Malicious Host Detection using Probabilistic Data Structures

A Thesis

submitted in partial fulfilment of the requirements for the award of the degree of

Master of Engineering

in

Computer Science and Engineering

by

Divya Gupta
(Roll No: 801532013)

Under the supervision of

Dr. Shalini Batra
(Associate Professor)

Computer science and Engineering Department
THAPAR UNIVERSITY
PATIALA-147004, PUNJAB, INDIA
June 2017

CERTIFICATE

I hereby certify that the work, which is being presented in the thesis, entitled "*Malicious Host Detection using Probabilistic Data Structures*", in partial fulfilment of the requirements for the award of the degree of *Master of Engineering* in *Computer Science and Engineering* submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of *Dr. Shalini Batra* and refers other researcher's work which are duly listed in the reference section.

The matter presented in the thesis has not been submitted for award of any other degree of this or any other University.

Divya Gupta

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

Dr. Shalini Batra
Associate Professor, CSED

Acknowledgements

First, I would like to express my deep gratitude to my supervisor **Dr. Shalini Batra** for her invaluable advice and encouragement at every step of my ME program. Without her unfailing support and belief in me, this thesis would not have been possible. Her contribution to this thesis goes well beyond her role as an academic supervisor and includes constant support on personal level too without which this journey may never have been completed. And for this, I am truly grateful. She is a great mentor for my life as well.

I would like to express my gratitude to our director **Prof Prakash Gopalan**, **Dr. Maninder Singh** head of CSED and **Dr. Ashutosh Mishra** PG coordinator for their constant motivation and encouragement. I also wish to thank the my research committee members and non-teaching staff of the institute for their help and support. I would also like to thanks to my Meditation guide **Mrs. Veena Gopalan** and friend from whom I learn the art of happiness and never give up approach.

I would like to give special acknowledgement to Ph.D scholars Mr. Amritpal Singh for providing moral support.

Finally, I would like to express my sincere and deep gratitude to my parents and family member for their love, encouragement, care and support.

Divya Gupta

Abstract

Internet is integrated platform where data is continuously increasing at an exponential rate. Since internet is lifeline of various business and personal activities and a growing number of users access all kind of data, there is an utmost requirement of protecting such data from illegal access or modification. To protect data from emerging attacks, a wide range of methods have been proposed in the literature. Intrusion detection systems are considered as one of the important tool for monitoring and analysing network traffic to protect against emerging attacks. In this work a novel method of intrusion detection is presented. In the proposed method a popular Probabilistic Data Structure (PDS) Bloom filter is employed to store information of suspicious nodes which reduces the storage requirement.
Further, Modified Count Min Sketch (MCMS), a variant of Count Min Sketch (CMS), a PDS used for frequency count is used to track hit rate of suspicious nodes in a defined time span. The work provides a detailed analysis of the proposed scheme and the output achieved shows that proposed approach is more efficient compared to CMS since the results obtained indicate that MCMS require less storage and computational time as compared to CMS.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

PDS	Probabilistic Data Structure
IDS	Intrusion Detection System
BF	Bloom Filter
CMS	Count Min Sketch
MCMS	Modified Count Min Sketch
FP	False Positive
FN	False Negative
NIDP	Network Intrusion Detection and Prevention
TP	True Positive
FPR	False Positive Rate
FNR	False Negative Rate

Chapter 1

Introduction

With the advancement in information technology and usage of internet in almost all kinds of business and personal activities, huge amount of data is being generated in various formats within hours. This network usage is leading to increased vulnerabilities as the number of attacks are also increase rapidly. Such increase is posing a serious challenge to data handling, storage complexity and computational cost. Attacker try to illegally access or modify sensitive data. In network any malicious attack can disturb the whole network process or block it temporarily; hence intrusion detection is required to detect and prevent network from malicious users. Intruders can attack computer network in number of ways for *e.g.* one can spoof various fields of message or alter behaviour of a node in the network or intentionally create a fault in network applications, *etc.*

1.1 Intrusion Detection System (IDS)

Intrusion Detection: A process of monitoring computer and network events and analysing them for detecting vulnerabilities and alert network administrator at any point for unauthorised intrusion is called intrusion detection.

An IDS is designed to monitor and analyse all network activities to identify any suspicious pattern indicating a network attack. For identifying exploits, attacks, probs and other vulnerabilities, IDS monitors network traffic and data. IDS includes following functions:

- Monitor and analyse user and system activities
- Inspect system configurations and vulnerabilities
- Identify pattern of attacks
- Inspect abnormal events
- Assessing system and file integrity

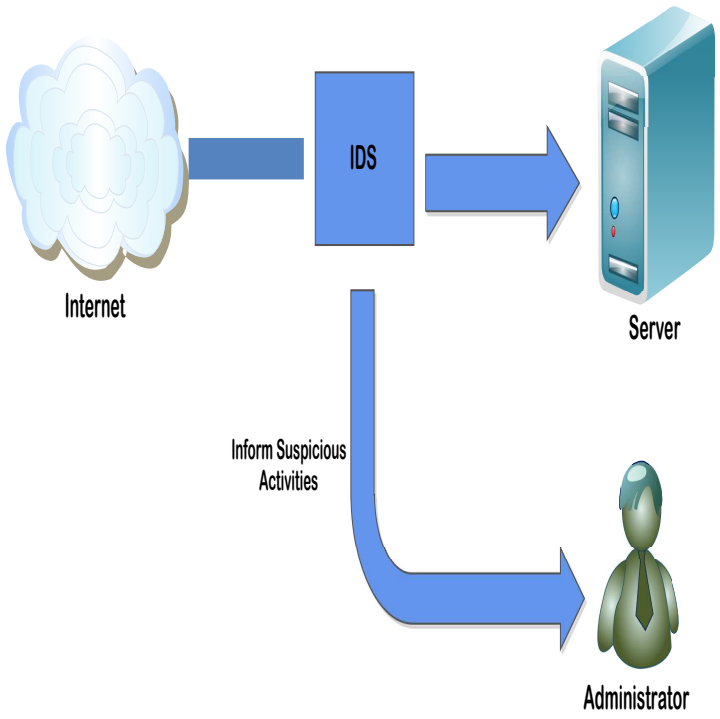

Figure 1.1: IDS

Based on detection techniques, IDS are classified as anomaly-based and misuse-based (also referred to as signature based). In anomaly based IDS, machine learning techniques are utilized to define baseline, protocol, state of the network and typical size of the packet (to define profile of normal behaviour of system). Based on this defined behaviour incoming traffic is classified into normal or abnormal behaviour. Anomaly based IDS has capability to detect new types of attack. However, they result in slow speed and high false positive(FP).
In signature based IDS a large database of attack signatures is maintained and an exact pattern matching algorithm is employed to match incoming packet to the attack signature database. Signature based IDS are faster, more accurate and have low false positives. Since they look up for a specific attack signature which has been already documented they are unable to detect new emerging attacks that have not been documented yet.
Pattern matching algorithm is core of the signature-based IDS. Research has shown that it is a most computationally large task of IDS [1] and between 30%-60% of total processing time of IDS is spent on it.

Broder *et al.* suggested that probabilistic data structures (PDS) are computationally fast and memory efficient, useful for processing of big data. PDS are based on probabilistic approaches and approximation principles along with hashing techniques for fast processing of data. Some commonly used PDS are shown

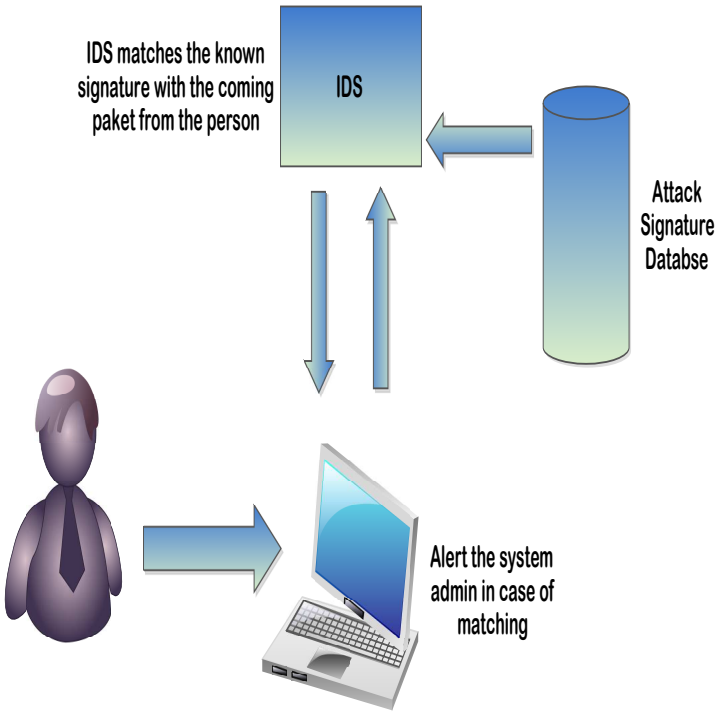

Figure 1.2: Signature-Based IDS

in Figure 1.3. Bloom Filter (BF), one of the well known PDS is memory efficient data structure. This feature of BF makes it a quite popular data structure, used in many applications. BFs are used in network security applications such as NIDP [2], [3], worm detection [4], [5], virus scanning [6] *etc.* Number of variants of IDS based on BF have been proposed in [7], [8], [9] . We propose a novel malicious detection technique based on Bloom Filter and Count Min Sketch.

1.2 Bloom Filter

Bloom Filter(BF) is a memory efficient PDS introduced by Burton Bloom in 1970 [10]. It is used to store a set $S = \{s_1, s_2, \cdots, s_n\}$ of N elements and supports set membership testing. It stores a set S of N elements in a compact probabilistic way, that may result in false positive (FP) (when item $s_1 \notin S$, claiming item $s_1 \in S$), but never result in false negative (FN) (resulting element $s_1 \notin S$, when it was inserted). Two basic operations: adding keys to the filter and testing set membership (querying for set membership) are supported by standard BF. Deletion of keys is not supported by basic BF. However, to support deletion of keys a number of variants have been invented such as Counting Bloom Filter [11], d-left CBF [12] and deletable BF [13] *etc.* are discussed in subsection 1.2.1. Some fea-

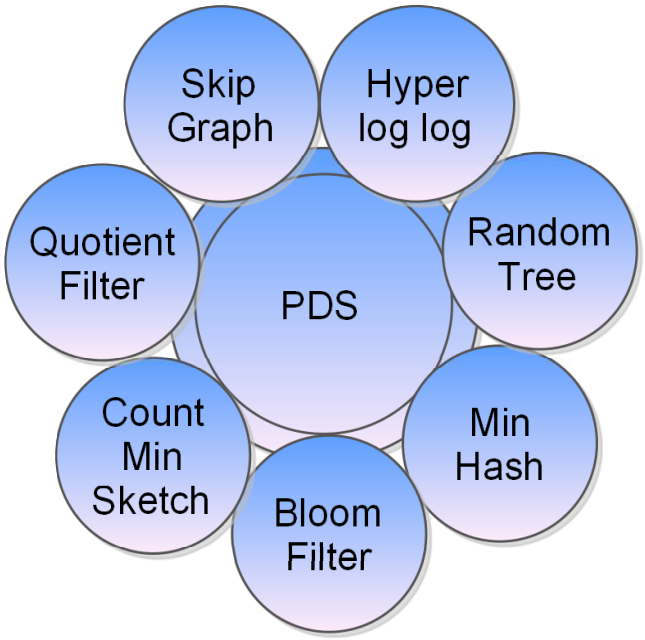

Figure 1.3: Types of Probabilistic Data Structures

tures of BF are:

- Requires less memory than required by original data set
- Time complexity for insertion and query is independent of size of set
- No false negatives
- Query time is proportional to $O(K)$
- Deletion of keys is not possible in basic BF
- Union and intersection of BF can be easily implemented with bitwise OR and AND operations.
- Can be easy divided into two equal size data sets

Figure 1.4: Initial Bloom Filter of size m

BF is an array of size m bits, $BF[1, 2, \cdots, m]$, initially all bits set to low(zero)(Figure 1.4), used to store a set $S = \{s_1, s_2, \cdots, s_n\}$ of N elements. To insert a key $s_1 \in S$, k independent hash functions are chosen that map elements uniformly random in the range $[1, 2, \cdots, m]$. For adding item $s_1 \in S$ in BF k bit indices are calculated by computing $B_i = h_1(s_1), 1 \leq i \leq k$ and all the bits at position

$B_i, 1 \leq i \leq k$ are set to high. As shown in Figure 1.5, key s_1 is inserted into BF with three hash functions (h_1, h_2, h_3) and bits at position $B_i = h_i(s_1), 1 \leq i \leq 3$ are set to high.

