Flooding Attacks Detection in Traffic of Backbone Networks

Osman Salem, Ali Makke, Jean Tajer, Ahmed Mehaoua

Flooding Attacks Detection in Traffic of Backbone Networks

Osman Salem, Ali Makke, Jean Tajer, Ahmed Mehaoua
University of Paris Descartes
Laboratoire d'Informatique PAris DEscartes (LIPADE)
{osman.salem, ali.makke, jean.tajer, ahmed.mehaoua}@parisdescartes.fr

***Abstract*—Internet services are vulnerable to flooding attacks that lead to denial of service. This paper proposes a new framework to detect anomalies and to provide early alerts for flooding attacks in backbone networks. Thus allow to quickly react in order to prevent the flooding attacks from strangling the victim server and its access network. The proposed detection scheme is based on the application of Least Mean Square (LMS) filter and Pearson Chi-square divergence on randomly aggregated flows in Sketch data structure. Instead of analyzing one time series for overall traffic, random aggregation of flows is used to investigate a fixed number of time series for grained analysis. Least mean square filter is used to predict the next value of the time series based on previous values, and Pearson Chi-square divergence is used to measure the deviations between the current and estimated probability distributions. We evaluate our approach using publicly available real IP traces (MAWI) collected from the WIDE backbone network, on trans-Pacific transit link between Japan and USA. Our experimental results show that the proposed approach outperforms existing techniques in terms of detection accuracy and false alarm rate. It is able to detect low intensity attacks covered by the large number of traffic in high speed network.**

***Index Terms*—Intrusion Detection System; Anomaly detection; DDoS; SYN flooding; BOTNET; Sketch; Least Mean Square; Chi-Square divergence.**

I. Introduction

Flooding attacks overwhelm web servers with connection requests in order to prevent access for legitimate users. With the use of BOTNET (roBOT NETworks), Distributed Denial of Service (DDoS) is able to make silent any web site. Recently, PayPal has been driven offline for eight hours after dropping WikiLeaks donations (operation PayBack in 2010), where it was easy for small group of users to have a large impact on the functioning of major web sites. Flooding attacks have increased significantly in the last decade, and the list of web server victims of DDoS through TCP SYN flooding is large: WikiLeaks, Twitter, CNN, Yahoo, Amazon, Ebay, etc.

TCP SYN flooding attack is on the top of the list of flooding attacks, especially with the use of BOTNET containing large number of compromised hosts (zombies). Usually, an attacker may compromise many computers through exploiting existing vulnerabilities, and a novice attacker rent a BOTNET with few dollars on the web, to launch distributed attack with the help of compromised computers.

Flooding attacks need to be accurately detected in real time, in order to cope with ongoing attack as soon as possible. Several techniques aimed to detect intrusion have been proposed and tested in the last decade. Signature based IDS, like Bro [1] and Snort [2] check payload of traffic for matching attack signature. This approach is not scalable for the backbone networks. On the other hand, anomaly based approach consists of monitoring network traffic to investigate and detect abnormal deviations from dynamically updated model. Such deviations are mostly direct indications of abnormal situations. The real time analysis for early detection of these anomalies allows reacting to prevent network deterioration and resources exhaustion.

In this paper, we focus on volume based attacks, such as flooding for DDoS. Our approach to detect flooding is based on anomaly detection, since flooding attacks will change some statistical metrics of traffic. With the distributed nature of these attacks, detection and reaction must be pushed to core network (backbone or Autonomous Systems), or near to the sources of attack. However, with the complexity in analyzing huge amount of data traffic from backbone network, the analysis of each traffic flow is not scalable and computationally expensive.

Many change point detection algorithms have been proposed and applied to the time series resulted from the aggregation of whole network traffic in one flow [3], [4], [5]. Therefore, these anomaly detection methods are based on the identification of the change point where heavy deviation occurs in the resulted time series. However, as these methods aggregate whole traffic in one time series, attacks can easily be buried in background traffic and pass undetected, i.e. flooding attack with intensity 10^6 packets/s does not produce a noticeable deviation when the total number of packets is greater than 10^9. Moreover, time series derived from IP traffic are subject to many variations that are irrelevant to anomaly. The time series is non-stationary and tends to change over time, leading to a lot of false alarms. Reducing the false alarms and increasing the detection accuracy in such methods are a challenging problem.

In this paper, we propose a new framework for anomaly detection over high speed network. Our proposed framework is aimed to resolve the issues of aggregating whole traffic in one time series in order to improve the detection of low intensity attacks. Thus allow to uncover hidden attacks by the large variation resulted from aggregating the whole traffic, and to distinguish attacks from normal traffic variations. We aim also to resolve the problem of static threshold and to reduce the false alarms ratio.

978-1-61284-927-0/10/$26.00 ©2011 IEEE

The proposed framework is based on random aggregation of traffic in Sketch [6], [7], [8] data structure for a discrete time interval T. At the end of each interval, the Least Mean Square (LMS [9]) filter is used to forecast the next values in each time series from previous values, and the Chi-square [10] (χ^2) divergence is used to measure the distance between the estimated and current probability distributions.

The Sketch data structure is an array of hash table, where each cell contains a shared counter for specific monitored attribute (number of: bytes, packets, SYN, etc.). Each counter belongs to network traffic having the same hash value of specific fields in packet header (e.g. destination IP address & port, source IP address & port, protocol, etc.). Sketch is used to reduce the number of time series through random aggregation of network traffic. The benefit of sketch is twofold. First, it resolves the problem of hidden attacks in the large variations of aggregated traffic, and allows more grained analysis for the detection of low intensity attacks. Second, it provides a fixed number of time series to investigate. Furthermore, we use Sketch data structure to establish a probabilistic model by exploiting the counters of hash table.

The value of counters resulted from different interval are used for anomaly detection. The least mean square filter is used to estimate the current value of each counter from some previous values. Afterward, the χ^2 divergence between the estimated and current distributions of monitored attribute is used to detect anomaly.

We have applied our anomaly detection algorithm on publicly available real data trace [11]. Our experimental results show that our proposed approach achieves good performance in term of detection accuracy and false alarm ratio, especially when comparing the result with existing methods.

The rest of this paper is organized as follow. In section II, we present some references related to our work. In section III, we briefly review background materials related to the proposed framework. Section IV gives a detailed description of our proposed approach for anomaly detection. Section V presents the experimental results of the proposed approach. Finally, some concluding remarks are given in section VI.

II. Related work

Anomaly detection in data networks resulted from distributed monitoring sensors is primordial for network security. Many interesting previous work on efficient and online change point detection algorithms have been proposed and tested for DDoS detection, such as Haar-wavelet analysis [12], [13], change point detection methods with the CUmulative SUM (CUSUM) and its non-parametric version [3], [4], [14], adaptive threshold analysis [15], [3], Exponentially Weighted Moving Average (EWMA) [5], Holt-Winters seasonal forecasting [16], [8], [17], ARIMA [7], Heavy Hitter [18], [6], [19], SNMP [20] (Simple Network Management Protocol) MIB statistical data analysis, combination of adaptive threshold, CUmulative SUM and the source IP monitoring algorithms [21], etc.

