Global Journals LaTeX JournalKaleidoscope™
Artificial Intelligence formulated this projection for compatibility purposes from the original article published at Global Journals. However, this technology is currently in beta. *Therefore, kindly ignore odd layouts, missed formulae, text, tables, or figures.*

Comparative Study of Bloom Filter Architectures

Dr. Ritu chhabra[1]

[1] GGSIPU

Received: 14 February 2012 Accepted: 3 March 2012 Published: 15 March 2012

Abstract

Hardware based virus protection systems are required for identifying the malicious content and further removing it from network streams. Network Intrusion Detection System(NIDS) is needed to protect the end user machines from threats. An effective NIDS is therefore a network security system capable of protecting the end user machines well before a threat affects.NIDS requires a space efficient data base for detection of threats in high speed conditions. Bloom Filters are one of the security filters that consume significant power to detect and then filter out malicious content. A Bloom filter is a space efficient randomized data structure for representing a set in order to support membership queries. The aim of this paper is to compare the different architectures of Bloom filter like Standard Bloom filter, pipelined bloom filter, counting Bloom filter and parallel processing architecture of bloom filter in terms of their merits and demerits by using algorithmic architectural techniques.

Index terms— Bloom filters, network intrusion detection, universal hash function, FPR (False positive rate)

1 INTRODUCTION

s the usage of portable devices continues to increase, more and more user applications catering to these device platforms are being developed. Also more often than not such devices are connected to one or more communications networks and must process a significant amount of incoming data. It is therefore becoming increasingly essential to secure these devices from malware of all kinds. The traditional approach for solving this problem on desktop computers is to provide for specialized antivirus software, firewall software and more recently antispyware software etc. However as opposed to a desktop computer the limited amount of computational power packed into a small footprint portable device precludes the use of resource intensive security software. It is therefore necessary to provide for alternate means to perform these functions on a small footprint device. Central to the ability to detect a malicious piece of code, a malicious packet in a data stream etc. is the ability to quickly determine if a given string of tokens belongs to a dictionary of known signatures. If we think of this dictionary of known signatures as a set, we have essentially reduced the problem of detecting malware to a problem of resolving set membership [1]. In this paper we give a recent survey on different types of bloom filter used for network Intrusion Detection system to benefit the research community to analyze and develop an efficient Bloom Filter which can have a prominent role in Network security, each having its own merits and demerits. The details of standard Bloom filter, pipelined Bloom filter, parallel processing bloom filter, counting Bloom filter is explained below. This paper also present the hardware architectures for the implementation of different Bloom filters. A Bloom Filter is a data structure that stores a given set of signatures by computing multiple hash functions on each member of the set and testing strings for membership of that set [3]. It acts as hardware antivirus device and connected with the CPU to remove the malicious input data. It consists of a set of hash functions, a hash function buffer to store hash results temporarily, a look up array to signify hash values and a decision component made of an AND to test the membership of testing string as shown in Fig. 1.

2 STANDARD BLOOM FILTER ARCHITECTURE

Standard Bloom filter is an important and widely used tool for supporting efficient query services in networking because of its ability to represent a set of items by using a bit array with several independent hash function [6].Bloom filter provide an effective tool for saving the space when space is at a premium. For pattern matching Bloom filters are used. They are hashed based structures which have a certain degree of accuracy for considerable savings in memory. Two basic operations are defined for Bloom Filter. First is programming for programming the look up array using hash functions of strings in data set and second is testing for checking the membership of test string [2]. A class of universal hash functions described here found to be suitable for hardware implementation. Following is a description of how this hash matrix is calculated [2,3,7].Given dataset of inputs X= {X 1 , X 2 , X 3 ???X n } Each input is of b bits X j = {x 1 , x 2 , x 3 ????x b } ith hash function over string Xj is given in eqn. (2.1) 1 1 2 2

(X) . ^. ^.....^.i j i i i b b H d x d x d x (2.1)

H i (X j) is the ith hash function of jth input string of input set ij d is a random coefficient ranging 1 to m j x 's are the bits in particular input string Where '.' is a bitwise AND operator, i.e.

3 PIPELINED BLOOM FILTER

In some application such as network intrusion detection due to very low rate of malicious traffic there is no need to compute all the hash functions to get a result of non membership. To exploit this pipelined architecture is intruduced. Pipelined architecture of bloom filter consists of several group of hash function that are utilized in different stages.The first stage always compute the hash values. The second and further stages are used only if there is a match in the previous stage [3].