For set membership checking $B_i = h_i(s_i), 1 \leq i \leq k$ is computed and query results

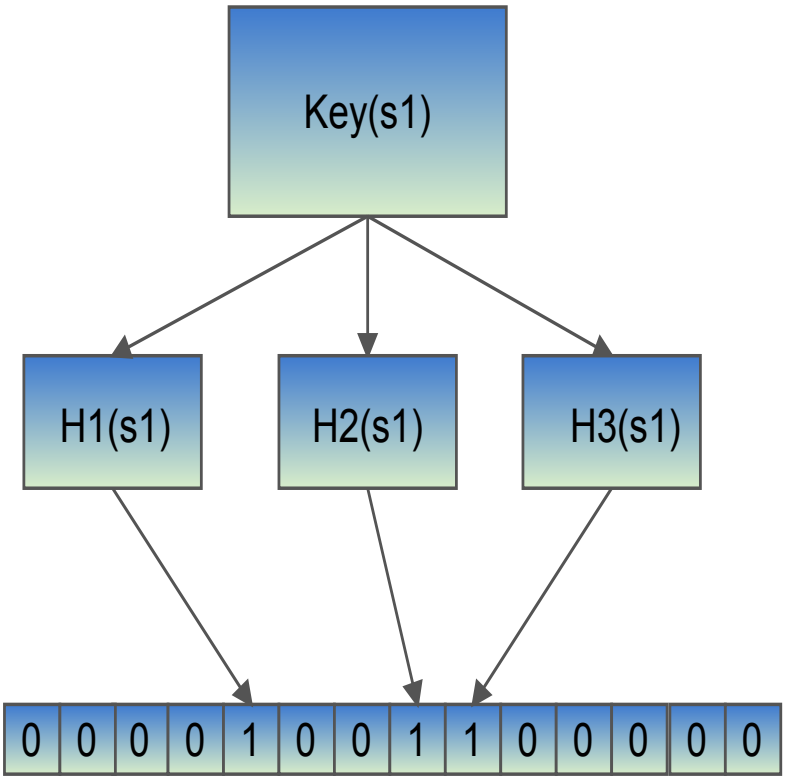

Figure 1.5: Insertion into Bloom Filter

$s_1 \in S$ if all bits at position $B_i = h_i(s_1)$ are high, if any bit at position $B_i, 1 \leq i \leq k$ is low than query results in negation *i.e.* $s_1 \notin S$. As seen in Figure 1.6 querying for key s_1 indicates that s_1 is part of BF *i.e.* query results is true positive. There is a possibility of FP (due to hash collision), i.e. there is a possibility that two item s_1 and s_2 have same calculated bit indices and $s_1 \in S, s_2 \notin S$, in such case filter gives the output $s_2 \in S$, leading to generation of false positives.

A BF with size m bit, k hash function and set cardinality $|S| = n$ has $O(k)$ time complexity for both insertion and query operations with probability of False Positive Rate (FPR):

$$p = (1 - (1 - \frac{1}{m})^{kn})^k \approx (1 - e^{\frac{-kn}{m}})^k \tag{1.1}$$

FPR depends on size of BF (m), number of hash functions(k) and number of elements inserted into BF(N). As the size of BF increases, FPR decreases and FPR increases with increase in number of elements. Acceptable error rate for the

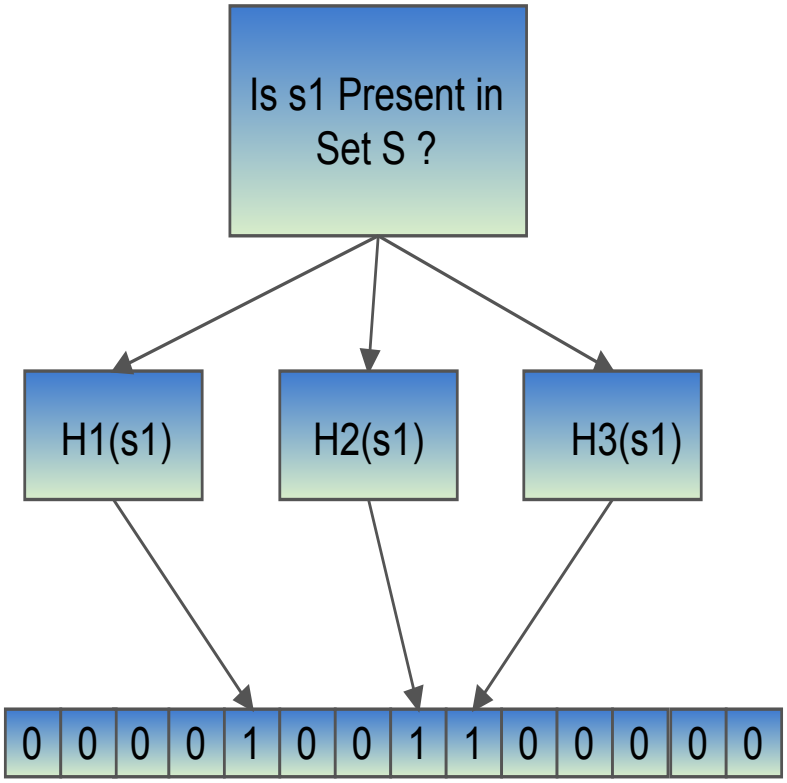

Figure 1.6: Membership Testing

application using BF decides the space advantage of BF. For maintaining a fixed FPR with n keys, value of m must be:

$$m = -\frac{n \ln p}{(\ln 2)^2} \tag{1.2}$$

1.2.1 Variants of BF

Some of the variants of BF which support removal of elements or improve efficiency of BF are discussed below:

1.2.1.1 Counting Bloom Filter (CBF)

Concept of CBF was firstly conceived by Fan *et al.* [11] Standard BF are not capable of element removal, CBF is a modified construct of standard BF which provides the element removal feature. In this modification, counter is added for each element of the data structure to solve the purpose. In the proposed CBF along with m bits, m counters are associated. Working of this structure is similar to standard BF; except that along with insertion it also supports deletion of

elements. In a CBF, with every element a small counter is corresponded with a bit of standard BF. When ever an element is added, the value of corresponding counters are incremented by one; when an element is removed from filter, the corresponding counters are decremented by one. For avoiding counter overflow, size of counter chosen should be sufficiently large. It is observed that for majority of applications, counter of 4 bits is sufficient.

1.2.1.2 d-left Counting Bloom Filter

A probabilistic data structure introduced by Bonomi *et al.* [12] is based on d-left hashing technique and fingerprints that has functionality similar to a CBF, but is more space efficient (reduces space requirement by a factor of two or more). Capacity of each bucket is c cells, along with a counter fingerprint of each element stored in it, hence to store fingerprints each cell has some fixed bit size. When a key is inserted, d independent hash values of the element are calculated to obtain d-candidate buckets. For a bucket that contain smallest number of elements, a hash-based fingerprint $f_y = H(y)$ is stored. If two or more bucket have same number of elements then according to d-left hashing scheme, data item is stored in the bucket of the leftmost subtable. To search an element, parallel d subtables are scanned to find the fingerprint and value of the counter. If deletion is performed then counter is decremented by one.

1.2.1.3 Compressed Bloom Filter

Compressed Bloom filter is particularly useful to enhance performance of a BF when a BF is used to send a message between distributed nodes. While transmission of information is required again and again and bandwidth of channel is a factor, this structure becomes very useful. To optimize size of the filter that is transmitted over the network, compressed Bloom are used. Compressed Bloom filter is particularly useful for applications such as P2P information sharing and Web caches. Use of Compressed Bloom filter reduces the number of bits broadcast, the FPR and computation amount per lookup. Some additional compression algorithm is required by this structure.

1.2.1.4 Deletable Bloom Filter

To resolve the issue of removing elements without the occurrence of false negatives, Deletable Bloom filter was introduced by Rothenberg [13]. It is optimal in terms

of storage cost as compared to Counting Bloom filter. In this variant, whenever collision occurs, the bit region is recorded and ensured that whenever one of the bit of element is reset, element is removed. In this structure bit matrix of size m is divided into number of regions (say r). Collision information is kept in a compact way in a bitmap of size r in which value 0 indicates that region is collision-free (*i.e.*, deletions of bit is permitted in this region) and value 1 indicates that the region is not collision free.

1.2.1.5 Hierarchical Bloom Filter

For supporting substring matching, Hierarchical Bloom filter was introduced by Shanmugasundaram *et al.* [14]. This structure provides facility for checking of a part of string. The basic idea of this filter is that input string is splitted into number of fixed-size blocks. Standard Bloom filter are used to insert these splitted blocks that will provide block size granularity for checking of substring. When substring matching is performed it may report false positives.
Accuracy of substring matching is enhanced through Hierarchical Bloom filter with separate insertion of blocks in it. Space needed by this structure is high as more number of blocks are added to the structure. In this structure, collection of standard Bloom filter is used for increasing block size. When a new string is added into the structure, it is first split into blocks and then starting from the lowest level of hierarchy, blocks are added into the filter. On the next level, firstly concatenation of two successive blocks is done and then added in the next level (*i.e.*, second level) of hierarchy. Same process is continued for the remaining levels of the hierarchy. Once the above process is completed, the resulting hierarchy can be used for substring matching.

1.2.1.6 Stable Bloom Filter

Streaming applications need to process data in single pass; such type of applications can be easily accommodated by Stable Bloom filter (SBF), proposed by Deng and Rafiei [15]. In streaming applications complete data should be available in main memory in a given time window, hence more elements are required to be stored resulting in an increasing number of 1s in the bit array. In such scenarios, a situation is reached where every distinct element is reported as duplicate. To avoid this situation in stable Bloom filters, removal of some information is done. To prevent from excess capacity a random deletion operation is applied in SBF.