Statistical approaches establish a profile of normal traffic during a training period, and deviations from the established profile are considered as anomaly. Usually, malicious activities provoke an abrupt change in the statistical values of the parameters describing the traffic (number of packets, bytes, SYN, etc.), e.g. NetScan produced by worms outbreak, that send a large number of SYN from the same source IP, to scan the network before propagation phase.

In [22], the CUSUM algorithm is used to detect SYN flooding over one time series resulted from aggregation the whole traffic in one flow. In [3] a comparison between CUSUM and adaptive threshold for the detection of SYN flooding is presented. These proposed approaches aggregate the whole traffic in one time series for dimensionality reduction, and attacks can easily be buried inside background traffic in backbone network. However, it is complex and intractable to investigate a time series per flow over backbone networks, but more grained analysis than aggregating the whole traffic in one time series is required.

Furthermore, when deviations are larger than a predefined threshold in previous approaches, the observation (e.g. number of packet) is considered as an outlier, and an alarm is triggered. Such approaches cannot adapt the temporal variation in the traffic, and a fixed threshold for different kind of networks is not suitable parameter, especially with the large variations in the traffic. It may induce large ratio of false alarm and miss detection. In this paper, we want to overcome these problems through the use of Sketch for more grained and scalable analysis, and we want to propose a solution for dynamically adjusting the threshold.

The Principal Component Analysis (PCA) method for anomaly detection in [23], [24] transforms high dimensional space into low space, and detects flow anomalies using the evaluation of flows correlations over single link, instead of single time series of whole traffic from many links. Authors in [25] show that Euclidean and commute distances are more stable and less sensitive than PCA method. The presented results in [25] show that PCA is incapable of detecting large anomalies. In [26], the authors showed that methods for tuning PCA are not adequate and starting with a new data set, adjusting parameters is unexpectedly difficult.

Sketch data structure [7], [27] is used as a dimensionality reduction technique. It uses the random aggregation to summarize monitored traffic in a fixed memory, and to provide scalable input for online anomaly detection. Different type of counters have been used for detecting change in traffic features, such as the number of: SYN, packets, flows, bytes, etc. For example, the number of SYN per destination IP address can be used to detect SYN flooding, since distributed attacks are directed toward unique victim.

In [8], a reversible sketch is proposed for detecting and identifying malicious traffic. In [28], Sketch has been used in HiFIND framework to detect and distinguish between many classes of attacks.

In this paper, our proposed framework extends all these previous works, through the use of LMS filter to predict the

next value of the time series, and the χ^2 divergence over Sketch, and dynamic threshold for anomaly detection. The method can be used to detect any type of flooding (UDP, ICMP, SYN, ACK, etc.). We will present the method for SYN flooding attack detection in this paper for the sake of simplification. However, the same procedure can be applied to detect any type of flooding attacks.

Under SYN flooding attack, the distribution of number of SYN toward a specific IP address will deviate from previously learned distribution under normal traffic condition. However, with the difficulty of finding a distribution probability that fits to traffic characteristics (self similarity, heavy-tailed, and long range dependence, etc.), we will use the χ^2 divergence to detect deviations between the probability values resulted from the shared counters of Sketch in current time interval, and the estimated values of each counter in the time series.

III. Theoretical background

In this section, we review the Sketch data structure, the least mean square filter and the χ^2 divergence. They are relevant to understand the proposed framework.

A. K-ary Sketch

K-ary Sketch S is an array of hash tables used to randomly aggregate large number of flows into a fixed size of memory. It is a two dimensional $L \times C$ array, with L mutual independent hash functions. Each row is associated with a hash function. Let $A = a_1, a_2, \ldots, a_n$ denotes the set of arrival, where each arrival $a_i = (\kappa_i, \nu_i)$ is identified by a key κ_i and its associated value ν_i. For example, during SYN flooding attack, high number of SYN requests are directed to the victim server. Therefore, we use the destination IP (DIP) address as a key $\kappa_i = DIP$, and $\nu_i = \#SYN$ to count the number of SYN received by destination. The arrival of a packet with key κ_i increments its associated counter in the j^{th} hash table by ν_i ($S_{j,h_j(\kappa_i)} += \nu_i$), as shown in algorithm 1 and in Fig. 1. The hash functions are chosen from the set of 2-universal hash functions $h_j(\kappa_i) = \{((a_j\kappa_i + b_j) \bmod P_U) \bmod C\} + 1$, to uniformly distribute keys (κ_i) over hash table and to reduce collisions. Collisions are resulted from two or many items having the same hash value:

$$\forall \kappa_i, \kappa_j \in U,\ \kappa_i \neq \kappa_j : h_k(\kappa_i) = h_k(\kappa_j) \tag{1}$$

The parameter P_U is a prime number larger than the maximum number in the universe, where Mersenne prime numbers of the form $2^i - 1$ are generally chosen for fast implementation. For κ_i on 32 bit, we use $P = 2^{61} - 1$. a_j and b_j are random integers smaller than P_U, with $a_j \neq 0$. Using L hash functions, the probability that two keys are aggregated in the same bucket over the L hash tables is $(1/C)^L$.

As the size of DIP (IPv4) is 32 bits, the hash functions reduce the dimension of monitored space (2^{32}) to a fixed size w (e.g. $w = 2^{10} = 1024$), through the random aggregation of multiple IP addresses in the same bucket, when the value resulted from hashing the addresses are the same ($h_j(DIP_1) = h_j(DIP_2) = K$). For $L = 5$, the probability that 2 flows share the all counters over the L hash table is 0.88×10^{-15}, and attacks can not be covered by the variations of normal flows sharing the same cell over the L lines in the Sketch. Usually, flows sharing the same cell in the first hash tables, have less probability to share the same cell in other hash tables.

Algorithm 1 Sketch Update procedure

```
for all TCP SYN segment received during T do
    for k = 1 to L do
        j = univ_hash_k(κ_i);
        S[k][j].counter+ = ν_i;
    end for
end for
```

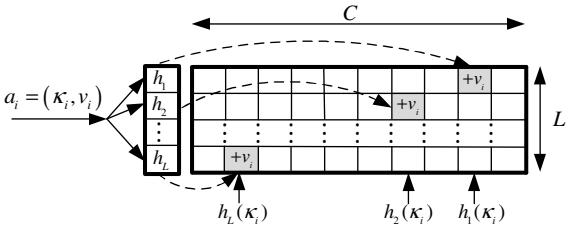

Fig. 1. Sketch data structure.