4 Advantage of Pipelined Bloom Filter

The advantage of using a pipelined Bloom filter is if the first stage pruduces a mismatch there is no need to use the second satge in order to decide whether the input string is a member of signature set because a bloom filter never pruduces a false nagatives. This saves the power consumed by pipelined bloom filter as compared to the standard bloom filter.

5 Draw Back of pipelined Bloom filter

Power saving ratio diminishes when there are high no of matches in the first stage and second stage is utilized more.To remove this problem we use fully pipelined architecture of Bloom filter [4]. Pipelined architecture of bloom filter minimizes the false positive probability because first stage utilize more no of hash function increases the probability of mismatch thus second stage is not utilized but more no of hash function consume the more power which becomes the drawback of this architecture. Hence fully pipelined architecture remove this drawback as each of its stage has only one hash function. Fully pipelined architecture has the same no of hash function as the regular bloom filter hence its false positive probability is same as the regular bloom filter [3].

6 IV. PARALLEL PROCESSING ARCHITECTURE

A standard bloom filter architecture can effectively represent the items with a single attribute but it cannot support the representation and querying of

7 COUNTING BLOOM FILTER

One property of Bloom filter is that it is not possible to delete a member stored into the filter. Deleting a particular entry requires that the corresponding k hashed bits in the bit vector be set to zero. This could disturb other members programmed into the filter which hash to any of these bits. In order to solve this problem, the idea of the Counting Bloom Filters was proposed in. A Counting Bloom filter maintains a vector to counters corresponding to each bit in the bit-vector. Whenever a member is added to or deleted from the filter, the counters corresponding to the k hash values are incremented or decremented respectively. When a counter changes from 0 to 1, the corresponding bit in the bit-vector is cleared. It is important to note that the counters are changed only during addition and deletion of strings in a Bloom filter. For applications like network intrusion detection, these updates are relatively less frequent than the actual query process itself [6]. Architecture of counting bloom filter is shown in fig. **??** Fig. **??** : Counting Bloom Filter

8 VI. CONCLUSION

A Bloom Filter can be used in variety of Network applications. Different architectures of Bloom filter have been described in terms of merits and demerits. Pipelined architecture of bloom filter minimizes the false positive probability because first stage utilize more number of hash function increases the probability of mismatch thus second stage is not utilized but more number of hash function consume the more power which becomes the

1

Figure 1: Fig. 1 :

2

Figure 2: Fig. 2 .

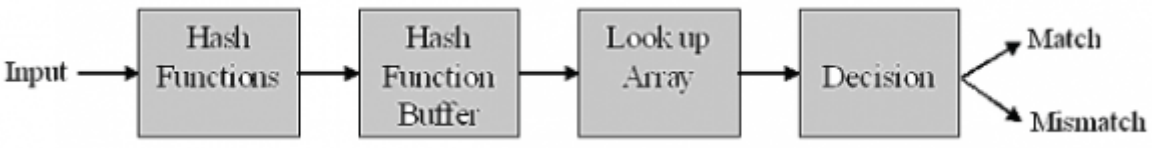
2

Figure 3: Fig. 2 :

Figure 4:

3

Figure 5: Fig. 3 :



4

Figure 6: Fig. 4 :