1.2.1.7 Adaptive Bloom Filter

An variant of Counting Bloom filter was introduced called Adaptive Bloom filter (ABF) [16]. It is particularly useful for applications that require large size counter without overflows and dynamic collision rates. An increasing set of hash functions are used to count frequency of elements. This structure works same as standard BF rather than dealing with fixed c-bit cells as in standard CBFs. The main problem is that as the filter gets filled, the estimation of the multiplicity of each key becomes less precise. Space requirement of this structure is very less and does not require estimation of multiplicity of individual key elements.

1.2.1.8 Scalable Bloom Filter

In Bloom filter maximum size of bit array is decided priori. So a variant of Bloom filter was introduced that does not require priori knowledge of set size and is named as Scalable Bloom filter. This structure is made of two or more plain Bloom filters, that supports dynamic extension of the set [17]. When threshold of one filter is reached, a new slice is added. While testing for set membership query, searching is done in each filter.

1.2.1.9 Generalized Bloom Filter

Hash function used in Generalized Bloom filter (GBF) has ability to set as well as reset bits [18]. In this variant, initial value of bit array are not necessary to be set to zero. In this approach, hash functions are divided into two groups: one in which bits are set to high and another in which bits are set to low. When collision between function h_i and g_i occurs, bits are always reset. For membership testing, all the bits corresponding to hash functions h_i are set and all bits corresponding to functions g_i are reset. If a single bit is reversed then filter reports that $element \notin S$ with high probability. If no bit is reversed, then the $element \in S$ with high probability.

1.2.1.10 Dynamic Bloom Filter

A dynamic Bloom filter (DBF) is a bit matrix of size $s * m$ in which each of the s rows is a standard BF. DBF is created repeatedly. When creation starts it consists of single BF *i.e.*, a $1 * m$ bit matrix. Capacity of initial bit vector is assumed to be n_r elements, and $n_r \leq n$.

When application is executed several keys are added in the filter, hence size of A grows. When a key is added in the DBF, an active BF in the matrix is searched. If the current cardinality of A (n) is strictly higher than the cardinality of keys recorded in filter, the filter is called active BF. In an active BF, a key is added in the filter and value of the n_r is incremented by one. If any active BF is not found during the execution, than a new row is inserted (new filter is added) and n_r value is set to one. If the k positions are set to high in one of the matrix rows, a key is considered as a member of the DBF.

1.2.2 Applications of BF

Due to space efficient and probabilistic nature BFs have been gaining much popularity and finds application in many domains of computer science. Some applications of Bloom Filter are discussed below:

1.2.2.1 General Applications

1. Spell Checkers
In the spell checking software, Bloom filters are useful to check whether a given word belongs to that particular language. For this purpose, a Bloom filter is created that stores all possible words of that language and membership of the queried word is tested in the filter and significant updates are made in the filter accordingly.

2. Longest Prefix Matching
In longest prefix matching algorithms, BF are particularly useful as they have ability for efficient exact match searches. For this application, a hash table is created that stores prefixes of all lengths. When a prefix is not a member of table, reduction in the number of look ups can be achieved using Bloom Filter.

3. Refining Web Search Result
For refinement of web search results, BF are very useful. BF are used for duplicate document detection and removing duplicate document from results given to user. In the text document, similarity detection is also done with Bloom filters. For similarity measure between two documents, bit wise AND operation is used. If result has more number of $1's$, the documents are considered as similar documents.

1.2.2.2 Networking Applications

1. Routing

If the network is represented in the form of a tree in which nodes holds resources and node requests for resource are send to nodes, Bloom filter can be used to check whether there exists any path for routing the request. False positives results in routing of request in a incorrect path resulting in backtracking of the tree. Another application is routing the request along the shortest path which requires checking that whether a replica of requested file is nearby or not. A Bloom filter is implemented for each node in the network for keeping record of all adjacent edge.

2. Network Traffic

For detecting heavy flows and network traffic, Bloom filter is applicable. Bloom filter records each packet. To keep record of flow, a counter is used in Bloom filter that keeps record of bytes traversing the router. The counter is incremented by the number of bytes in the packet. If value of counter reaches a defined filled ratio, the corresponding flow is marked as heavy flow.

1.2.2.3 Applications in Security & Database management

1. Intrusion Detection

String matching is performed in Network Intrusion Detection and Prevention Systems (IDS/IPS) to scan Internet packets for malicious content. As discussed earlier Bloom filter is quite useful for string matching. It is used to store signature of different lengths and to search for signatures in parallel. If the Bloom filter result shows that signature is present in the filter, then signature hash table is searched for identifying the exact match. BF are also used in worm detection, virus scanning, Denial of Service (DoS) prevention and network forensics.

2. Encrypted Search

For encrypted search, Bloom filters are frequently used. On the client side, firstly a BF is created that stores documents, then using an encryption algorithm, the document is encrypted and BF is send to the server. Whenever the client wants to search the document, it sends keyword to the server and the server checks the presence of the keyword in the BF. If the keyword is present in the BF, the encrypted document is returned to the client.

3. Database Applications

Bloom filters have been frequently used for managing databases. It is used to es-

timate the size of joins in databases and to speed up semi-join operations. Bloom filters are specially useful in distributed databases. For reducing overall communication load in distributed applications, information is transmitted in the form of Bloom filter between the hosts.
Currently Bloom Filter is used by various software giants:

- Google uses bloom filter in the core of their search engines.
- Oracle uses BF for performing Bloom pruning of partitions for certain queries.
- Facebook uses bloom filters for type ahead search, to fetch friends and friends of friends to a user typed query.
- For reducing disk lookups Apache HBase, Google BigTable and Apache Cassandra, and Postgresql uses BFs.
- Used by the Exim mail transfer agent (MTA) in its rate-limit feature.
- They are used by Squid Web Proxy Cache for cache digests.
- Used for detecting previously stored data (used by Venti archival storage system).
- For avoiding recommendation of articles that user read previously

Table 1.1: Summary of Bloom Filter Applications

Filter	Applications
Standard Bloom Filter	Routing and Forwarding
Counting Bloom Filter	Caching, Routing and Forwarding, Monitoring and Measuring
Compressed Bloom Filter	Peer to peer systems, Resource allocation
d-left Counting Bloom Filter	Caching, Routing and Forwarding
Hierarchical Bloom Filter	Monitoring and measuring, Substring matching, Database and Networking
Stable Bloom Filter	Streaming Application
Adaptive Bloom Filter	Caching, Routing and Forwarding
Scalable Bloom Filter	Duplicate detection, Distributed Search, Informed Search, Caching
Dynamic Bloom Filter	Distributed search, caching, shortest path distance calculation

1.3 Count Min Sketch

CMS is one of the sketch data structure invented by Cormode and Muthukrishnan in 2003 [19]. It is a simple and most popular PDS for summarizing data streams. CMS is one of the memory efficient data structure, used to compute:

- simple frequencies,
- quantiles,
- heavy hitters,
- top-K most frequent items.

CMS is a 2-D matrix of counters of depth d and width w, $CMS[1,1] \cdots CMS[d,w]$, initially all indices set to low(zero). One hash function is associated with each row (Figure 1.3), uniformly random d hash functions are chosen from a family of pair-wise independent hash functions:
$h_1 \cdots h_d : 1 \cdots h \rightarrow 1 \cdots w$
It provides small summarization of large data sets, hence called sketch. Sketch-

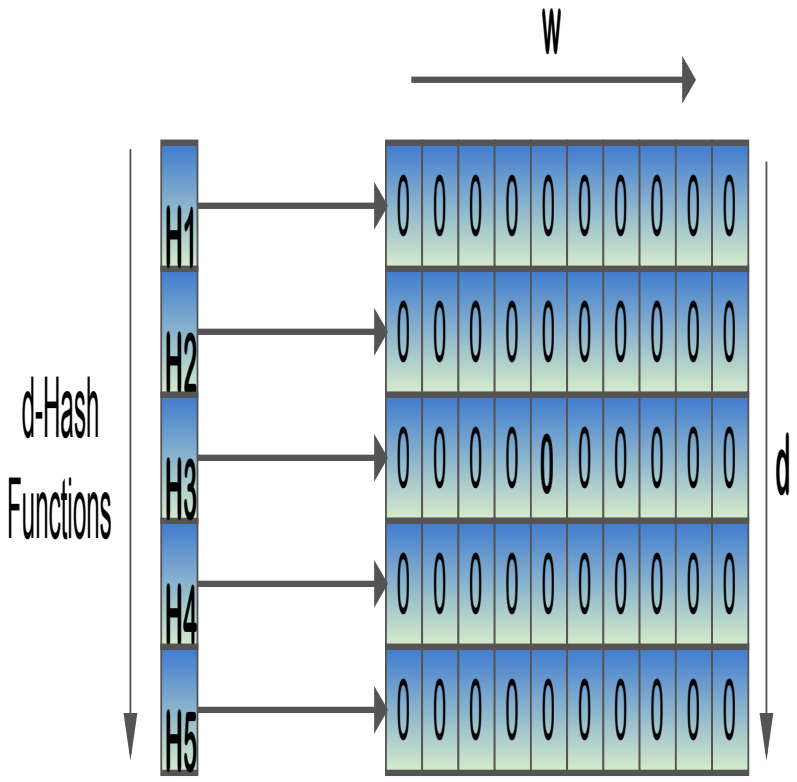

Figure 1.7: Basic CMS

ing data structures can answer certain questions very efficiently. At the cost of

marginal error probability, programmer can control the expected error rate based on the requirement of application. Count Min sketch has the following advantages:

1. Space requirement is proportional to $1/\epsilon$
2. Update time is independent of the size of sketch
3. Only pairwise independent hash functions are required
4. Can be used for multiple applications and different queries like point query, range query and inner product query *etc.*

1.3.1 Insertion and Query Operations in CMS

Let S be a set of N elements $S = \{s_1, s_2, \cdots, s_n\}$. To insert an item $s_1 \in S$ into CMS[1,1 : d,w], bit indices $B_i = h_i(s_1), 1 \leq i \leq d$ are calculated to map element in each row and counter is incremented by one at that positions. Insertion of element in CMS is depicted in figure 1.8

For getting count of an item $s_1 \in S$ similar operation is performed, bit indices

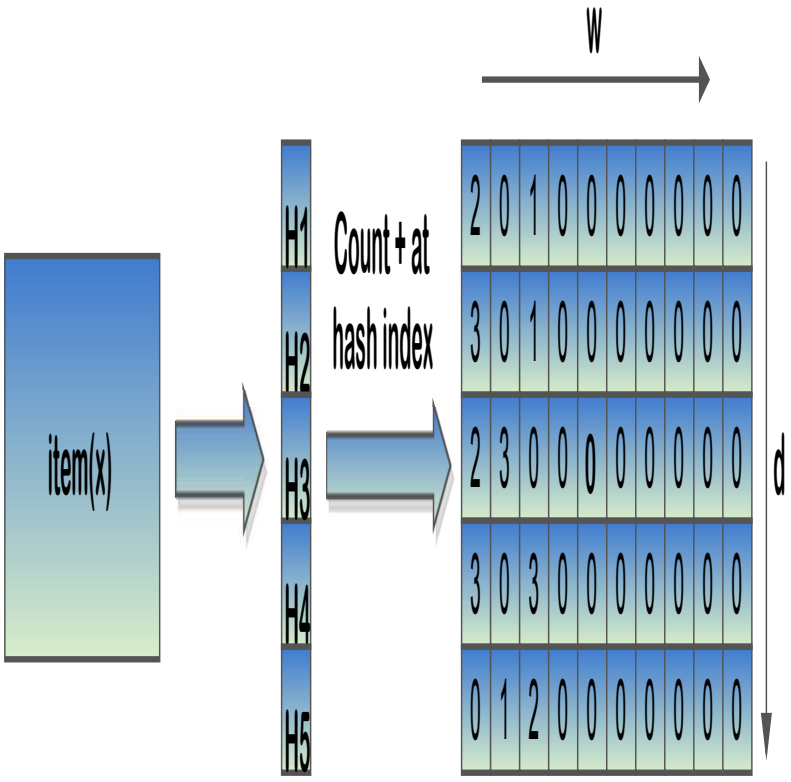

Figure 1.8: Insertion into CMS

$B_i = h_i(s_1), 1 \leq i \leq d$ are calculated and count of item s_1 in set S is found

by $minimum(CMS[1, B_1] \cdots, CMS[d, B_i])$. Since there is a possibility of hash collisions of different hash functions, count of an item may be incremented extra times due to hash collision between objects. By taking minimum value from all d rows we ensures that we are taking the value that has least number of hash collisions and ensures that we are getting to be most correct answer
For getting desired level of accuracy two parameters ϵ and δ are used to find dimensions depth d and width w of CMS. Here ϵ and δ are error and probability parameters respectively.

$$w = \lceil \frac{e}{\epsilon} \rceil \tag{1.3}$$

$$w = \lceil \ln(\frac{1}{\delta}) \rceil \tag{1.4}$$

where, ln is natural log and e is Euler's constant.
In [19] author shows that space requirement of CMS for finding heavy hitters is proportional to:

$$O(\frac{1}{\epsilon} log(\frac{l}{\delta})) \tag{1.5}$$

where l is length of sequence.