B. Least Mean Square

LMS estimates the next value $\hat{x}_{n+1}$ in the times series from known previous values $x_1, x_2, \ldots, x_n$. It attempts to minimize the error between the measured value x_{n+1} and the estimated value $\hat{x}_{n+1}$ using a gradient-based method of steepest decent [9]. The predicted value in the n^{th} step is calculated using Eq. 2, which depends on the weight vector W and a vector of previous values X_N. The vector W is initiated randomly, and the vector X_N contains the last N measured values from the time series (N is the filter order).

$$\hat{x}_{n+1} = W_n^T . X_N \tag{2}$$

LMS uses an iterative algorithm based on the least mean square for the correction of the weight vector W, as given in Eq. 3:

$$W_{n+1} = W_n + \mu.(x_n - \hat{x}_n).X_N \tag{3}$$

Where μ is a step size parameter that controls the convergence characteristics, and must be chosen with respect to the condition in Eq. 4 for fast convergence.

$$0 < \mu < \frac{2}{\lambda_{\max}} \tag{4}$$

Where $\lambda_{\max}$ is the largest eigenvalue of the matrix $R_n = X_n.X_n^T$. However, R_n is symmetric and the largest eigenvalue is the trace of the matrix. Therefore, μ is updated online as given in Eq. 5:

$$\mu = \frac{1}{2 \times trace(R)} \tag{5}$$

C. Chi-square divergence

χ^2 divergence is used to measure distance between two discrete probability distributions (P and Q). For 2 probability sets $P = (p_1, p_2, p_3, \ldots, p_n)$ and $Q = (q_1, q_2, q_3, \ldots, q_n)$, with $p_i \geq 0$, $q_i \geq 0$ and $\sum_{i=1}^{n} p_i = \sum_{i=1}^{n} q_i = 1$, the Pearson χ^2 divergence between P and Q is given by:

$$\chi^2 (P||Q) = \sum_{i=1}^{n} \frac{(p_i - q_i)^2}{q_i} \tag{6}$$

Where Q is the estimated probability distribution, and P is the measured probability distribution, and $\chi^2(P||Q)$ is the distance between distributions P and Q.

For hypothesis testing, such as H_0 (normal traffic hypothesis) and H_1 (traffic with anomalies), χ^2 values can run from zero into infinity. χ^2 will be zero iff P and Q are identical ($p_i = q_i$) under hypothesis H_0, and χ^2 increases as the distributions become dissimilar, and eventually so high (infinity) when the two distributions are independent ($P \neq Q$) under hypothesis H_1. It is important to note that χ^2 divergence is nonnegative and the division $0/0$ is treated as 0, and the division by zero is replaced by a very small value ε [29].

The χ^2 divergence between 2 probability distributions P and Q must be near zero under normal traffic, with a large deviation (one spike) when distributions change occurs. χ^2 is asymmetric $\left(\chi^2 (P||Q) \neq \chi^2 (Q||P)\right)$, and its symmetric version (given in Eq. 7) raises two spikes. One spike at the beginning and the second at the end of the attack.

$$\chi^2 (P||Q) + \chi^2 (Q||P) = \sum_{i=1}^{n} \frac{(p_i - q_i)^2 (p_i + q_i)}{p_i . q_i} \tag{7}$$

We intend to use Pearson chi-square divergence (asymmetric) to detect anomaly through the detection of deviations from normal traffic profile, and we will modify the input time series to constrain χ^2 to raise alarms (spikes) for the whole duration of attack. In [30], authors prove that χ^2 divergence behaves better than all classical divergences (Hellinger distance, Kullback-Leibler, Likelihood, etc.). In our proposed approach, we use χ^2 divergence with dynamic threshold to increase the detection accuracy, and to reduce the false alarm ratio.

IV. Proposed approach

The proposed approach for anomaly detection in backbone networks is based on Sketch, LMS filter and χ^2 divergence. The detection system records the number of monitored attribute (e.g. #packets, #SYN, #flows, etc.) in the Sketch for each discrete time interval T. Random aggregation of traffic flows in Sketch is the first step of our processing, followed by time series forecasting with Least Mean Square, and change detection with χ^2 divergence (Fig. 2).

In fact, each cell in Sketch is a counter for randomly aggregated flows during T (e.g. $T = 1$ min). The counters in 2D table is used to record the number of SYN. Each cell may contain many counters for detecting different type of flood (TCP, UDP, ICMP, SIP INVITE, etc.). For the sake of simplicity, we will restrict our analysis in this paper to TCP SYN flooding by counting only the number of SYN, but the same method can be applied to detect different type of flooding attacks. Many web services are based on TCP, and SYN flooding is the most used flooding attacks in these days.

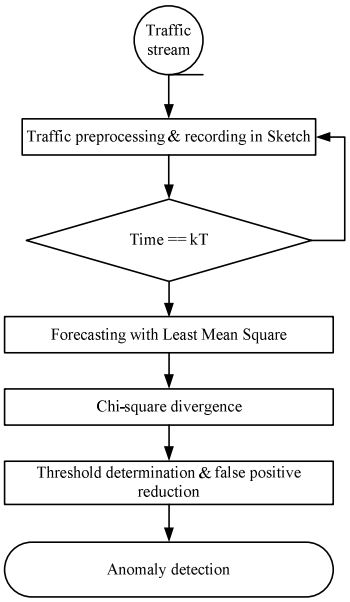

Fig. 2. Architecture of the proposed approach for network anomaly detection.

During SYN flooding attack, the number of SYN directed to victim server increases significantly. The destination IP (DIP) address is used as key in the update procedure ($\kappa_i = DIP$ in algorithm 1) to derive the colon index, and the associated value ν_i is equal to 1 for SYN segment and zero otherwise. Therefore, we can detect anomalies targeting any host regardless the number of the involved sources (one source or geographically distributed sources).

At the end of each epoch T, the LMS filter predicts the next values by using the previous N counters in the time series, and adjusts its forecasting with respect to the error between the estimated and obtained values. To predict the next value of each counter, we use the first few minutes as training phase, and a fixed size sliding window of N (filter order) time slot, to adjust the weight vector W, and to predict the next value of each counter.

Afterward, we derive two probability distributions from the estimated and measured number of SYN. This is done by dividing each counter in one hash table by the sum of whole counters in this table (Eq. 8). The probability $p_{i,j}$ in each cell is given by:

$$p_{i,j} = S_{i,j}.counter / \sum_{j=1}^{C} S_{i,j}.counter \tag{8}$$

Sketch data structure holds a vector X_N in each bucket of the 2D array. Each cell $S_{i,j}$ becomes a data structure, that contains: an array of fixed size for the last N counters (X_N),

two counters for current and estimated value of observations, and 2 float variables for holding the associated probabilities.

