

Figure 3. Single level ranking scheme for BF.

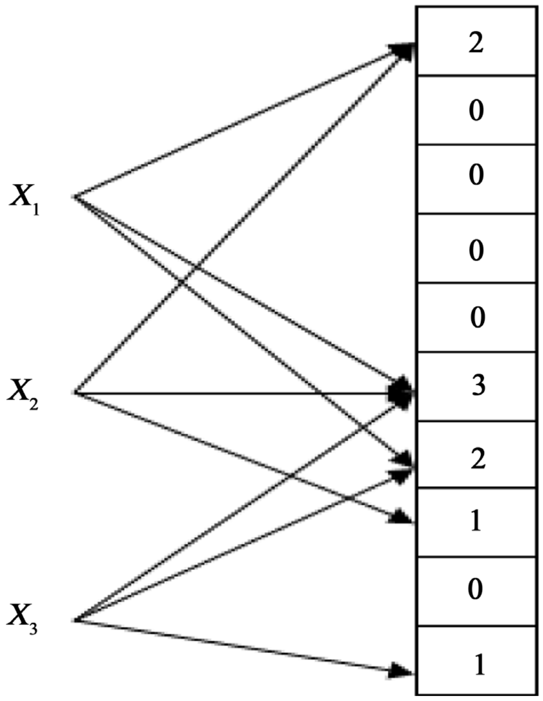

Figure 4. Single level ranking scheme for CBF.

CBF has a ranked structure which is composed of r-levels $b_1 \cdots b_r$ and an idle array b_i. The CBF uses k hash functions $h_1 \cdots h_k$ to hash an element x into k bits in the first level and is controlled by the parameter m for a given n and k. As the element insertion is on the hashing in the first level, it is enough to conduct query in the first level. Hence, in the ranked structure, b_1 is employed to substantiate membership query, $b_2 \cdots b_r$ is used to manipulate the element insertion, b_i is utilized for insertion of new signatures into the dataset. The first level b_1 has the same size as that of CBF. Since m is 10, the size of $b_1 = 10.$ The remaining 20 bits are used for idle array b_i. Assume, l_1 is the bit size of b_1.

The hashed positions are incremented by 1 initially. If the same position is indexed second time, then the counter stages are utilized instead of incrementing the same location once again. The offset index for next level is given by the number of one's present below the current location. Assume if the location 5 is hashed for the first time by an element x and upon insertion if it is hashed again by another, element say y, under this circumstance the counter should have incremented to 2 as in CBF. But in ranked structure, the first entry is made as in

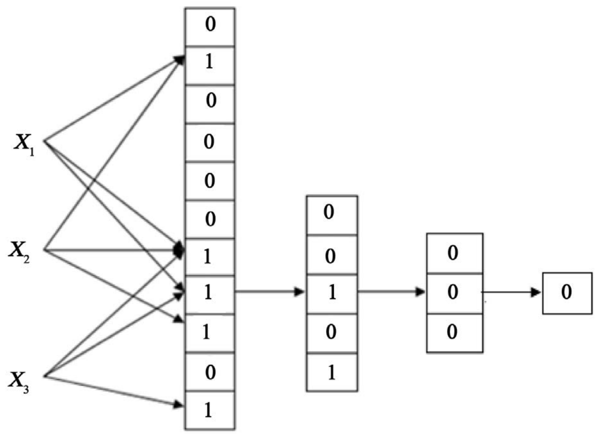

Figure 5. Hierarchical structure for multi-hash CBF.

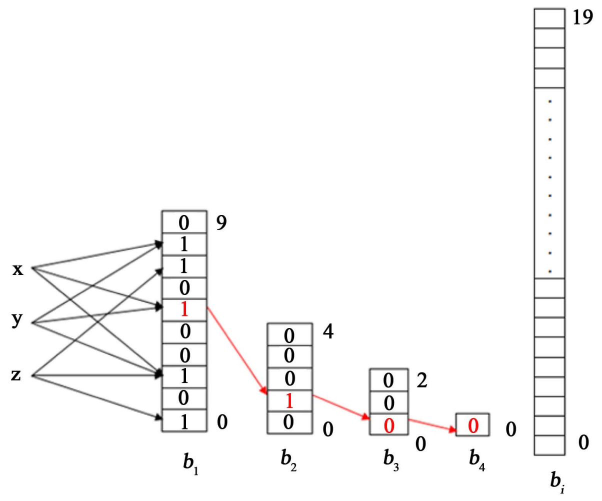

Figure 6. Multi-level ranking scheme for CBF.