A CMS(d*w) with N elements has $O(d)$ time complexity for both insertion and query operation. If $T(s_1)$ and $E(s_1)$ are true count and count estimated by CMS of data item s_1 in set S respectively then with probability $(1 - \delta)$ CMS guarantees that:

$$T(x) \leq E(x) \leq T(x) + \epsilon * number\ of\ items\ added \tag{1.6}$$

1.4 Thesis Organization

The thesis is organized into following 6 chapters:

- **Chapter 1**: This chapter introduces intrusion detection, intrusion detection system and role of probabilistic data structures in IDS. This chapter also provides a brief introduction of BF and CMS.
- **Chapter 2**: This chapter provides a survey of literature in the related area.
- **Chapter 3**: In this section problem formulation along with research objectives and research methodology are provided.
- **Chapter 4**: This chapter described proposed method along with hashing techniques used.

- **Chapter 5**: Results obtained from proposed method are provided in this section. This chapter discusses the performance of the proposed method and compares its result on various parameters with standard CMS method.
- **Chapter 6**: Finally, conclusion and future scope of proposed method are provided into this chapter.

Chapter 2

Related Work

2.1 Bloom Filter

Till date several techniques that uses PDSs have been introduced to detect intrusion as well as to improve efficiency of the existing work. To get an overview of work done in the area of IDS based on PDSs some techniques are discussed in this section.

Anagnostakis *et al.* [20] proposed a pattern matching algorithm called E^2xB for Network intrusion detection system (NIDS)string matching. E^2xB is an exclusion filter, that is designed to increase efficiency and capacity of NIDS. In this method, packets are preprocessed using a 256 cell map and within the packet existence of each character of pattern is marked. An aggregated BF for NIDPS was introduced by Sertac *et. al.* [21] where hash functions are calculated sequentially for removing redundant hash calculations and memory access to optimize query. If no match is found with any of the hash result, the query is terminated immediately and for better utilization of on chip memory, small distributed BFs are aggregated into single BF.

Al-Dalky *et. al.* [22] introduced a method for accelerating efficiency of snort. It is an two layer implementation of NIDS. The proposed design combines hardware and software components. For analysing and filtering incoming packets BF based implementation of NetFPGA is used. Attig *et. al.* [23] implemented a string matching circuit using bloom filters within the FPX platform. The circuit scans for 35,475 unique signatures by using a 155 block RAM. Dharmapurikar *et. al.* [24] proposed a more effective string matching approach for network IDS. In this approach authors implements BF on a commodity FPGA. Song *et. al.* [7] proposed a novel fast lookup algorithm which uses extended BF for speeding up the Hash table lookups and for finding exact matches BFs are modified. By cutting the number of hash collisions and using a small external memory, speed up is achieved. But its main shortcoming is that it requires an expensive cache like on-chip memory. Dharmapurikar *et. al.* [8] proposed a scalable hardware pattern matching algorithm which is a combination of classic AC algorithm and BF. In

this method BF is implemented using embedded on chip memory blocks in FPGA. Kocak *et. al.* [25] proposed a deep packet inspection method based on bloom filter where hash functions are divided into two pipeline stages, hash functions of second stage are calculated when all matches are found for hash functions of stage one.

Chen *et. al.* [26] proposed a novel method to speed up pattern matching for IDS that combines fingerprint and pattern matching techniques. In this method a short digest for each pattern is generated by the system, then digest of the incoming packets are computed and then exact match BM algorithm is used to find the patterns. To detect all possible attacks a hardware parallel architecture based on BF was proposed by Chaudhary *et. al.* [9] in 2010. For detecting possible attacks with minimum delay, testing of input string is performed simultaneously. For all test string hash functions are calculated concurrently. To improve performance of multi pattern matching algorithm a system called sigMatch was introduced by Kandhan *et. al.* [27] in 2010. In this method, patterns are organized into sigTree based on common sub pattern, for this patterns are prepocessed. It has features of faster tries and memory efficient BFs. To discard unmatched pattern, all patterns are match against sigTree.

Exscind, a software based pattern matching algorithm was proposed by Aldwairi *et. al.* [28]. In this authors proposed a modified BF; an exclusion-inclusion filter used to produce list of probable matching signatures and modified Wu-Manber algorithm is used for pattern matching. A distributed, fast and light-weight intrusion detection method for smart grid SCADA have been proposed by Parthasarath *et. al.* [29]. Proposed method is anomaly-based and uses BFs for memory efficiency which exploits the nature of the SCADA communication pattern for detecting intrusion in the field devices. Snort is an open source software based NIDS. Aldwairi *et. al.* [30] proposed a novel method to modify Wu-Manber pattern matching algorithm with BF. It is a software based pattern matching algorithm, where an exclusion filter is used to reduce number of access to large HASH table. Liu *et. al.* [31] introduced a variant of BF called vector bloom filter(VBF) for detecting super-points in a network (In a short time a large number of connections made by an infected host; such a host is known as superpoint). Six hash functions are used in VBF.

Meghana *et. al.* [32] proposed a hardware based NIDS using counting bloom filter. In this method a warning message is circulated in case of receiving signals regarding presence of virus in network and it is displayed on the LCD. The proposed hardware based method reduces the limitations of software based method and provide increased protection. A method for identifying frequent traffic flow was

introduced by Xiong *et. al.* in 2017 [33]. In this authors introduced a new data structure called probabilistic bloom filter (PBF), which is an extension of standard BF and an extension of PBF for dynamic data streams was also introduced. Zhou *et. al.* [34] proposed a hardware base IDS. This method is introduced for on-line detection of intrusion in microprocessors using system call routine fingerprinting. Bloom filter is employed for examining validity of fingerprints.
some applications of BFs are provided into figure 2.1.

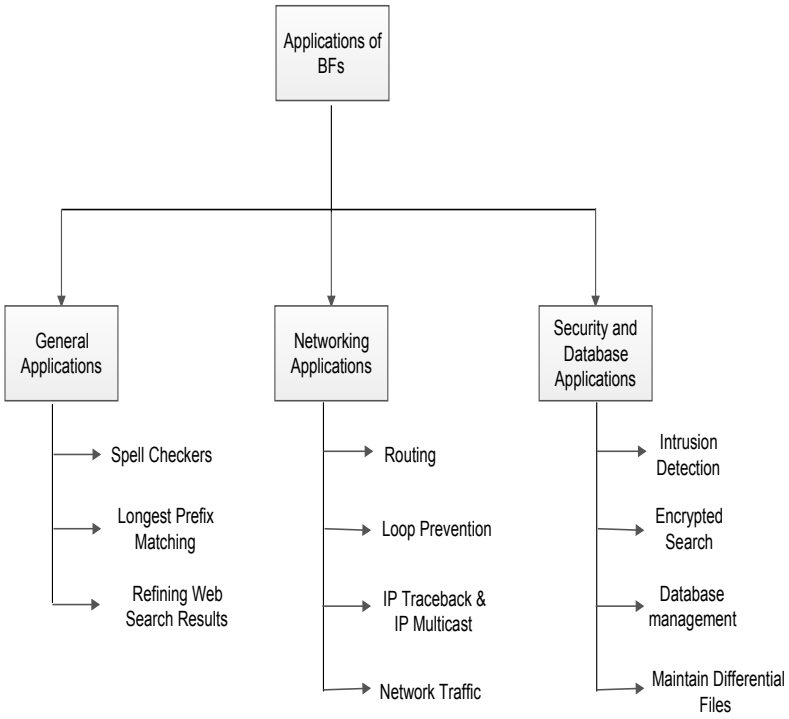

Figure 2.1: Some Applications of BFs

2.2 CMS

As CMS is also one of the memory efficient data structure and most suitable for finding heavy hitters it has been used in different applications. Some of them are: Barman *et. al.* [35] proposed a method of detecting attacks in routers based on sketches and change detection. In this method detection of various attacks such as viruses, worms, super-spreaders and DOS in routers is performed by monitoring only IP headers. In this method CMS, FM sketch and counting and multi-counting BFs are employed to identify attacks in the routing architecture. Salem *et. al.* [36] introduced a method of anomaly based IDS. This method has two phase, in the first phase data reduction is through flow sampling. In the second phase two data structures CMS and multi layer reversible sketch are employed for random

aggregation of description. This method detects anomaly as well as classifies the anomaly as DoS, DDoS or flash crowd *etc.* A method for DDoS attack detection is introduced by Liu *et. al.* in 2011 [37]. This is a two level approach, in the first level modified CMS is used for fast detection of attacks and in second level variant of CMS called Bidirectional count sketch (BCS) have been proposed to achieve better accuracy. This method does not require to store every IP address found in network traffic to detect attacks.

Li *et. al.* [38] proposed a method of anomaly detection in WSN. This method is based on CMS and clustering method. In this method sensor network is partitioned in clusters based on physical adjacency and data correlation. Each member of cluster has a CMS, all members CMS are collected by cluster header and it compares it with own sketch. A method of detecting anomaly in WSN based on CMS is proposed by Li *et. al.* [39]. Irodache *et. al.* [40] proposed a method of anomaly detection for data centre networks. They proposed a two level approach for anomaly detection, in which they modify CMS to store IP address of source and destination and transport layer ports.

Chapter 3

Problem Statement

3.1 Motivation

IDS are powerful tools for protecting network against attacks. Performance of signature-based IDS depends on efficiency of pattern matching algorithm and it is a computationally extensive task. Since number of attacks are increasing day by day, attack signatures are also increases making IDS task more harder. Further space requirement for storing signature database is also increasing leading to issue of memory requirement and computational cost. For addressing these issues we proposes a novel malicious detection technique based on memory efficient BF and CMS.

