# A Novel Approach for Anomaly Detection over High-Speed Networks

Osman Salem, Sandrine Vaton, and Annie Gravey

ENST Bretagne, Department of Computer Science,
Technopôle Brest - Iroise - CS 83818 - 29238 Brest Cedex 3 - France
{osman.salem,sandrine.vaton,annie.gravey}@enst-bretagne.fr

**Abstract.** This paper provides a new framework for efficient detection and identification of network anomalies over high speed links, in early stage of its occurrence to quickly react by taking the appropriate countermeasures. The proposed framework is based on change point detection in counters value of reversible sketch, which aggregates multiple data streams from high speed links in a stretched database. To detect network anomalies, we apply the cumulative sum (*CUSUM*) algorithm at the counter value of each bucket in the proposed reversible sketch, to detect change point occurrence and to uncover culprit flows via a new approach for sketch inversion. Theoretical framework for attacks detection is presented. We also give the results of our experiments analysis over two real data traces containing anomalies, and extensively analyzed in OSCAR French research project. Our analysis results from real-time internet traffic and online implementation over Endace DAG 3.6ET card show that our proposed architecture is able to detect culprit flows quickly with a high level of accuracy.

**Key words:** Network anomaly detection, Change point detection, Multi-chart Cumulative Sum, DoS, Sketch

## 1 Introduction

Security threats for computer network have increased significantly, which include viruses, worm-based attacks, PortScan, NetScan, denial of service (DoS), and its distributed version (DDoS), etc. However, with the increasing of link speeds and traffic volume, the real time monitoring and analyzing of IP traffic to detect attacks become a complicated task, but crucial for managing large networks (e.g. for ISP, Enterprise, etc.).

Two approaches to network anomaly detection are used. The first is signature based-approach, which is extensively explored in many software systems and toolkits such as Bro [17] and Snort [18]. This approach is used for anomaly detection with signatures known in advance, and it can not be applied to identify new anomalies.

The second approach is the statistics based approach, which does not require prior knowledge about the nature and properties of anomalies and therefore can be effective even for new anomalies or variants of existing anomalies. A very important component of statistics based approach is change detection [11]. It builds a model for normal user behavior in learning phase, and any inconsistent behavior with the build model is considered as anomaly. While wide range anomaly detection algorithms have been introduced to undermine attacks, the effectiveness of these models is largely dependent of traffic distributions parameters and their variations. They lack the capability of handling shape irregularities and unpredictable large fluctuations of real IP traffic.

Furthermore, most existing intrusion detection systems (IDS) reside at end host or end router. They are mostly host based or located on end routers. They lack scalability in handling large state space traffic information at high speed links, where even the handling at flow level is very costly in term of per-flow storage requirement, and update/search operations complexity. Flows are usually characterized by 5 fields (e.g. Netflow [2]): source and destination IP address, source and destination port, and protocol number. This means monitoring flows state space requires updating and handling a database table of size $2^{104}$.

A naïve idea for monitoring flows over high speed links is to maintain a database for active flows in a fixed time interval $T$, and to track the $k$ frequent destination addresses in IP traffic (top ten or heavy hitter destinations). For example, to detect victim servers of SYN flooding DDoS attack, an ISP can query database for destinations IP with a received number of SYN that exceeds a given percentage of the total number of SYN, which was relayed by the monitoring node. However, this strategy is not scalable, where spatial and temporal complexities, for update and query operations, on active flows database prevent its use for handling a large number of flows at high speed links in real time.

In response to these limitations, an efficient data structure based on $k-ary$ hash tables (Fig. 1), called sketch [1, 11, 12] was proposed and used to handle large state space, with a small amount of memory requirement and a linear computational (update/query) complexity. It is a multistage bloom filter based on random aggregations, where flow identifier (denoted by key) is mapped to index of bucket using $k$ different hash functions (one index per stage of the array of $k$ hash tables). These hash functions are generally chosen to reduce collision effect and to uniformly distribute keys over buckets of hash table.