CBF and for further entries the rest of the levels are involved. For second time hash entry, the entries in below the active location are counted. If it returns a value 2, then second stage b_2 is indexed at 2. As the position 5 is hashed only two elements x and y, the further levels are not involved. They remain at zero value. For query process the first level b_1 is monitored. If it returns a value zero, then the element is not a member. If it returns one, then the element is a member. Since the first level counter array is only involved in query process the throughput increases. The false positive rate of the Multi-level ranking scheme is given by Equation (9),

$$FPR = \left(1-\left(1-\frac{1}{l_1}\right)^{nk}\right)^k \approx \left(1-e^{-kn/l_1}\right)^k \tag{9}$$

where, n is the maximum number of elements; l_1 is the bit size of the first level and k is the number of hash functions. The experimental results are enumerated in the next session.

6. Experimental Result

The False Positive rate of the proposed techniques is compared with the existing techniques as shown in Table 1. The FPR for the Multihash is verified experimentally by considering 100 sigantures in dataset, length of the counter array m = 16 and k = 3. From the iteration conducted and from the theorical formula in Equation (7), 30% is reduced. For multi-level ranking scheme Equation (9) is used to obtain the FPR. From the result, the FPR is approximately 30% better compared to single hash CBF. The FPR is not affected by the Multi-Level Ranking Scheme in the proposed architecture. Table 2 illustrates that, as the number of hash function increases the number of LUTs utilized also significantly increases. As the Multi-Level scheme is introduced there is no abrupt change in the resource utilization in the proposed technique. This happens as the Multi-Level scheme is only rearranging the counter array in a Hierarchical fashion. Next most important parameter is power. As the query process is carried out only by monitoring the first level of the counter array, the switching activity reduces which in turn deduce 13% - 16% of the power. Similarly, the throughput increases for about 23% - 30%. So, this proposed Multi-Level Ranking scheme in Multi-Hash architecture will have high speed and low complexity with low FPR is well suited for signature detection.

7. Conclusion

Hardware optimized architecture entitled as Multi Hash CBF and Multi-Level scheme is explored in this paper. As single Hash CBF has high False Positive Rate, Multi Hash CBF is proposed which will check for multiple hashes to improve the FPR. But still, this proposed work has low throughput and power. To enhance a hierarchical counter array vector is employed. The simulation and implementation results depict the deduction in the

Table 1. Comparison to show the False Positive Rate of Multi Hash CBF and Multi-Level Ranking CBF.

	Existing Techniques			Proposed Techniques	
	CBF	Spectral BF	dl-CBF	Multi Hash CBF	Multi-Level Ranking CBF
False Positive Rate	10^{-3}	10^{-3}	1.5×10^{-3}	0.7×10^{-3}	0.7×10^{-3}

Table 2. Implementation result for multi hash CBF architecture and multi-level ranking CBF architecture.

	Parameters	Multi Hash CBF Architecture			Multi-Level Ranking CBF Architecture		
		1 MHz	20 MHz	100 MHz	1 MHz	20 MHz	100 MHz
K = 2	No. of 6-input LUTs	49	49	49	50	50	50
	Power (mW)	73.55	73.53	73.53	63.9	63.9	63.7
	Throughput (MHZ)	309	309	307	407	407	407
	Delay (ns)	3.236	3.236	3.257	2.457	2.457	2.457
	Power-Delay-Product (PDP)	238.01	237.9	239.49	157	157	156.5
K = 3	No. of 6-input LUTs	61	61	61	62	62	62
	Power (mW)	76.32	76.36	76.56	65.3	65.32	65.25
	Throughput (MHZ)	301	301	301	399	398	398
	Delay (ns)	3.322	3.322	3.322	2.506	2.512	2.512
	Power-Delay-Product (PDP)	253.53	253.67	254.33	163.64	163.64	163.9
K = 4	No. of 6-input LUTs	74	74	74	75	75	75
	Power (mW)	79.34	79.45	79.44	68.5	68.51	68.5
	Throughput (MHZ)	291	290	286	387	387	387
	Delay (ns)	3.436	3.448	3.496	2.583	2.583	2.583
	Poer-Delay-Product (PDP)	272.61	273.94	277.72	176.93	176.96	176.93
K = 5	No. of 6-input LUTs	89	89	89	91	91	91
	Power (mW)	83.12	83.11	83.2	72.6	72.6	72.6
	Throughput (MHZ)	284	284	279	365	365	363
	Delay (ns)	3.52	3.52	3.58	2.739	2.739	2.754
	Power-Delay-Product (PDP)	292.58	292.54	297.85	198.85	198.85	199.94

13% - 16% of the power and shows better performance in terms of throughput and PDP. Further, the sidon sequence can be employed in this proposed architecture to enhance the FPR. The sidon sequence provides variable increment for the counter, which helps in reducing the FPR. The proposed architecture is well suited for signature matching applications.