3.2 Problem Description

Definition 1: **Heavy Hitter:** Let S be a set of N items, $S = \{s_1, s_2, \cdots, s_n\}$ and ϕ is a predefined threshold (referred to as minimum support). An item $s_1 \in S$ is called a heavy hitter if and only if $freq\_count(s_1, S) \geq \phi * N$, where $freq\_count(s_1, S)$ is frequency count of item s_1 in set S.

A list S of N suspicious node is provided by the network administrator along with a predefined threshold ϕ. Main aim is to monitor the network traffic and track the hitting frequency of suspicious nodes and find all malicious nodes (heavy hitters) in the given time slot t .

3.3 Objectives

The main focus of our thesis work is to monitor network traffic and track hit rate of the suspicious nodes to identify malicious nodes in the network. The key objectives of this work are listed below:

- To study existing Probabilistic Data Structures for storing and counting of the massive data sets efficiently.
- To propose a novel technique for malicious node detection using PDS.
- To validate the performance of the proposed technique through various evaluation parameters.

3.4 Methodology

Major aim of this work is improving the existing techniques, making them more efficient by reducing storage and computational cost. Below is the methodology that will be followed to achieve the same:

- Comprehensive and thorough study of existing PDS and IDS will be done.
- Extensive study of R-programming language and R-studio will be done.
- A novel technique will be proposed for detection of malicious host.
- Simulation and theoretical analysis of the proposed method will be done.

Chapter 4

Intrusion Detection through BF and modified CMS

4.1 Proposed Method

In this work we proposed a method of malicious detection based on bloom filter and count min sketch, which reduces the storage requirement and computational cost. For storing list of suspicious nodes BF is used (it is assumed that list of suspicious nodes is provided by network administrator) since it reduces memory required to store suspicious nodes. For identifying hit rate of suspicious nodes in every t time slot, count min sketch is used. Work flow of proposed method is illustrated in Figure 4.1

4.1.1 d-left Hashing Technique

In the d-left hashing technique hash table is divided into d-sub tables of equal size. n buckets are considered where each sub table has n/d buckets and capacity of each bucket is c cells. In the d-left hashing scheme d hash functions are used. Whenever an item is inserted using d-left hashing scheme d-indices are computed and element is placed in the least loaded bucket. For breaking the ties element is placed in the bucket of left most sub table. In the d-left hashing :

1. Maximum load is quite less, proportional to $O(loglogn)$.
2. For small values of d, maximum load is close to average load.

Proposed method work in the following steps:

4.1.2 Suspicious Node Storage

Let $S = \{x_1, x_2, \cdots, x_n\}$ is a list of suspicious nodes provided by network administrator and $BF[1 \cdots m]$ is a bloom filter of size m. To store set S in BF, k independent hash function are chosen. To insert an item $x \in S$ in BF k indices

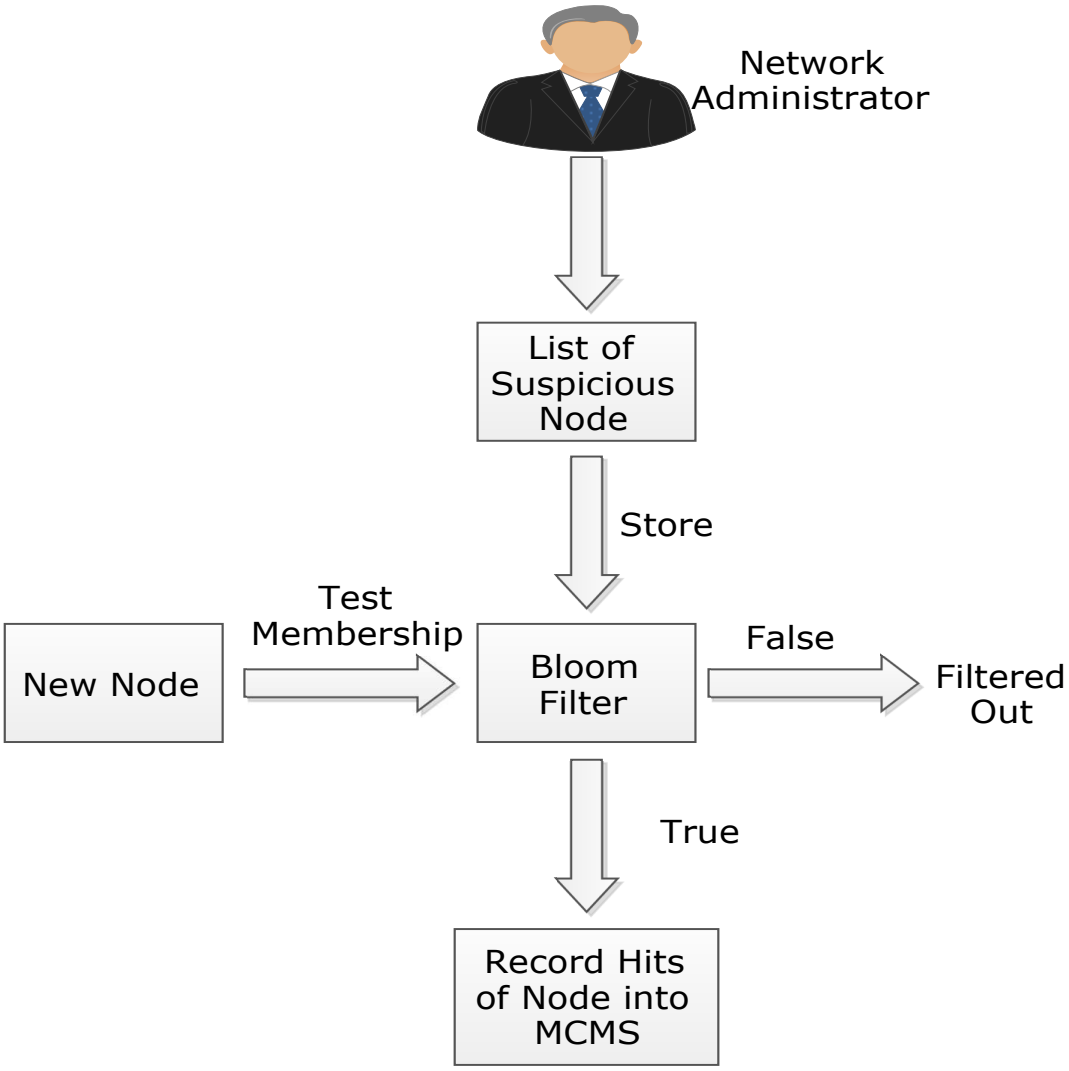

Figure 4.1: Workflow of Proposed Method

are calculated by computing $B_i = h_i(x), 1 \leq i \leq k$, and corresponding positions to B_i are set to high in BF. Steps of store function are illustrated in algorithm 4.1.

4.1.3 Membership Testing

Whenever a node x_i tries to access network resources, checking is performed to identify that whether x_i belongs to the list of suspicious nodes provided by network administrator or not.

- Firstly, list of suspicious nodes are inserted into $BF[1 : m]$, with k hash functions.

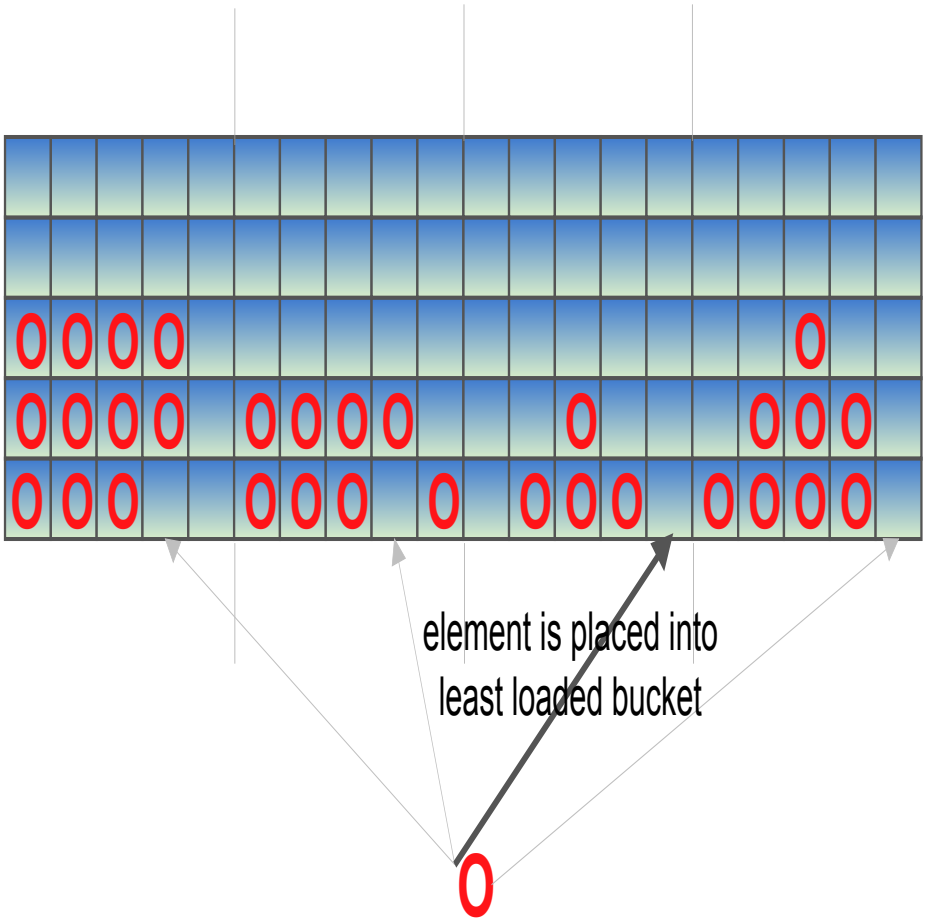

Figure 4.2: d-left Hashing technique

- Every node trying to access network is checked to identify whether it belongs to list of suspicious nodes or not. For testing set membership of new node x_i query is made to BF, if query results says that $x_i \in S$, count of x_i is recorded.
- If $x_i \notin S$ it is filtered out.