To use sketch in context of network anomalies detection, IP flows can be classified by some combinations of fields in packet header, such as destination IP address ($DIP$), or source and destination IP address ($SIP/DIP$), etc. This flow identifier is used as key to update the $k^{th}$ hash table by its associated value ($key, value$). The value is a reward associated with key, and which can be the number of: packets, bytes, connection requests ($\#SYN$), number of half open connections ($\#SYN - \#SYNACK$), or other flow characteristics.

Recent work with Count-Min Sketch ($CMS$ [5]) has showed that random aggregation of flows does not significantly disrupt their variations. The $CMS$ query request algorithm can indicate if a given key exhibits large accumulated value, and even one can query sketch data structure about an approximate estimation of occurrence frequency for a given key.

However, sketch is based on universal hash functions, which are not reversible. Consequently, we cannot use sketch to report the set of frequent or heavy hitter keys, because sketch does not store any information about its key entry. Thus, the only way to get all heavy keys (which exhibit heavy accumulated value) is to test all possible entry, by hashing for the second time all keys, in order to determine those mapped to heavy buckets. That mean when monitoring high speed links, all keys must be recorded and verified. Unfortunately, this approach is neither scalable nor accurate for online monitoring.

In this paper, we consider the problem of online detection of network anomalies over high speed links, in order to cope with attacks as soon as possible. We propose a new variation of sketch by adding an inversion procedure for sketch, and we use a parametric version of multi-channel CUSUM (M-CUSUM [10, 21, 22]) as sequential algorithm [9] for anomaly detection in each bucket of sketch. Our contribution is twofold. First, we resolve the problem of sketch inversion with a software compliant procedure method, and second we analyse the efficiency of M-CUSUM (a sequential algorithm for anomaly detection) over this compact way for storing flows information. Sketch was used for offline anomaly detection through searching heavy hitter flows, and the use of sequential change point detection algorithm over sketch was never been addressed.

In fact, the combination of M-CUSUM and sketch improves the efficiency of the detection mechanism, where CUSUM is used to undermine anomalies, and sketch to significantly reduce the required memory and the computational complexity, when handling a large amount of data. Proposed method has been validated practically and implemented over Endace DAG 3.6ET, and has been rigorously analyzed. Our results are encouraging in terms of accuracy and response time.

The remainder of this paper is organized as follows. In the following section, we discuss the related work and previous research related to our work. In section 3, we give a brief overview of *CMS* Sketch and parametric M-CUSUM mechanisms that are related to our studies. Section 4 describes our proposed method for detecting change point in a reversible sketch. In section 5, we present the analysis results from the application of the proposed framework over real internet traces. Finally, section 6 presents concluding remarks and the future work.

## 2 Related work

The authors of [24] use a non parametric version of CUSUM, as sequential hypothesis testing algorithm [9] for anomaly detection with TCP SYN flooding. In [20], the authors evaluate and compare two anomaly detection algorithms (adaptive threshold and CUSUM) for detecting TCP SYN attacks. They conclude that CUSUM is more efficient than adaptive threshold, especially for the detection of low intensity attacks. Moreover, they compare their results obtained with a parametric version of CUSUM (by assuming a normal distribution for SYN inter-arrival packets), with the result of non-parametric version used in [24], and they conclude to better performance. For this reason, we will restrict our study in this paper to the parametric version of CUSUM. However, CUSUM algorithm is not able to pinpoint the malicious flow responsible of anomaly. It only raises an alarm after the detection of anomaly. In [10, 21, 22], the authors prove that multi-channel CUSUM is more efficient than single channel, which means applying CUSUM at flow level (given some criteria for packets aggregations and flows classifications), is more efficient in anomaly detection than applying this algorithm over the total number of ingoing packets.