Thus result in $2 \times L$ distributions (P_i, Q_i with $i \in [1, L]$), where P_i is the probability distribution $(p_{i,1}, p_{i,2}, \ldots, p_{i,C})$ resulted from the i^{th} hash table, and Q_i is the estimated probability distribution. The χ^2 divergence between the current (P_i) and estimated probability (Q_i) distributions is calculated for each line in the sketch, at the end of each time interval (i.e. at kT). L values of $\chi^2_{i,kT}$ will be calculated (one per line). During malicious activities, many time series $\chi^2_{i,kT}$ induce one spike, and when more than H ($H \leq L$) values of $\chi^2_{i,kT}$ exceed a dynamic threshold for more than η consecutive intervals, an alarm is raised.

The waiting time η is used to avoid false alarms due to normal traffic variations (e.g. flash crowd), and the fact that DDoS attack must span for many consecutive intervals to overload a server. Therefore, if the lifetime of the deviation is smaller than predefined number of time interval, no alarm will raise.

When an alarm is triggered, we halt the updating procedure of vector X_N until the end of attack, to prevent the poisoning of the forecasting procedure. This technique constrains χ^2 to produce spikes for the whole duration of the attack, and allow the detection of the start and stop instant of the attack.

Under normal traffic hypothesis (without anomaly), the first value of the array X_N will be dropped, and other values in this vector are shifted one position ($X_N[i] \leftarrow X_N[i+1]$), and the current counter is pushed to the end of the vector X_N. Afterward, the weight of LMS is updated, and the next value is estimated. When an alarm is triggered, the update procedure of X_N is halted until the end of ongoing attack.

When the attack stops, the $\chi^2_{i,kT}$ divergence values will drift back near to zero, and the vector X_N, used in the forecasting procedure, continues to be updated with the value in the current time interval.

To detect deviations in the time series $\chi^2_{i,kT}$, we derive a subsequent time series $\hat{\chi}^2_{i,kT}$ containing the values in the $\chi^2_{i,kT}$, but without spikes, i.e. only values smaller than dynamic threshold. In the time series $\chi^2_{i,kT}$, we use the dynamic bound of $\mu_{i,k-1} + \alpha\sigma_{i,k-1}$ given by Cheysheve inequality. At least $1 - 1/\alpha^2$ of the values of $\chi^2_{i,kT}$ are within α standard deviations from the mean. Significant deviations (outlier values) are larger than the dynamic bound, and are replaced by the last value in $\hat{\chi}^2_{i,kT}$ (e.g. $\hat{\chi}^2_{i,kT} \leftarrow \hat{\chi}^2_{i,(k-1)T}$). In contrast, inliers are included in the $\hat{\chi}^2_{i,kT}$ time series. Therefore, $\hat{\chi}^2_{i,kT}$ contains only the values of $\chi^2_{i,kT}$ that satisfy the equation:

$$\chi^2_{i,kT} < \mu_{i,(k-1)T} + \alpha\sigma_{i,(k-1)T} \tag{9}$$

Where $\chi^2_{i,kT}$ is the value of chi-square in the time slot kT for line i in the Sketch data structure, and $\mu_{i,k}$ & $\sigma_{i,k}$ are the mean and the standard deviation of $\hat{\chi}^2_{i,kT}$ respectively. $\mu_{i,k}$ and $\sigma_{i,k}$ are updated dynamically using the Exponentially Weighted Moving Average (EWMA):

$$\mu_{i,kT} = \beta\mu_{i,(k-1)T} + (1-\beta)\chi^2_{i,(k-1)T} \tag{10}$$

$$\sigma^2_{i,kT} = \beta\sigma^2_{i,(k-1)T} + (1-\beta)(\hat{\chi}^2_{i,kT} - \mu_{i,kT})^2 \tag{11}$$

The threshold is updated dynamically by adjusting the value of $\mu_{i,kT}$ and $\sigma_{i,kT}$ as shown in Eqs. 10 & 11. α is a parameter used for calibrating the sensitivity of the detection algorithm to variations, and to reduce the false alarm rate. High value of threshold reduces the false alarm rate, and increases the probability of misclassification of attack as normal traffic. The choice of the threshold (implicitly α) reflects a trade-off between misclassification (false negative) and the false alarm rate.

Under normal traffic, divergence $\chi^2_{i,kT}$ falls down the threshold ($\mu_{i,(k-1)T} + \alpha\sigma_{i,(k-1)T}$). When $\chi^2_{i,kT}$ exceeds the dynamically updated threshold over H lines (hash tables), an alarm is triggered. The decision function for alarms is given in Eq. 12. The alarm will not trigger before the value η exceeds a specified threshold used to reduce false alarm rate. Another interesting approaches for estimating and adjusting dynamic threshold were proposed in [31], [32] for SIP INVITE flooding detection.

$$d(A_i) = \begin{cases} 1 & \textit{if } \chi^2_{i,kT} \geq \mu_{i,(k-1)T} + \alpha\sigma_{i,(k-1)T} \\ & \textit{and } \eta \geq 3 \\ 0 & \textit{Otherwise} \end{cases} \tag{12}$$

Where $d(A_i)$ is the decision function for alarms, with a value 1 when an alarm must be triggered, and 0 otherwise. $\chi^2_{i,kT}$ is the chi-square divergence obtained from i^th hash table at time kT. $\mu_{i,(k-1)T}$ & $\sigma_{i,(k-1)T}$ are the mean and the standard deviation of the time series $\hat{\chi}^2$. η is a counter for the number of consecutive intervals where $\chi^2_{i,kT}$ exceeds the dynamic threshold. η is used to reduce false alarm ratio with the dynamic and unpredictable fluctuations of network traffic.

V. Experimental results

The validation of proposed approach is realized with publicly available IP traces over the web. We get the traces from MAWI [11] repository of WIDE project. We use daily traces in tcpdump format, collected at simplepoint-F which is a trans-Pacific link between Japan & USA, from 15/04/2010 07h30 to 23h as background traffic from real life of the internet. We analyzed 62 files with more than 50GB without application data. IP addresses in these traces are scrambled by a modified version of tcpdpriv [11] tool, but correlation between addresses are conserved. We have analyzed these traces of wide area network using the proposed approach, with $\kappa_i = DIP$ and $\nu_i = 1$ for SYN request only, and $\nu_i = 0$ otherwise. We found many existing anomalies as we show later. Afterward, we inject real SYN flooding DDoS attacks with different intensity inside these traces.