4.1.4 Track Count of Suspicious Nodes

For a node x_i if the result of step II (Membership testing) is true, then to monitor number of times this node has tried to send the data in the given time span t referred as hits of x_n, modified count min sketch is used, which uses partial d-left hashing and performs selective updates. To insert x_n into MCMS following steps are performed:

- Calculate the hash functions using double hashing, *i.e.* only two hash func-

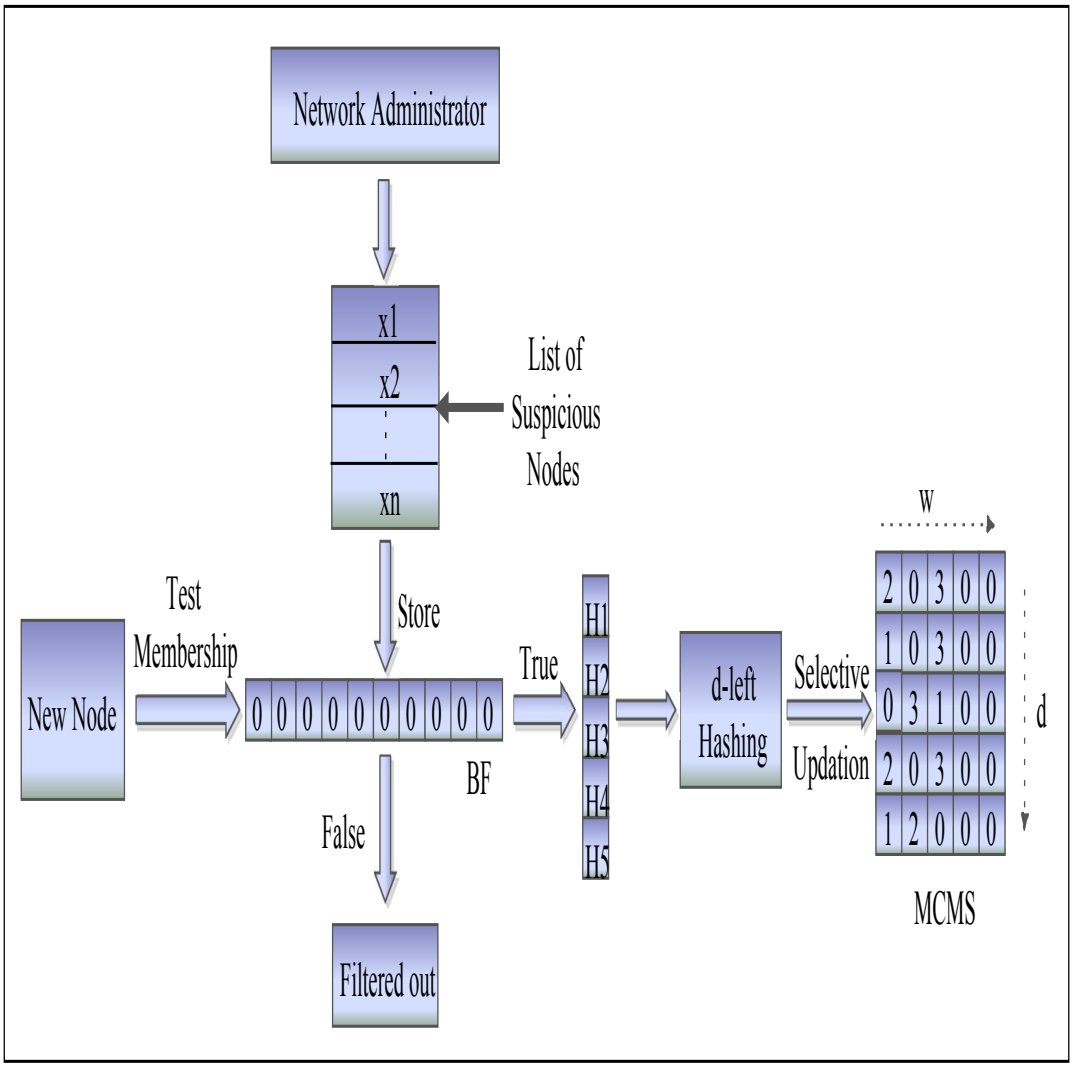

Figure 4.3: System Architecture

tions are used to generate all d hash functions of MCMS.

- Selective updation are performed at computed hash indexes in each row of MCMS.
- Counter of row with minimum value is increased *i.e.*
  $A \leftarrow min(MCMS^i[h_1(x)], MCMS^i[h_2(x)], \cdots, MCMS^i[h_d(x)])$
  (Here $MCMS^i[j]$, denotes ith row and jth column),
  $A \leftarrow A + 1$.
- For breaking ties, updation is performed in counter of each row.

Algorithm 4.1 Store Function

Data: S is the set of suspicious node to be stored
Function: store(S)

```
for i in S do
  for j=1 to k do
    /* for each element of S compute indices with K hash functions */
    A ← h_i(y)
    if B_A == 0 then
      /* check bits at position A_i is zero or not */
      B_A ← 1 /* if yes set bit to one */
    end if
  end for
end for
```

4.1.5 Monitoring Heavy Hitters

For analysing network traffic and tracking activities of suspected nodes network activities need to be monitored in real time. In on-line monitoring process, nodes which exceed the predefined threshold are identified and informed to network administrator immediately, to take suitable action against the malicious node. In the given time slot t:

- If any suspicious node crosses the predefined threshold ϕ it is declared as highest probability malicious node (HMP) and informed to network administrator.
- If there are some hits from a node $x < \phi$ then it is categorised it as medium probability malicious node (MMP).
- If there is no hit from suspicious node x then it is categorised as least probability malicious node (LMP).

4.2 Technologies Used

4.2.1 R-programming Language

Implementation of proposed method is done in R programming language. It is an open source programming language with an environment for statistical computing and graphics. It is an interpreted language; users access it by command line interpreter. It provides a wide variety of statistical and graphical techniques. Since it creates unique data visualization, popularity of R is increasing rapidly.

The R environment: It is a effective, simple and well developed programming

Algorithm 4.2 Finding Malicious Node

Data: x is the suspecious node
Function: findmalicious(x)

```
Step1 testing x ∈ S or not
Set: j ← 1
for i=1 to k do
    /* compute indices with all k hash functions */
    A ← h_i(x)
    if B_A == 1 then
        /* check bits at position B_A is set to one or not */
        continue;
    else
        Set: j ← 0
        break;
    end if
end for
if j==1 then
    return true
else
    return false
end if
Step2
if result of step1 is true then
    /* if result of step1 is true then record count
    of x */
    for i=1 to d do
        /* compute indices with all d hash functions */
        A_i ← h_i(x)
        mini ← min(mcms[A_1], mcms[A_2], ··· mcms[A_d])
        Increase counter value by one that has minimum count
        if tie then
            /* all have same value increase counter value by one at eachrow */
            mcms[A_1] ← mcms[A_1] + 1
            mcms[A_d] ← mcms[A_d] + 1
        end if
    end for
else
    filtered out it
end if
if count of x ≥ φ then
    generate signal
end if
```

language, providing facilities of data manipulation, calculation and graphical display. It includes:

- Storage and effective data handing facility.
- Large number of tools for data analysis.
- Graphical facility for analysing data and displaying it.
- Loops and conditions analysis, I/O facilities and user defined recursive function.
- suite of operators for calculations on arrays, in particular matrices.
- Over 4800 packages available in R from multiple repositories specializing in topics like data mining, econometrics, bio-informatics and spatial analysis.
- In R anyone can provide code enhancements, bug xes, and new packages.

Some business intelligence and big data tools that integrate with R are:

1. hp-Vertica
2. Azure
3. Spark
4. Mongo DB
5. Hortonworks
6. Microstrategy
7. SAP. HANA
8. Haddop *etc.*

4.2.2 R-studio

R-studio, developed by Allaire, is an free, open source integrated environment(IDE) for R programming language. It is written in $C++$ programming language.

- It runs on desktops, servers and it is easily accessible over the web.
- It integrates tools into a single environment.
- Supports rapid navigation to functions and files.
- To enhance productivity, it provides powerful coding tools.
- Support interactive graphics

- Comes with interactive debugger.
- Using projects easily manage multiple working directories.

Chapter 5

Implementation and Results

In this section results obtained by implementing the proposed approach in R are discussed. Hashing for storing list of suspicious nodes into bloom filter and monitoring of network traffic is done using CityhashFunction. For monitoring network traffic, insertion is performed in MCMS using partial d-left hashing technique. For monitoring $x \in S$ we compute d indices and update value of counter in row-i that has value $min(mcms[h_1(x)], mcms[h_2(x)], \cdots mcms[hd(x)])$. For breaking tie, instead of updating value into left one, we update value in all d-rows.

City Hash Function: It is a family of non-cryptographic hash functions; designed for fast hashing of strings. It has 32-bit, 64-bit, 128-bit and 256-bit variants.

Simulation are performed on a PC with Intel core i5 processor having 4 GB of main memory running Microsoft Windows-7. For the experimental purposes list of suspicious node considered are 10000, time slot considered is two hours, *i.e.*, ($t = 2$) and $\phi = 2.5\%$. For MCMS parameters considered are $k = 5, d = 2, w = 150$. Experiment is carried out for 10 days between 8 a.m. to 8 p.m. and average performance results are obtained as shown in Table 5.1. From figure 5.2 it is observed that suspicious nodes are highly active between time slot 6 p.m. to 8 a.m. and 2 p.m. to 4 p.m. while they are least active between time slot 8 a.m. to 10 a. m. From figure 5.4 it is observed that update requirement of MCMS is less than CMS because in CMS when an item is inserted updation is carried out in all d-rows where as in proposed method at every insertion updation is not necessarily carried out in all d-rows. In network, memory requirement and computational cost is limiting factor, our experimental results show that by using small value of d, w false positive rate decreased in MCMS compared with standard CMS as depicted in figure 5.1, 5.3. Further, it is also observed that false positive rate decreases as value of d increases. Similarly if d is constant and value of w is increased, false positive rate decreases and false positive rate increases with increase in number of elements. Figure 5.6 shows that accuracy of CMS decreases with a small increase in number of elements, but accuracy of MCMS is almost constant with the increase in number of elements upto certain level. It is also

observed that accuracy of MCMS is better than CMS with d=2.

Two parameters ϵ and δ are used to calculate width w and depth d of count min sketch. Figure 5.5 shows value of $(d * w)$ with respect to parameter ϵ and δ.

Table 5.1: Performance Result

Time	Highest Probability Malicious Node	Medium Probability Malicious Node	Least Probability Malicious Node
8am - 10am	50	500	9450
10am - 12pm	200	3000	6800
12pm - 2pm	1000	6000	3000
2pm - 4pm	1700	7500	800
4pm - 6pm	800	1500	7700
6pm - 8pm	2200	7500	300