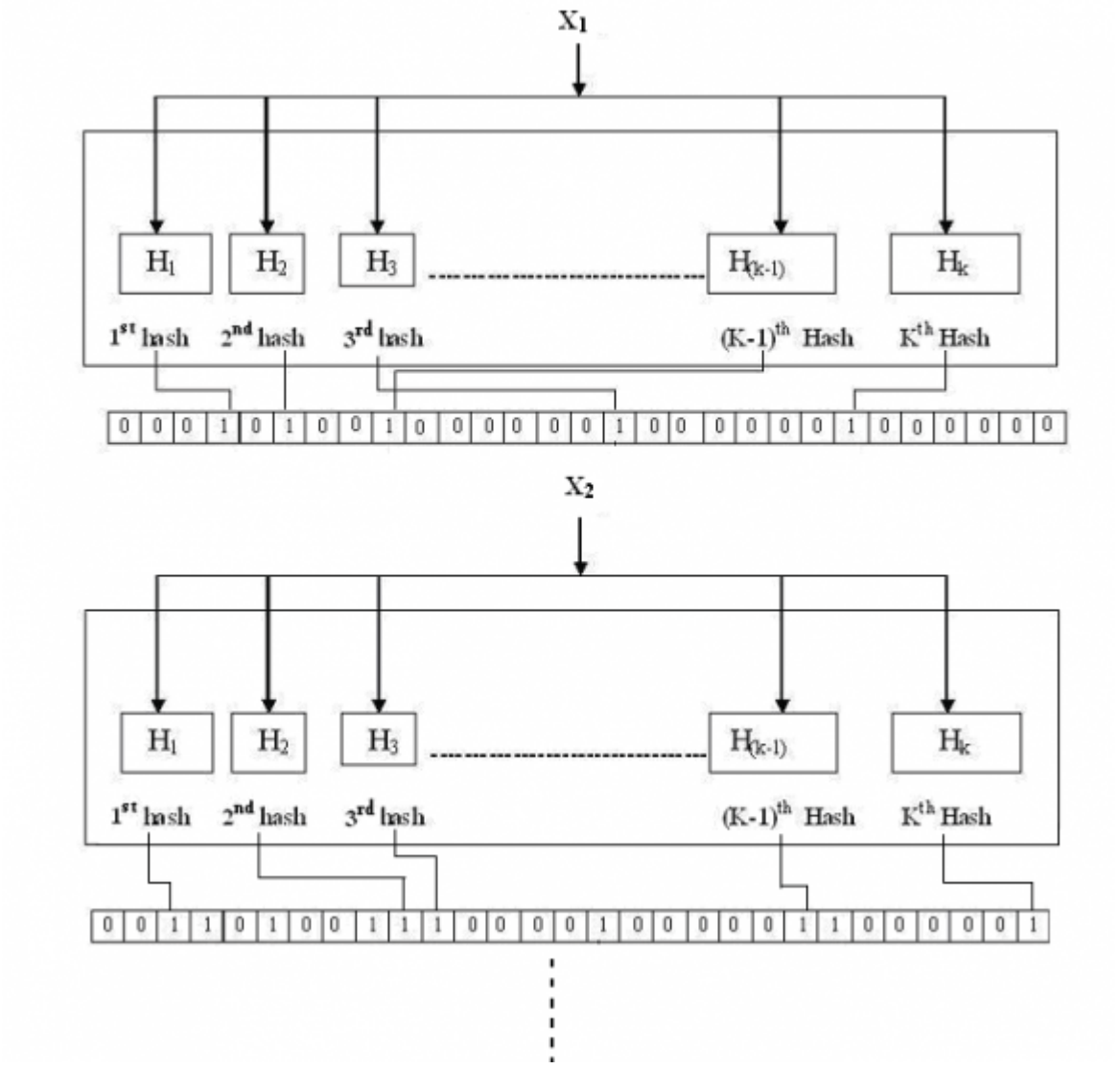
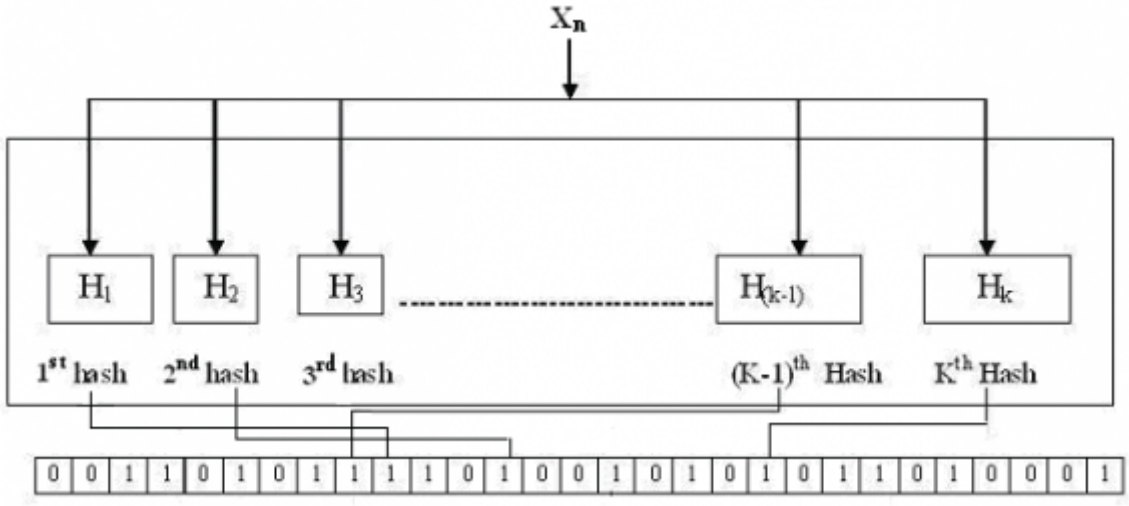