The numeric values of used parameters are: $C = 1024$ and $L = 5$, $N = 5$, $H = 3$, $\alpha = 3$, $\beta = 0.7$, $T = 1$. The time interval has an impact on detection delay, where a small value of T reduces the detection delay at the cost of potentially increasing the false alarm ratio. On the other hand, a large value of T interval increases the detection delay. One minute

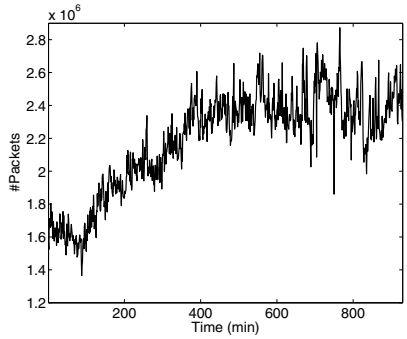

Fig. 3. Total number of packets.

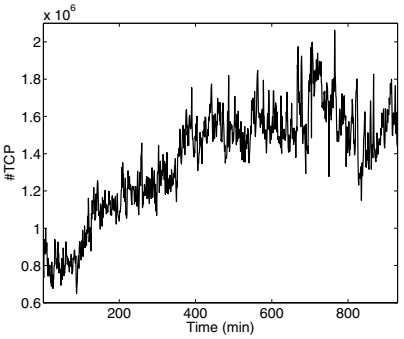

Fig. 4. Total number of TCP segments.

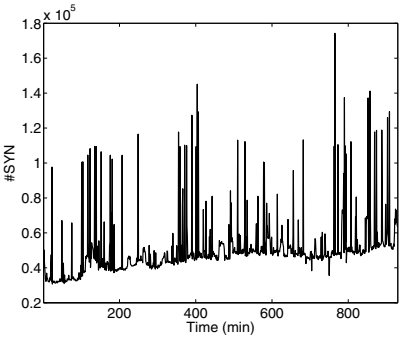

Fig. 5. Total number of SYN segments.

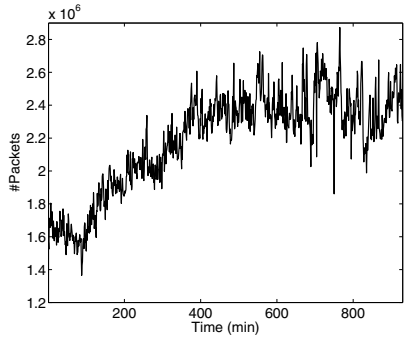

Fig. 6. Total number of packets after SYN flooding attacks.

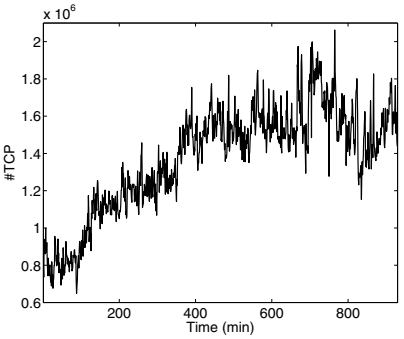

Fig. 7. Total number of TCP segments after SYN flooding attacks.

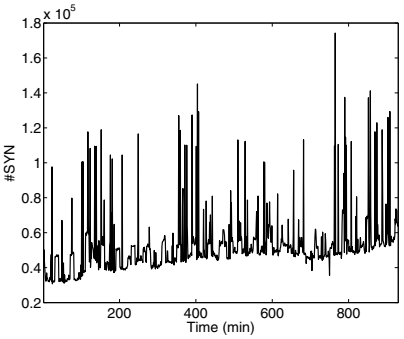

Fig. 8. Total number of SYN segments after SYN flooding attacks.

is chosen as a tradeoff between detection delay and false alarm ratio.

Computational complexity and required memory storage are central issues for online anomaly detection. The processing time for whole traces on Ubuntu box with 2.2 GHz Intel Core 2 Duo CPU & 3 GB of RAM is about 1072s.

With the use of Sketch, the spatial complexity is $O(C \times L)$ cells, where each cell contains an array of integer with size N ($4 \times N$ Bytes), and two integer variables for current and estimated values of observations (8 Bytes), and two float variables for current and estimated probabilities (8 Bytes). Thus, the total required memory with used parameters in our experiments is 44.8 KBytes. The required memory space for Sketch data structure is fixed and independent of the amount of traffic that must be handled.

We begin by applying our approach over MAWI traces to detect existing anomalies. χ^2 divergence raised many spikes and we extract malicious flows as we will show later. Afterward, to analyze the detection accuracy of our proposed approach, we inject 23 DDoS TCP SYN flooding attacks with different intensity in these traces. The injected attacks are presented in Fig. 14. Attacks duration are 10 min and separated by 30 minutes.

The variations of the total number of packets before and after attacks are given in Fig. 3 & Fig. 6 respectively. Attacks cannot be detected when inspecting the variation of total number of packets. In fact, there are no heavy deviations in the time series describing the number of packets when comparing Fig. 3 & Fig. 6, because the aggregation of attacks with the high number of background packets, and the subsequent low variations induced by injected attacks.

Also, the injected SYN flooding attacks don't provoke any obvious variations when comparing the total number of TCP segments before and after attacks injection, as shown in Fig. 4 & Fig. 7 respectively. In fact, the number of TCP segments is around 1.4×10^6 and attacks (shown in Fig. 14) are of order 10^4 (relative insignificant variations). Therefore, the induced variations by attacks can not be observed when comparing Fig. 4 & Fig. 7 (same similarity when comparing Fig. 3 & Fig. 6).

In Fig. 5, we present the variation of total number of SYN in the original trace (before attacks injection), where we can observe significant variations with time, and large deviations in the number of these requests. In spite of these heavy deviations, we found, after inspection, some of these variations are legitimate, where SYN requests are not directed towards a specific destination. The analysis of the aggregation of whole SYN segments in one time series leads to lot of false alarms, whereas the grained analysis with Sketch doesn't produce false alarms unless a deviation in the number of SYN requests received by a destination.

To inspect these variations in the number of SYN, we apply the proposed approach over the original IP dataset. We conduct many experiments by incrementing the value of η from 1 to 10. Fig. 9 & Fig. 10 show the number of anomalous SYN received by given destinations, and Fig. 11 shows the

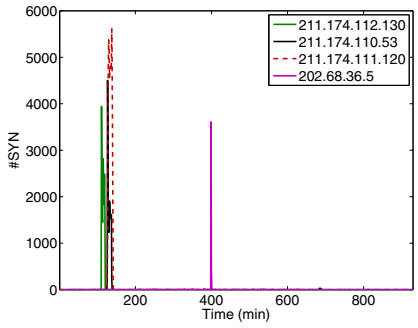

Fig. 9. #SYN received by given DIP.

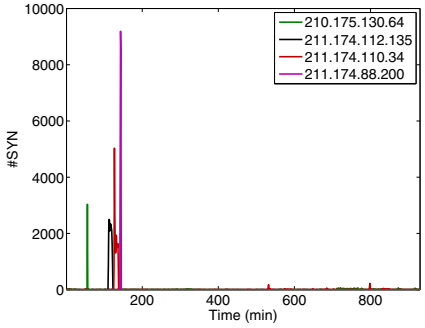

Fig. 10. #SYN received by given DIP.