The Multi-channel CUSUM [10, 21, 22] (M-CUSUM) is statistical self learning algorithm for initializing required parameters (e.g. mean and variance) in order to build an initial normal profile, in an adaptive manner with various network load and traffic patterns. M-CUSUM belongs to anomaly-based intrusion detection class, which detect a change in traffic parameters, through using the assumption that most anomalies induce a change in distributions of monitored parameters (mean, variance, etc.).

However, per flow application of CUSUM is prohibitive for real time operations over high speed networks, where storage and update operations of state-space flows information are very costly. In [1, 11, 12], a stretched data structure with a linear complexity of update/search operations, was proposed and used to handle large data. The authors in [3, 4, 13] have tackled the problem of offline anomaly detection over sketch by verifying if the values of the sketch buckets associated to a given key are heavy hitter or not. Recent work in [19] lookup for heavy buckets in the sketch resulted from the difference between current epoch and time series forecasting sketches. However, heavy hitter flows do not necessarily correspond to malicious flows. Therefore, we will address this problem by using M-CUSUM over sketch buckets in this paper.

Sketch is based on universal hash functions and random aggregations, and does not store information about active flows identifiers. In [12], it was used with the storage of all existing flows identifiers during a time interval, and through re-hashing of all stored identifiers to determine malicious flows. Unfortunately, storing all flows identifiers, especially when monitoring high speed links, is neither scalable nor efficient for online monitoring. There is a need for a software compliant reversal procedure over sketch to pinpoint corresponding keys to malicious flows.

## 3 Background

In this section, we briefly survey the underlying count-min sketch data structure and multi-channel CUSUM theory related to our work.

### 3.1 Count-Min Sketch

Let $S = s_1 s_2 \dots s_n$ be the set of input stream that arrives sequentially, item by item [5]. Each item $s_i = (\kappa_i, v_i)$ is identified by a key $\kappa_i \in U$ drawn from a fixed universe $U$ of items. A reward (or frequency occurrence) value $v_i \in \mathbb{R}$ is associated with each key. The arrival of item with key $\kappa_i$ increments its associated

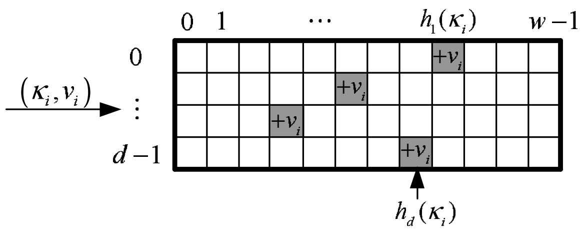


**Fig. 1.** Sketch data structure

counter in the $j^{th}$ hash table by $\nu_i$ $(C_{j,h_j(\kappa_i)} += \nu_i)$, as shown in Fig. 1. The update procedure is realized by $d$ different hash function, chosen from the set of 2-universal hash function $H_j(\kappa_i) = \{((a_j\kappa_i + b_j) \bmod P_U) \bmod w\}$, to uniformly distribute $\kappa_i$ over hash tables and to reduce collision. Parameter $P_U$ is a prime number larger than the maximum number in universe, and Mersenne prime numbers of the form $2^i - 1$ are generally chosen for fast implementation.

The Count-Min point query returns an estimate of the accumulated value for a given key, as the minimum of $d$ counter $(\hat{s}_k(\kappa_i) = \min_{0 \le j < d}\{C[j][h_j(\kappa_i)]\})$.

Ongoing IP packets into an ISP can be classified as series of $(\kappa_i, v_i)$, where $\kappa_i$ can be the destination IP address (DIP), or any other fields in packet header, and the value $v_i$ can be the number of $SYN$ request. *CMS* query can estimate if a given DIP (key) is under $SYN$ flooding attack by verifying the value of $\hat{s}_k(\kappa_i)$.

In *CMS* we use $d = \lceil \ln(1/\delta) \rceil$ pairwise independent universal hash functions, where each one receives $\kappa_i$ as parameter and return a random integer in the range $w = [0, e/\epsilon]$. $\varepsilon$ is the error rate with probability less than $\delta$. Thus, it maintains modest storage requirements of $O(\ln(1/delta) \times (1/\varepsilon))$ count cells.