Figure 7:



5

Figure 8: Fig. 5 :

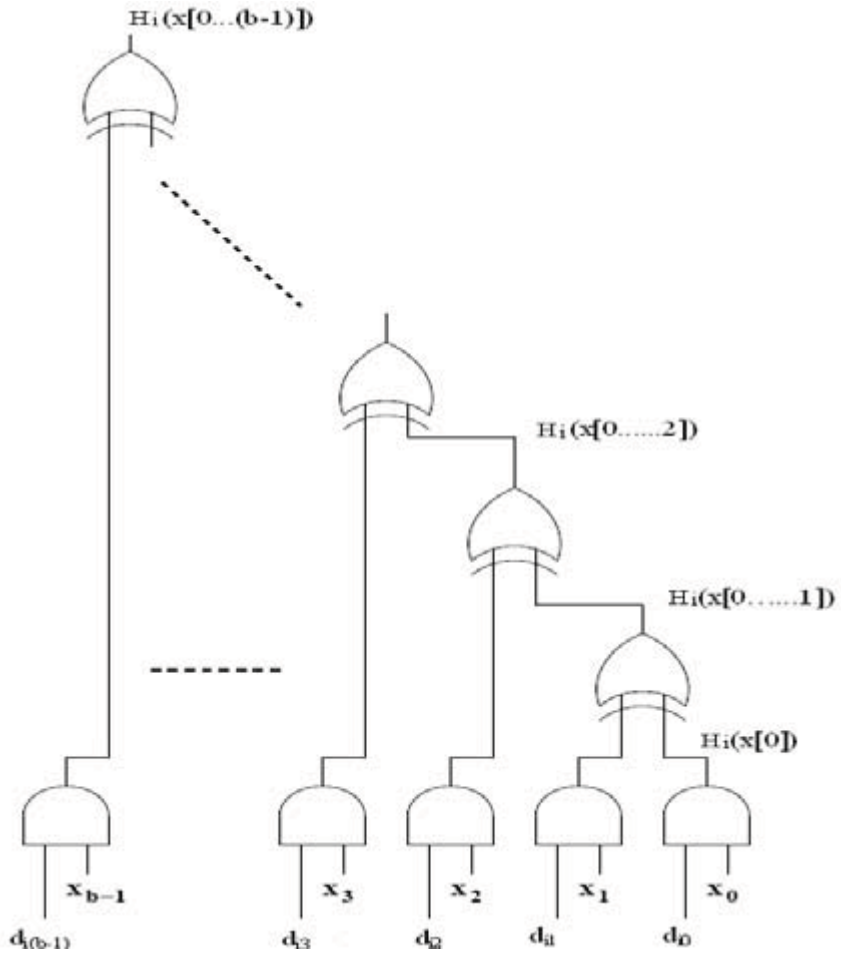
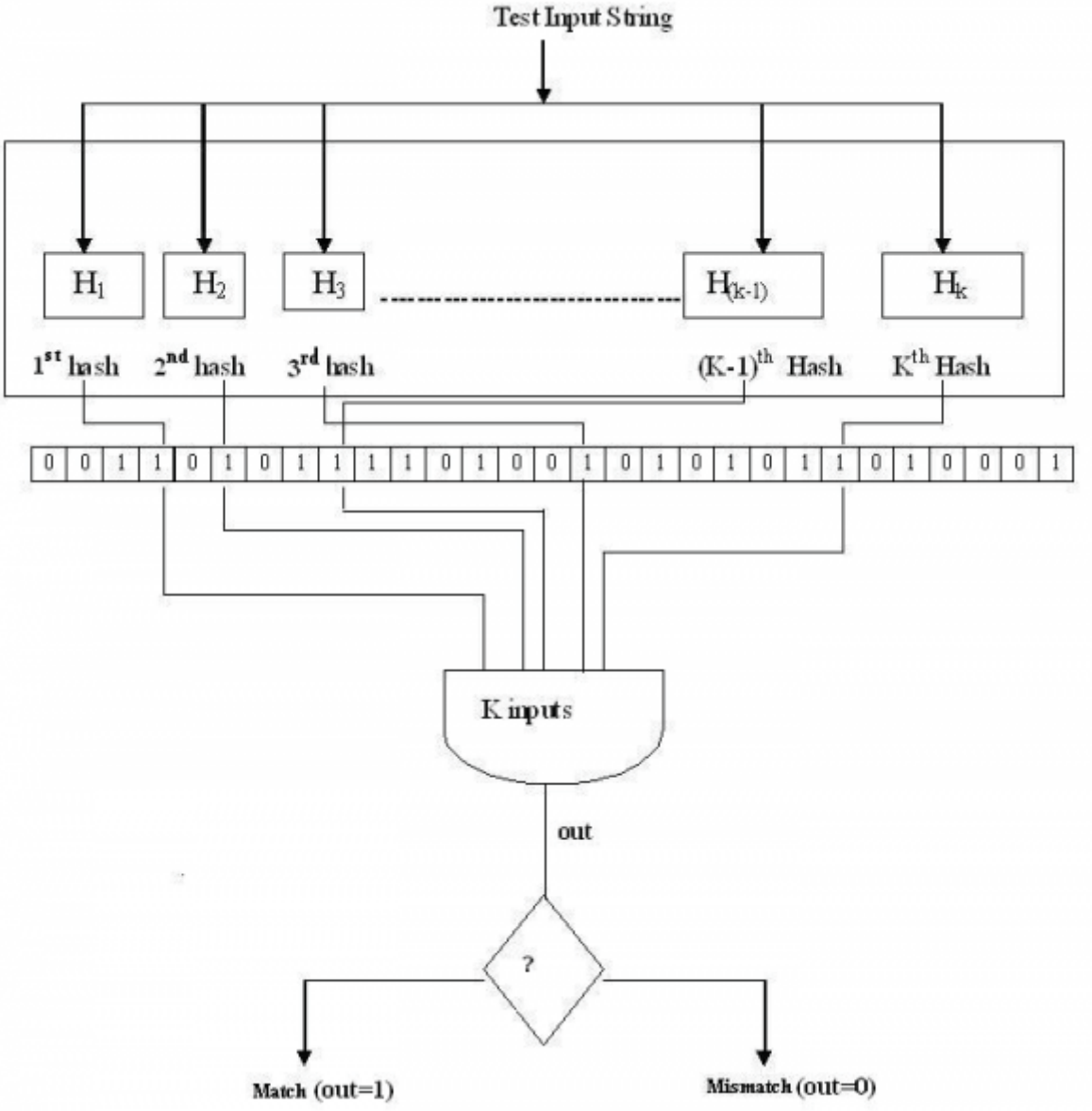