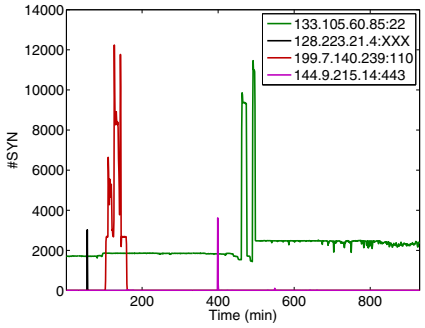

Fig. 11. #SYN sent by given SIP.

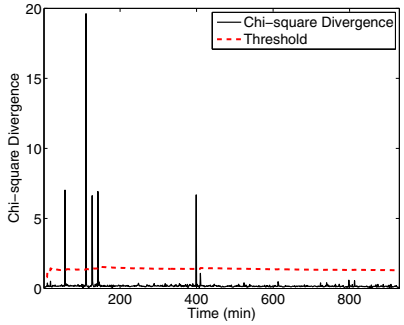

Fig. 12. Basic χ^2 divergence between current & estimated values with $\eta = 1$.

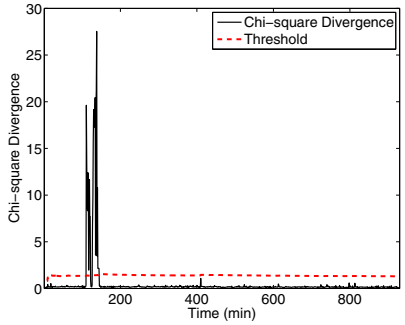

Fig. 13. χ^2 divergence with $\eta = 3$ & halt forecasting for continuous alarms.

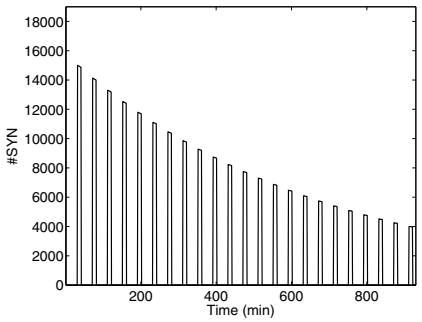

Fig. 14. Injected SYN flooding attacks with different intensity.

number of anomalous SYN generated by given sources for PortScan & SYN flooding. These anomalies are extracted from original traces, after the application of χ^2 divergence on the estimated and current distributions resulted from Sketch. We get 5 alarms for flooding attacks as shown in Fig. 12. Due to space limitation, we present only the variations of χ^2 on the first hash table, where other lines exhibit very similar variations. If less than H values of chi-square divergence are higher than the dynamic threshold, no alarm is raised.

Raised spikes between 111 & 142 are the result of abnormal (large) number of SYN sent from the same source to 3 servers. In fact, we found that the raised alarm at instant 56 is the result of PortScan with SYN, and the instant 399 is caused by large number of SYN (sent from 144.9.215.4 to the port 443) for 1.5 minutes. To reduce false alarms, we set $\eta = 3$ in our experiment to prevent χ^2 from raising alarms for such small time variations. It is important to note, that increasing the value of η will increase the detection delay, and a tradeoff between accuracy and detection delay is required. Furthermore, to continuously raise alarms during attack, we stop the forecasting procedure at the start instant and until the end of attack. The result of these modifications (halt updating and $\eta = 3$) on the chi-square divergence is shown in Fig. 13.

The variation of SYN segments after the injection of SYN flooding attacks is shown in Fig. 8. When comparing Fig. 5 & Fig. 8, which represent the variations of SYN requests before and after attacks injection, we may not notice the small variations induced by attacks unless a deep inspection. In fact, the underlying changes invoked by injected attacks (Fig. 14) are smoothed by the large fluctuations of backbone traffic. These attacks cannot be detected by methods based on analyzing one time series, which results from aggregating the whole traffic, such as single channel CUSUM and adaptive threshold used in [3], [4].

These methods identify attacks as heavy change in the analyzed time series, and when applied on the aggregation of whole traffic to reduce computational complexity and to alleviate scalability problem (one time series per destination IP), they cannot detect low intensity attacks. The proposed approach resolves the problems of scalability, temporal complexity and detection accuracy of low intensity attacks, through analyzing a fixed number of time series resulted from randomly aggregating traffic flows in Sketch data structure.

As we want to compare the efficiency of Chi-square divergence with widely used divergence measures, we implement 3 distance measures over Sketch: Chi-square Divergence [29], Hellinger Distance (HD [31], [32]) and Jensen-Shannon Divergence (JSD [29]). In contrast to χ^2, HD & JSD are symmetric and raise two spikes: one at the start instant and the second at the end of the attack.

The results of the application of these measures over traces with attacks are given in Fig. 15 & Fig. 16 & Fig. 17. These figures are the results of comparing only the predicted value with the current observation (without halt updating technique). HD and JSD raise two spikes for each inserted attacks, and they achieve a very good detection, but with many additional spikes (false alarms) with respect to χ^2, by comparing Fig. 16 & Fig. 17 with the result of chi-square in Fig. 15. χ^2

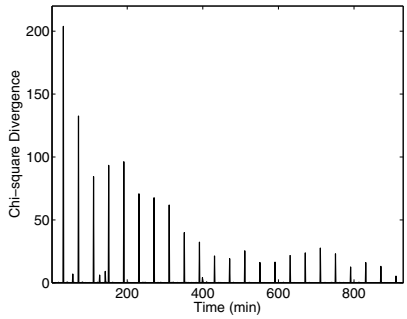

Fig. 15. χ^2 Divergence with $\eta = 1$ after the injection of SYN flooding attacks.

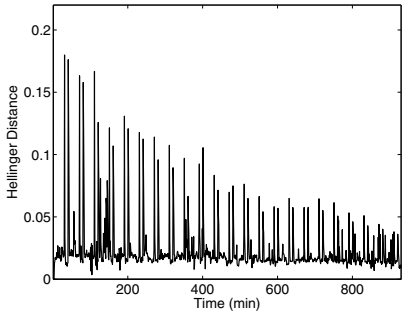

Fig. 16. Hellinger Distance after the injection of SYN flooding attacks.

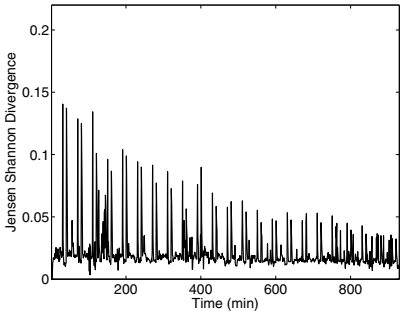

Fig. 17. Jensen-Shannon Divergence after the injection of SYN flooding attacks.