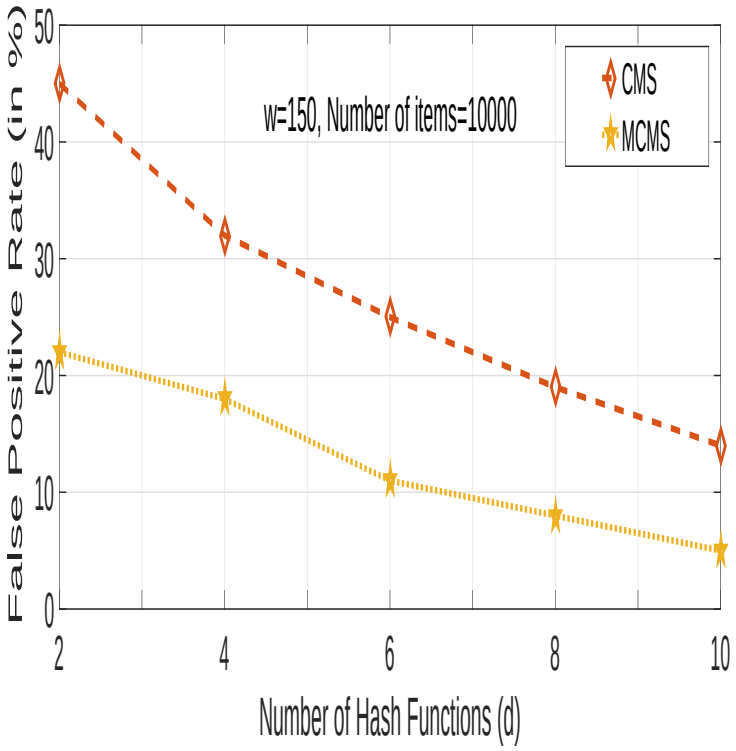

Figure 5.1: False Positive rate With Respect to d

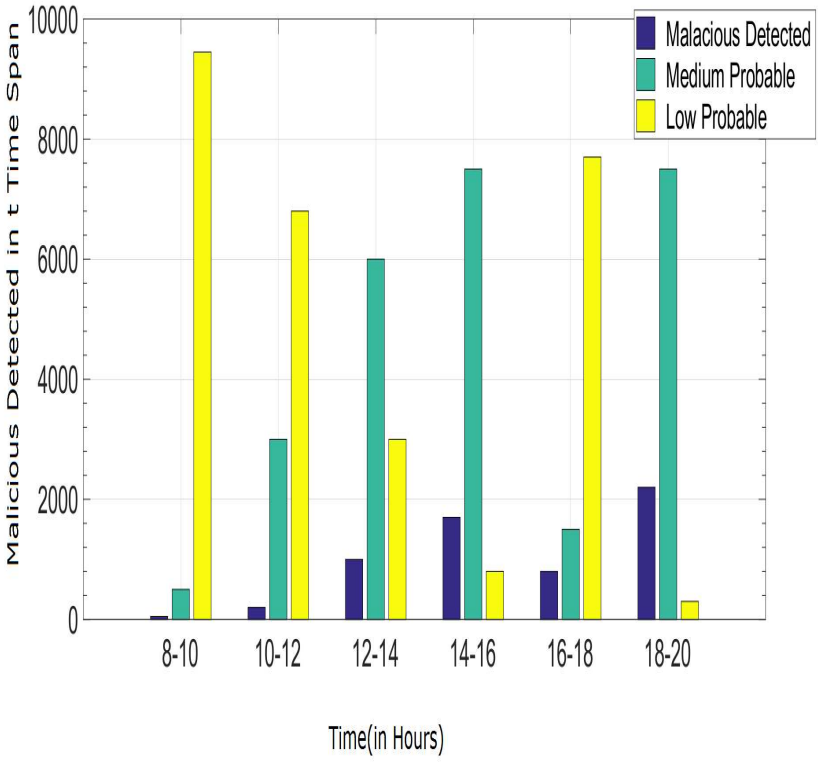

Figure 5.2: Malicious Detected in t Time Span

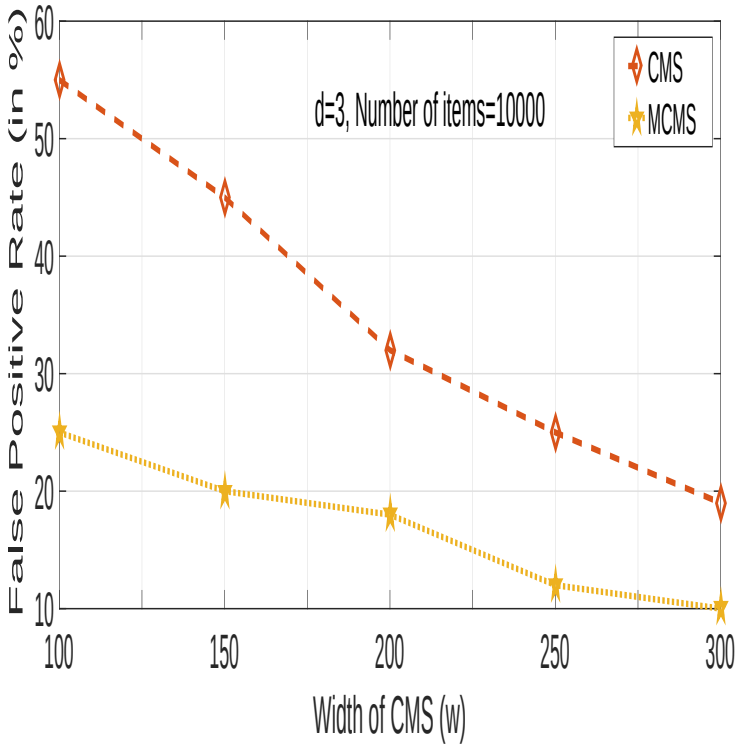

Figure 5.3: False Positive Rate vs. width of CMS w

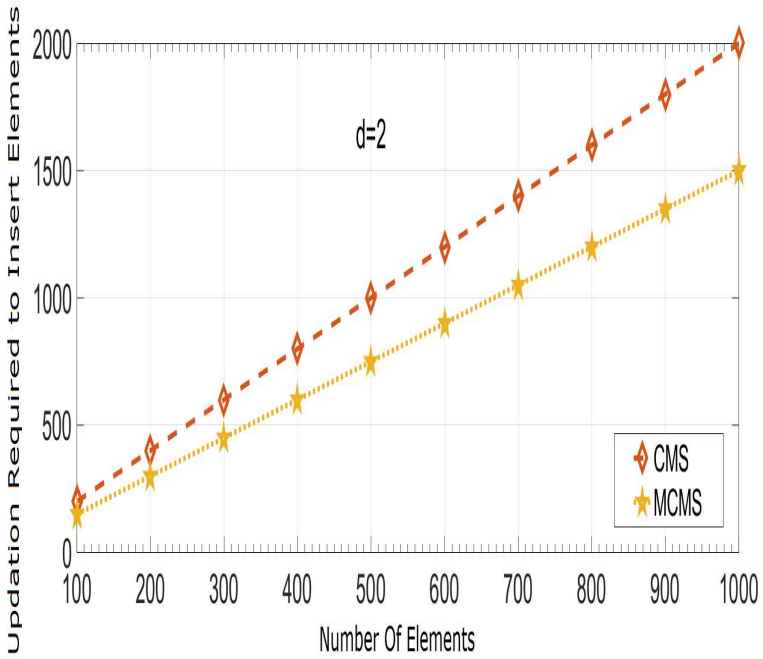

Figure 5.4: No. of Updation required to Insert Elements

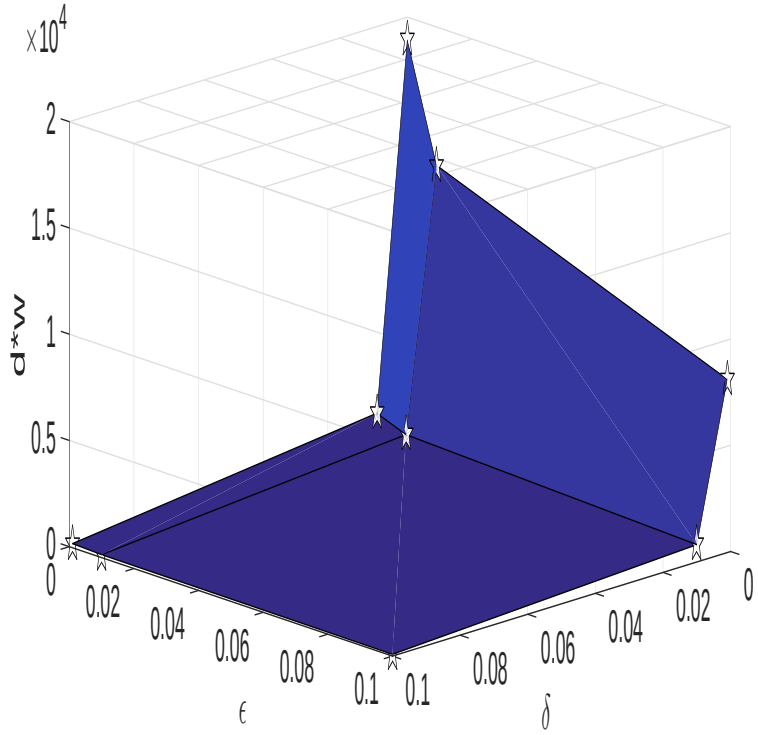

Figure 5.5: value of $w * d$ with ϵ, δ

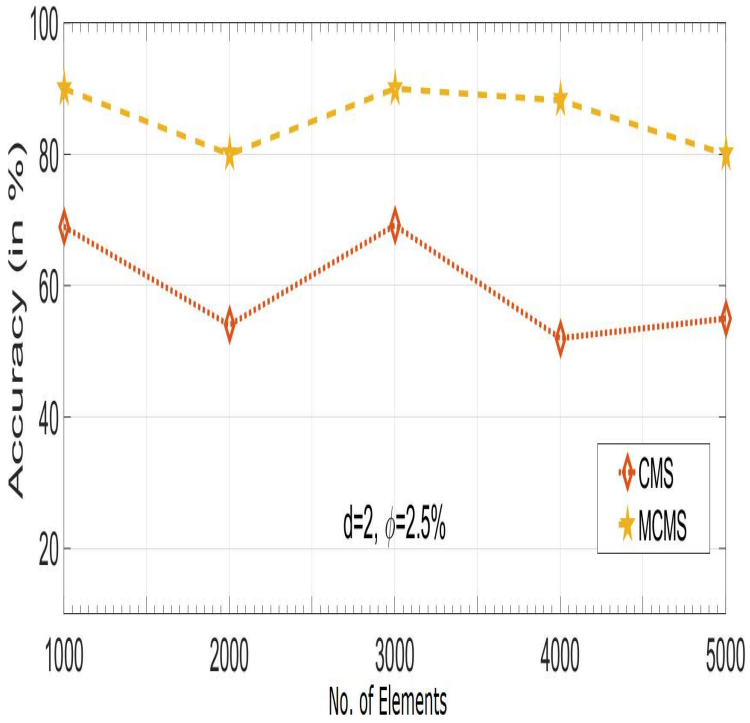

Figure 5.6: Accuracy of CMS and MCMS

Chapter 6

Conclusions and Future Work

In this chapter along with conclusion of the thesis, some suggestions for extending presented work are also discussed.

6.1 Conclusion and Summary

As Internet is evolving in all kinds of activities and data is increasing rapidly, there is an increasing need for analysing the network traffic for detecting and preventing malicious activities. In this work:

- A novel technique of malicious detection have been proposed which utilizes Bloom filter and Count min sketch.
- Bloom filter is employed for reducing storage and computational cost and access list of suspicious nodes.
- Modified Count Min Sketch is employed for monitoring and analysing the network traffic and tracking count of suspicious nodes.
- Experimental results presented in Chapter 5 show that MCMS results in low false positive rate compared to CMS with small value of parameter d and w.
- For decreasing collision rate, d-left hashing has been employed in the CMS.

6.2 Scope for future work

Research is a continuous and iterative process. The work proposed in the thesis is an attempt to solve problem of intrusion detection for network traffic with reduction in storage and computational cost. There are some suggestions in which proposed work can be extended. This are:

- In the future, it can be extended for detection of malicious events in streaming data.

- In the proposed method it is assumed that list of suspicious nodes is provided by network administrator, in future efforts will be devoted to go one step ahead and an efficient method will be proposed for generating list of suspicious node.

References

[1] R. Proudfoot, K. Kent, E. Aubanel, and N. Chen, "High performance software-hardware network intrusion detection system," in *Field-Programmable Technology, 2007. ICFPT 2007. International Conference on*. IEEE, 2007, pp. 309–312.

[2] N. S. Artan and H. J. Chao, "Design and analysis of a multipacket signature detection system," *International Journal of Security and Networks*, vol. 2, no. 1-2, pp. 122–136, 2007.

[3] S. Dharmapurikar, P. Krishnamurthy, T. Sproull, and J. Lockwood, "Deep packet inspection using parallel bloom filters," in *High performance interconnects, 2003. proceedings. 11th symposium on*. IEEE, 2003, pp. 44–51.