Figure 9:



6

Figure 10: Fig. 6 :

drawback of this architecture. Hence fully pipelined architecture remove this drawback as each of its stage has only one hash function.counting bloom filter remove the problem of deleting a member stored in filter. [1]

[Saravanam] , K Saravanam , Dr .

[Kaya and Kocak ()] 'A low power lookup technique for multi-hashing network applications'. T Kaya , Kocak . *Proc. IEEE Annual Symp. on VLSI (ISVLSI)*, (IEEE Annual Symp. on VLSI (ISVLSI)Karlsruhe; Germany) 2006. p. .

[Kumar and Dolian (2011)] 'A Recent Survey on Bloom Filters in Network Intrusion Detection Systems'. A Kumar , J Dolian . *International Journal on Computer science and Engg* Mar 2011. 3 (3) . (References Références Referencias)

[Dharmapurikar et al. ()] 'Deep packet inspection using parallel Bloom filters'. S Dharmapurikar , P Krishnamurthy , T S Sproull , J W Lockwood . *IEEE Micro* 2004. 24 (1) p. .

[Paynter and Kocak (2008)] 'Fully Pipelined Bloom Filter Architecture'. Michael Paynter , Taskin Kocak . *IEEE Communications Letters* November 2008. 12 (11) p. .

[Kaya and Kocak ()] 'Increasing the Power Efficiency of Bloom Filters for Network String Matching'. Ilhan Kaya , Taskin Kocak . *P Proc.IEEE International Symposium on VLSI*, 2006. p. .

[Ahmadi and Wong ()] 'K-stage Bloom Filter Architecture'. Mahmood Ahmadi , & Stephan Wong . *International conference on Computer science and Engg*, 2009.

[Kaya and Kocak ()] 'Low power Bloom filter architecture for deep packet inspection'. Ilhan Kaya , Taskin Kocak . *IEEE Commun. Lett* 2006. 10 (3) p. .

[Bloom ()] 'Space/time trade-offs in hash coding with allowable errors'. B Bloom . *Commun. ACM* 1970. 13 (7) p. .

[Xio and Yuhua (2010)] 'using Parallel Bloom Filters for multiattribute Representation on Network Services'. Bin Xio , & Yuhua . *IEEE Transaction on Parallel and Distributed Systems* January 2010. 21 (1) .