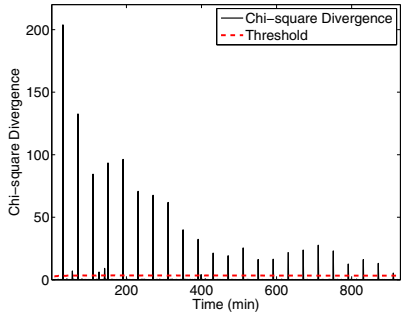

Fig. 18. χ^2 Divergence with $\eta = 1$ and dynamic threshold.

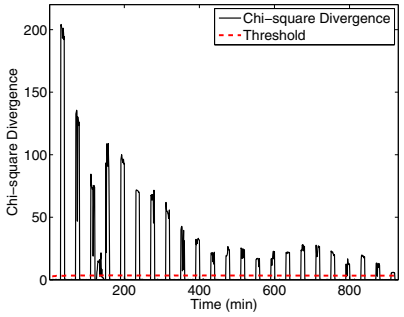

Fig. 19. χ^2 Divergence with $\eta = 3$ and modified input.

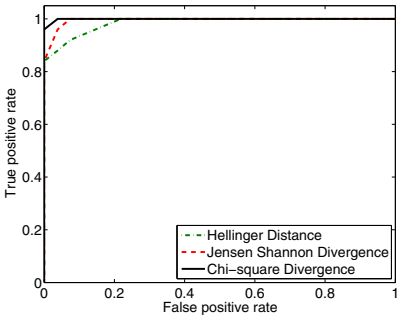

Fig. 20. Receiver Operating Characteristic for χ^2, HD & JSD.

divergence performs better detection with less false alarms. In Fig. 15, we can see that the intensity of raised spikes by χ^2 are proportional to the intensity of the attacks, where high intensity attack produces high spike, and thus can be exploited for classifying attacks by severity. We can distinguish clearly the performance of χ^2 in terms of reduced false alarms, and simple required threshold for detecting all attacks.

We add the dynamic threshold given in Eq. 9 on the χ^2 divergence for both case: one alarm and continuous alarms on attack detection. Fig. 18 & Fig. 19 show the results of our investigations for both cases respectively. In fact, when the current value of Chi-square is larger than dynamic threshold, an alarm is triggered.

To evaluate the performance of the proposed approach, we investigate the detection ratio (true positives) and the proportion of false alarms (false positives). These indicators are used by ROC (Receiver Operating Characteristic) for accuracy analysis, when varying the value of threshold. The detection rate is defined as the ratio of detected attack to the number of existing attacks:

$$DR = \frac{TP}{TP + FN} \tag{13}$$

Where TP is the number of true positive alerts, and FN is the number of false negative. The false alarm rate is defined as the ratio of false alarms to the number of raised alarms:

$$FAR = \frac{FP}{TP + FP} \tag{14}$$

Where FP is the number of false positive alerts. A high true positive and a low false alarm rate are desired to achieve good performance. We compare the performance of 3 divergence measures over Sketch: χ^2, (HD) and (JSD). Fig. 20 shows that the ROC curves (DR versus FAR) when varying the value of α in the threshold. With the use of dynamic threshold, and the halt technique of forecasting after the detection of attack, HD achieves a DR=100% with a FAR of 22%, JSD achieves a DR=100% with FAR=7.4%, and χ^2 achieves a DR=100% with FAR=3.8%. Thus, when comparing the performance of these algorithms, we found JSD achieves better than HD, and Chi-square outperforms both measures for anomaly detection with the lowest FAR. χ^2 divergence leads to very attractive performance in terms of detection rate & false positives ratio.

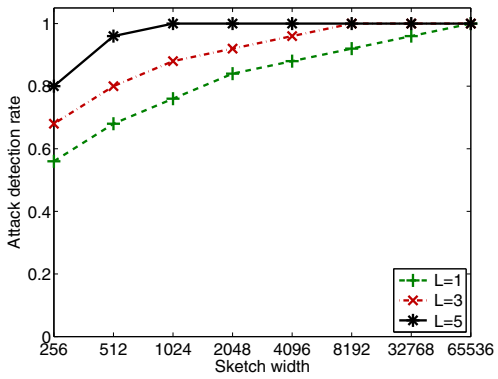

Fig. 21. Sketch width vs. detection rate.

Finally, we investigate the impact of varying Sketch parameters (width C and depth L) on the attack detection rate of

the proposed approach. Fig. 21 shows the variation of attack detection rate when increasing the Sketch width and depth. We conduct 3 experiments with $H = 1$, 2, 3 for Sketch depth $L = 1$, 3 and 5 respectively.

The attack detection rate increases with the Sketch width C (Fig. 21) at the cost of computational complexity and required memory. For $L = 3$ and $C = 2^{10}$, the attack detection rate is 88%, whereas, for $C = 2^{13}$, the detection rate is 100%. On the other hand, an high value of C increases the detection rate, but incurs sparse Sketch, per flow tracking and computational overhead. A more practical solution is to increase the Sketch depth (L), which increases the detection accuracy with lower cost than increasing Sketch width. In online deployment of Sketch for anomaly detection, a tradeoff between detection accuracy and temporal complexity is desired to take decision in less than T.

VI. Conclusion and perspectives

In this paper, we presented a new sequential approach for network anomaly detection in traffic of high speed networks. The proposed approach is based on random aggregation of flows in Sketch data structure. It uses LMS filter & Chi-square divergence to detect network anomalies. The experimental results on real IP datasets prove the effectiveness of the proposed approach in detecting low intensity attacks in massive data flows with high accuracy.

Ongoing and future work will focus on improving the proposed approach to identify and extract malicious flows related to the detected change. Thus provide valuable information and allow an automatic reaction to prevent ongoing attacks. Also, we seek to exploit the proposed anomaly detection method, to automatically generate signatures for newly detected attacks. Signature generation is one of our interests for future work.

References

[1] V. Paxson, "Bro: A System for Detecting Network Intruders in Real-Time," in *Computer Networks*, vol. 31 (23–24), 1999, pp. 2435–2463.

[2] M. Roesch, "Snort - Lightweight Intrusion Detection for Networks," in *Proceedings of the 13th USENIX conference on System administration (LISA '99)*, 1999, pp. 229–238.

[3] V. A. Siris and F. Papagalou, "Application of Anomaly Detection Algorithms for Detecting SYN Flooding Attacks," in *Proceedings of IEEE Global Telecommunications Conference (GLOBECOM'04)*, vol. 4, Dallas, USA, 2004, pp. 2050–2054.

[4] H. Wang, D. Zhang, and K. G. Shin, "SYN-dog: Sniffing SYN Flooding Sources," in *Proceedings of the 22th International Conference on Distributed Computing Systems (ICDCS'02)*. Washington, DC, USA: IEEE Computer Society, 2002, pp. 421–429.