[4] G. Gu, M. Sharif, X. Qin, D. Dagon, W. Lee, and G. Riley, "Worm detection, early warning and response based on local victim information," in *Computer Security Applications Conference, 2004. 20th Annual*. IEEE, 2004, pp. 136–145.

[5] B. Madhusudan and J. Lockwood, "Design of a system for real-time worm detection," in *High Performance Interconnects, 2004. Proceedings. 12th Annual IEEE Symposium on*. IEEE, 2004, pp. 77–83.

[6] O. Erdogan and P. Cao, "Hash-av: fast virus signature scanning by cache-resident filters," in *Global Telecommunications Conference, 2005. GLOBECOM'05. IEEE*, vol. 3. IEEE, pp. 6–pp.

[7] H. Song, S. Dharmapurikar, J. Turner, and J. Lockwood, "Fast hash table lookup using extended bloom filter: an aid to network processing," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 4, pp. 181–192, 2005.

[8] S. Dharmapurikar and J. W. Lockwood, "Fast and scalable pattern matching for network intrusion detection systems," *IEEE Journal on Selected Areas in communications*, vol. 24, no. 10, pp. 1781–1792, 2006.

[9] D. Chaudhary, "Parallel processing of bloom filter," *International Journal of Electronic Engineering Research*, vol. 2, no. 1, pp. 35–40, 2010.

[10] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[11] L. Fan, P. Cao, J. Almeida, and A. Z. Broder, "Summary cache: A scalable wide-area web cache sharing protocol," in *ACM SIGCOMM Computer Communication Review*, vol. 28, no. 4. ACM, 1998, pp. 254–265.

[12] F. Bonomi, M. Mitzenmacher, R. Panigrahy, S. Singh, and G. Varghese, "An improved construction for counting bloom filters," in *European Symposium on Algorithms*. Springer, 2006, pp. 684–695.

[13] C. E. Rothenberg, C. A. Macapuna, F. L. Verdi, and M. F. Magalhaes, "The deletable bloom filter: a new member of the bloom family," *IEEE Communications Letters*, vol. 14, no. 6, 2010.

[14] K. Shanmugasundaram, H. Brönnimann, and N. Memon, "Payload attribution via hierarchical bloom filters," in *Proceedings of the 11th ACM conference on Computer and communications security*. ACM, 2004, pp. 31–41.

[15] F. Deng and D. Rafiei, "Approximately detecting duplicates for streaming data using stable bloom filters," in *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*. ACM, 2006, pp. 25–36.

[16] Y. Matsumoto, H. Hazeyama, and Y. Kadobayashi, "Adaptive bloom filter: A space-efficient counting algorithm for unpredictable network traffic," *IEICE transactions on information and systems*, vol. 91, no. 5, pp. 1292–1299, 2008.

[17] P. S. Almeida, C. Baquero, N. Preguiça, and D. Hutchison, "Scalable bloom filters," *Information Processing Letters*, vol. 101, no. 6, pp. 255–261, 2007.

[18] R. P. Laufer, P. B. Velloso, and O. C. M. Duarte, "A generalized bloom filter to secure distributed network applications," *Computer Networks*, vol. 55, no. 8, pp. 1804–1819, 2011.

[19] G. Cormode and S. Muthukrishnan, "An improved data stream summary: the count-min sketch and its applications," *Journal of Algorithms*, vol. 55, no. 1, pp. 58–75, 2005.

[20] K. G. Anagnostakis, S. Antonatos, E. P. Markatos, and M. Polychronakis, "E 2 xb: A domain-specific string matching algorithm for intrusion detection," in *Security and Privacy in the Age of Uncertainty*. Springer, 2003, pp. 217–228.

[21] N. S. Artan, K. Sinkar, J. Patel, and H. J. Chao, "Aggregated bloom filters for intrusion detection and prevention hardware," in *Global Telecommunications Conference, 2007. GLOBECOM'07. IEEE*. IEEE, 2007, pp. 349–354.

[22] R. Al-Dalky, K. Salah, H. Otrok, and M. Al-Qutayri, "Accelerating snort nids using netfpga-based bloom filter," in *Wireless Communications and Mobile Computing Conference (IWCMC), 2014 International*. IEEE, 2014, pp. 869–874.

[23] M. Attig, S. Dharmapurikar, and J. Lockwood, "Implementation results of bloom filters for string matching," in *Field-Programmable Custom Computing Machines, 2004. FCCM 2004. 12th Annual IEEE Symposium on*. IEEE, 2004, pp. 322–323.

[24] S. Dharmapurikar, M. Attig, and J. Lockwood, "Design and implementation of a string matching system for network intrusion detection using fpga-based bloom filters," 2004.

[25] T. Kocak and I. Kaya, "Low-power bloom filter architecture for deep packet inspection," *IEEE Communications Letters*, vol. 10, no. 3, pp. 210–212, 2006.

[26] Z. Chen, Y. Zhang, Z. Chen, and A. Delis, "A digest and pattern matching-based intrusion detection engine," *The Computer Journal*, vol. 52, no. 6, pp. 699–723, 2009.

[27] R. Kandhan, N. Teletia, and J. M. Patel, "Sigmatch: fast and scalable multi-pattern matching," *Proceedings of the VLDB Endowment*, vol. 3, no. 1-2, pp. 1173–1184, 2010.

[28] M. Aldwairi and D. Alansari, "Exscind: Fast pattern matching for intrusion detection using exclusion and inclusion filters," in *Next Generation Web Services Practices (NWeSP), 2011 7th International Conference on*. IEEE, 2011, pp. 24–30.

[29] S. Parthasarathy and D. Kundur, "Bloom filter based intrusion detection for smart grid scada," in *Electrical & Computer Engineering (CCECE), 2012 25th IEEE Canadian Conference on*. IEEE, 2012, pp. 1–6.

[30] M. Aldwairi and K. Al-Khamaiseh, "Exhaust: Optimizing wu-manber pattern matching for intrusion detection using bloom filters," in *Web Applications and Networking (WSWAN), 2015 2nd World Symposium on*. IEEE, 2015, pp. 1–6.

[31] W. Liu, W. Qu, J. Gong, and K. Li, "Detection of superpoints using a vector bloom filter," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 3, pp. 514–527, 2016.

[32] V. Meghana, M. Suresh, S. Sandhya, R. Aparna, and C. Gururaj, "Soc implementation of network intrusion detection using counting bloom filter," in *Re-*

cent Trends in Electronics, Information & Communication Technology (RTEICT), IEEE International Conference on. IEEE, 2016, pp. 1846–1850.

[33] S. Xiong, Y. Yao, M. Berry, H. Qi, and Q. Cao, "Frequent traffic flow identification through probabilistic bloom filter and its gpu-based acceleration," *Journal of Network and Computer Applications*, vol. 87, pp. 60–72, 2017.

[34] L. Zhou and Y. Makris, "Hardware-based on-line intrusion detection via system call routine fingerprinting," in *2017 Design, Automation & Test in Europe Conference & Exhibition (DATE).* IEEE, 2017, pp. 1546–1551.

[35] D. Barman, P. Satapathy, and G. Ciardo, "Detecting attacks in routers using sketches," in *High Performance Switching and Routing, 2007. HPSR'07. Workshop on.* IEEE, 2007, pp. 1–6.

[36] O. Salem, S. Vaton, and A. Gravey, "A scalable, efficient and informative approach for anomaly-based intrusion detection systems: theory and practice," *International Journal of Network Management*, vol. 20, no. 5, pp. 271–293, 2010.

[37] H. Liu, Y. Sun, and M. S. Kim, "Fine-grained ddos detection scheme based on bidirectional count sketch," in *Computer Communications and Networks (ICCCN), 2011 Proceedings of 20th International Conference on.* IEEE, 2011, pp. 1–6.

[38] G. Li and Y. Wang, "Sketch based anomaly detection scheme in wireless sensor networks," in *Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2013 International Conference on.* IEEE, 2013, pp. 344–348.

[39] G. Li, Y. Liu, and Y. Wang, "Analysis of the count-min sketch based anomaly detection scheme in wsn," in *Trust, Security and Privacy in Computing and Communications (TrustCom), 2014 IEEE 13th International Conference on.* IEEE, 2014, pp. 961–966.

[40] M. Iordache, S. Jouet, A. Marnerides, and D. Pezaros, "Distributed, multilevel network anomaly detection for datacentre networks," 2017.

List of Publications

International Conference

1. Divya Gupta and Shalini Batra , *"A Short Survey on Bloom Filter and its Variants"*, International Conference on Computing, Communication and Automation (ICCCA-2017) sponsored by IEEE, Held on $5^{th} - 6^{th}$ *may* 2017 in Galgotias University, Noida (UP).

2. Divya Gupta, Shalini Batra, *"Intrusion Detection System based on Bloom Filter and Modified Count Min Sketch for Malicious Host Detection"*, IEEE GLOBECOM-2017. [Under Review]

Video Link

https://youtu.be/6N2ZhOrZR4w

Reflective Diary

August 2016 (Semester 3)

This was the starting of third semester in ME program, in this semester along with the capstone project we had to select the research area for our thesis work. The major challenge was first to select the major domain under which we had to choose the specific topic. I was interested to work in the domain of Big data processing, so I preferred to work under the guidance of Dr. Shalini Batra, as she is one of the best faculty and guide who works in this domain.

September 2016 to November 2016 (Semester 3)

During September to November along with working on capstone project, I did thorough study of research area in the domain of Big data and we decided to work in the field of probabilistic data structures and I started reading about probabilistic data structures.

December 2016 (Semester 3)

This was the ending time of third semester in ME program and evaluation of capstone project was done. The next important task we had to do was finalizing the topic of our thesis work that to be carried out in the fourth semester. We had decided to work on two popular probabilistic data structures Bloom Filter and Count min sketch.

January 2017 (Semester 4)

This was the starting of fourth semester in ME program. In this month I had done the thorough study of PDS Bloom Filter its variants and its applications in different domains and decided to work in the field of intrusion detection based on PDS. There are three major objectives proposed by my guide for my thesis work:

- To study existing Probabilistic Data Structures for storing and counting of the massive data sets efficiently.
- To propose a novel technique for malicious node detection using PDS.
- To validate the performance of the proposed technique through various evaluation parameters.

February 2017 (Semester 4)

By this time I had done thorough study of Bloom filter hence my guide said to put this efforts into some substantial output. She said to write a review paper on it. In this month I wrote a review paper titled A Short Survey on Bloom Filter and Its variants and studied the PDS count min sketch.

March 2017(Semester 4)

In this month implemented the basic Bloom Filter and Count min sketch in R programming language. Problem formulation and a method to solve the problem was also proposed in this month.

April 2017 (Semester 4)

In this month algorithm for proposed method was designed and theoretical and practical analysis was done to test proposed method. For practical analysis, simulation was done and results was achieved were recorded. In this month we also got the acceptance of review paper in the ICCCA-2017 IEEE sponsored conference to be held on 5th-6th May 2017 in Galgotias University Noida.

May 2017 (Semester 4)

In May 2017 I went to present the paper in the ICCCA-2017 conference and under the supervision of my guide Dr. Shalini Batra I wrote a paper on the proposed method and communicated the paper to Information Processing Letters, Elsevier which is under review.

June 2017 (Semester 4)

In this month I primarily concentrated on my thesis work and drafting it nicely so that my guide finds it to be good enough to be submitted. Thesis was reviewed by my guide and she suggested some changes to improve the writing. After final revision, thesis was submitted in the CSE department.