References

[1] KPMG: Cybercrime Survey Report (2015) KPMG India. 1-24. https://www.kpmg.com/IN/en/IssuesAndInsights/ArticlesPublications/Documents/Cyber-Crime-Survey-2015-30Nov15.pdf

[2] Aho, A.V. and Corasick, M.J. (1975) Efficient String Matching: An AID to Bibliographic Search. *Communication ACM*, **18**, 333-340. http://dx.doi.org/10.1145/360825.360855

[3] Lin, C.-H. and Chang, S.-C. (2011) Efficient Pattern Matching Algorithm for Memory Architecture. *IEEE Transaction on VLSI System*, **19**, 33-41. http://dx.doi.org/10.1109/TVLSI.2009.2028346

[4] Boyer, R. and Moore, J. (1987) A Fast String Searching Algorithm. *Communication ACM*, **20**, 762-772. http://dx.doi.org/10.1145/359842.359859

[5] Barkalov, A. and Titarenko, L. (2008) Logic Synthesis for Compositional Microprogram Control Units. Springer Verlag, Berlin.

[6] Spitznagel, E.W. (2005) High Performance Packet Classification. PhD Thesis, Washington University, Washington DC.

[7] Fisk, M. and Varghese, G. (2001) Fast Content-Based Packet Handling for Intrusion Detection. UCSD Tech. Report, CS2001-0670.

[8] Coit, C.J., Staniford, S. and McAlerney, J. (2001) Towards Faster String Matching for Intrusion Detection or Exceeding the Speed of Snort. *Proceedings of the DARPA Information Survivability Conference and Exposition*, **1**, 367-373. http://dx.doi.org/10.1109/DISCEX.2001.932231

[9] Gokhale, M., Dubois, D., Dubois, A., Boorman, M., Poole, S. and Hogsett, V.G. (2002) Towards Gigabit Rate Network Intrusion Detection Technology. *Proceedings of the* 12*th International Conference on Field Programmable Logic and Applications*, **2438**, 403-413.

[10] Tan, L. and Sherwood, T. (2005) A High Throughput String Matching Architecture for Intrusion Detection and Prevention. *Proceedings of the* 32*nd Annual International Symposium on Computer Architecture*, Washington DC, 4-8 June 2005, 112-122. http://dx.doi.org/10.1109/ISCA.2005.5

[11] Jung, H.J., Baker, Z.K. and Prasanna, V.K. (2006) Performance of FPGA Implementation of Bit-Split Architecture for Intrusion Detection Systems. *Proceedings of the* 20*th IEEE International Parallel & Distributed Processing Symposium*, Rhodes Island, 25-29 April 2006. http://dx.doi.org/10.1109/IPDPS.2006.1639434

[12] Bloom, B.H. (1970) Space/Time Trade-Offs in Hash Coding with Allowable Errors. *Communication ACM*, **13**, 422-426. http://dx.doi.org/10.1145/362686.362692

[13] Broder, A. and Mitzenmacher, M. (2005) Network Applications of Bloom Filters: A Survey. *Internet Mathematics*, **1**, 485-509.

[14] Brindha, P., Senthilkumar, A. and Raja, R. (2014) A State-of-Art Survey on Bloom Filter in Network Applications. *International Journal of Applied Engineering Research*, **10**, 29002-29007.

[15] Fan, L., Cao, P., Almeida, J. and Broder, A.Z. (2000) Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol. *IEEE/ACM Transactions on Networking*, **8**, 281-293. http://dx.doi.org/10.1109/90.851975

[16] Ramakrishna, M.V., Fu, E. and Bahcekapili, E. (1994) A Performance Study of Hashing Functions for Hardware Applications. *Proceedings of the International Conference on Computing and Information*, 1621-1636.

[17] Brindha, P. and Senthilkumar, A. The Veracious Counting Bloom Filter. The International Arab Journal of Information Technology. (In Press)