[5] N. Ye, S. Vilbert, and Q. Chen, "Computer Intrusion Detection through EWMA for Autocorrelated and Uncorrelated Data," *IEEE Transactions on Reliability*, vol. 51, no. 1, pp. 75– 82, March 2003.

[6] M. Charikar, K. Chen, and M. Farach-Colton, "Finding Frequent Items in Data Streams," in *Proceedings of the 29th International Colloquium on Automata, Languages and Programming (ICALP '02)*. London, UK: Springer-Verlag, 2002, pp. 693–703.

[7] B. Krishnamurthy, S. Sen, Y. Zhang, and Y. Chen, "Sketch-based Change Detection: Methods, Evaluation, and Applications," in *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement (IMC'03)*, New York, NY, USA, 2003, pp. 234–247.

[8] R. Schweller, Z. Li, Y. Chen, Y. Gao, A. Gupta, Y. Zhang, P. Dinda, M.-Y. Kao, and G. Memik, "Reversible Sketches: Enabling Monitoring and Analysis Over High-Speed Data Streams," *IEEE/ACM Transactions on Networking*, vol. 15, no. 5, pp. 1059–1072, Oct. 2007.

[9] S. Haykin, *Adaptive Filter Theory*, 4th ed. Prentice Hall, 2001.

[10] I. J. TANEJA, "Bounds on Triangular Discrimination, Harmonic Mean and Symmetric Chi-Square Divergences," *Journal of Concrete and Applicable Mathematics*, vol. 4, no. 1, pp. 91–111, 2006.

[11] "MAWI working group traffic archive," http://mawi.wide.ad.jp/mawi/.

[12] W. Lu and A. A. Ghorbani, "Network Anomaly Detection Based on Wavelet Analysis," *EURASIP Journal on Advances in Signal Processing*, pp. 1–16, 2009.

[13] S. Siripanadorn, W. Hattagam, and N. Teaumroong, "Anomaly Detection in Wireless Sensor Networks using Self-Organizing Map and Wavelets," *International Journal oF Communications*, vol. 4, no. 3, pp. 291–297, 2010.

[14] A. Tartakovsky, B. Rozovskii, R. Blazek, and H. Kim, "Detection of Intrusion in Information Systems by Sequential Chang-Point Methods," *Statistical Methodology*, vol. 3, no. 3, pp. 252–340, 2006.

[15] S. Bu, R. Wang, and H. Zhou, "Anomaly Network Traffic Detection Based on Auto-Adapted Parameters Method," in *Proceedings of the 4th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM'08)*, 2008, pp. 601–607.

[16] J. D. Brutlag, "Aberrant Behavior Detection in Time Series for Network Monitoring," in *Proceedings of the 14th USENIX conference on System administration (LISA'00)*, Berkeley, CA, USA, 2000, pp. 139–146.

[17] H. A. Nguyen, T. V. Nguyen, D. I. Kim, and D. Choi, "Network Traffic Anomalies Detection and Identification with Flow Monitoring," in *Wireless and Optical Communications Networks(WOCN'08)*, 2008, pp. 1–5.

[18] G. Cormode, F. Korn, S. Muthukrishnan, and D. Srivastava, "Diamond in the Rough: Finding Hierarchical Heavy Hitters in Multi-Dimensional Data," in *Proceedings of the 23rd ACM SIGMOD*, 2004, pp. 155–166.

[19] G. Cormode and S. Muthukrishnan, "What's new: Finding significant differences in network data streams," in *Proceedings of IEEE Infocom*, 2004, pp. 1534–1545.

[20] J. Yu, H. Lee, M.-S. Kim, and D. Park, "Traffic Flooding Attack Detection with SNMP MIB using SVM," *Computer Communications*, vol. 31, no. 17, pp. 4212–4219, 2008.

[21] R. M. Mutebi and I. A. Rai, "An Integrated Victim-based Approach against IP Packet Flooding Denial of Service," *International Journal of Computing and ICT Research*, vol. 4, no. 1, pp. 70–80, 2010.

[22] H. Wang, D. Zhang, and F. Kang G. Shin, "Change-Point Monitoring for the Detection of DoS Attacks," *IEEE Trans. On Dependable and Secure Computing*, vol. 1, no. 4, pp. 1993–2004, 2004.

[23] A. Lakhina, M. Crovella, and C. Diot, "Diagnosing Network-Wide Traffic Anomalies," in *Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM'04)*, 2004, pp. 219–230.

[24] L. Huang, X. Nguyen, M. Garofalakis, and J. M. Hellerstein, "Communication-Efficient Online Detection of Network-Wide Anomalies," in *IEEE Conference on Computer Communications (INFOCOM'07)*, 2007, pp. 134–142.

[25] N. L. D. Khoa, T. Babaie, S. Chawla, and Z. Zaidi, "Network Anomaly Detection Using a Commute Distance Based Approach," in *International Conference on Data Mining Workshops (ICDW'10)*, 2010, pp. 943–950.

[26] H. Ringberg, A. Soule, J. Rexford, and C. Diot, "Sensitivity of PCA for Traffic Anomaly Detection," in *Proceedings of the ACM SIGMETRICS'07*, 2007, pp. 109–120.

[27] X. Li, F. Bian, M. Crovella, C. Diot, R. Govindan, G. Iannaccone, and A. Lakhina, "Detection and Identification of Network Anomalies using Sketch Subspaces," in *Proceedings of the 6th ACM SIGCOMM on Internet measurement (IMC'06)*, 2006, pp. 147–152.

[28] Z. Li, Y. Gao, and Y. Chen, "HiFIND: A high-speed flow-level intrusion detection approach with DoS resiliency," *Computer Networks*, vol. 54, pp. 1282–1299, 2010.

[29] S.-H. Cha, "Comprehensive Survey on Distance/Similarity Measures between Probability Density Functions," *Int. J. Math. Models Methods Appl. Sci.*, vol. 1, no. 4, pp. 300–307, 2007.

[30] M. Broniatowski and S. Leorato, "An estimation method for the Neyman chi-square divergence with application to test of hypotheses," *J. Multivar. Anal.*, vol. 97, pp. 1409–1436, 2006.

[31] H. Sengar, H. Wang, D. Wijesekera, and S. Jajodia, "Detecting VoIP Floods Using the Hellinger Distance," *IEEE Transactions on Parallel Distributed Systems*, vol. 19, no. 6, pp. 794–805, 2008.

[32] J. Tang, Y. Cheng, and C. Zhou, "Sketch-based SIP Flooding Detection using Hellinger Distance," in *Proceedings of the 28th IEEE conference on Global telecommunications (GLOBECOM'09)*, 2009, pp. 3380–3385.