## 3.2 Multi-channel Cumulative Sum Algorithm

In contrast to the most widely used techniques with sketch for anomalies detection (heavy hitter), sketch can be used with various sophisticated sequential detection procedures [10, 20, 24] to uncover anomalies. In this paper, we will focus on multi-channel CUSUM algorithm, due to its low computational overhead and modest storage requirements.

In this section, we briefly review the sequential parametric multi-channel CUSUM algorithm used to detect change point in traffic in [20]. CUSUM relies on two phases: training and detection. In training phase, it establishes and updates a dynamic behavior profiles for normal flows. In detection phase, it uses log likelihood ratio to detect any kind of abrupt deviation from well established profile. In this context, CUSUM define anomaly as any unusual event that does not fit the normal behavioral profile. Hence, it is indicative of unpredictable or unreliable events, that threat to the network. In multi-channel version of CUSUM, the algorithm is applied over many channels, and once an anomaly is detected in any channel, an alarm is raised.

Let $\{X_{ij}^{nT}, 1 \leq i \leq d, 1 \leq j \leq w\}$ be the value of each bucket during the $n^{th}$ time interval. Observations $X_{ij}^{nT}$ ar e *i.i.d* with a *pdf* $f_{ij,\gamma_0}(x)$ for $n < t_a$ (before attack occurrence) and with another *pdf* $f_{ij,\gamma_1}(x)$ for $n \geq t_a$ (after attack), where $t_a$ is the instant of attack detection. M-CUSUM test statistical hypotheses $H_{ij}$ (eq.( 1)) to detect abrupt change in bucket with index $(i, j)$ at the time epoch $n = t_a$:

$$H_{ij,0}: \quad \gamma_{ij} = \gamma_0 \qquad \text{versus} \qquad H_{ij,1}: \quad \gamma_{ij} = \gamma_1 \tag{1}$$

Where $\gamma_0$ and $\gamma_1$ are respectively the *pdf* parameters before and after change occurrence. The detection of anomaly is based on log likelihood ratio for an observation $X_{ij}^{nT}$ test between the two hypotheses:

$$s_{ij}^{nT} = \ln\left(\frac{pr(X_{ij}^{nT}|\gamma_1)}{pr(X_{ij}^{nT}|\gamma_0)}\right) \tag{2}$$

If we assume Gaussian distribution $f_{ij,\gamma_0}(X_{ij}^{nT}) = N(\mu_{ij,0}, \sigma_{ij,0}^2)$ for the hypothesis $H_{ij,0}$ and $f_{ij,\gamma_1}(X_{ij}^{nT}) = N(\mu_{ij,1}, \sigma_{ij,1}^2)$ for $H_{ij,1}$, the log likelihood ratio take the following form:

$$s_{ij}^{nT} = \ln\frac{\sigma_{ij,0}}{\sigma_{ij,1}} + \frac{(X_{ij}^{nT} - \mu_{ij,0})^2}{2\sigma_{ij,0}^2} - \frac{(X_{ij}^{nT} - \mu_{ij,1})^2}{2\sigma_{ij,1}^2} \tag{3}$$

The cumulative sum function is a summation of the log likelihood ratio:

$$S_{ij}^{nT} = \sum_{k=1}^{n} s_{ij}^{kT} \tag{4}$$

$S_{ij}^{nT}$ will increase when $s_{ij}^{nT} > 0$, and decreases for $s_{ij}^{nT} < 0$. When the value of $S_{ij}^{nT}$ become greater than threshold $h$, a decision can be taken about the hypotheses ($H_{ij,0}$ for normal condition and $H_{ij,1}$ for attack condition). Therefore, the relevant information for detecting change lies in the difference between the value of the log-likelihood ratio and its current minimum value [20]. Hence the stopping time for the M-CUSUM algorithm is given by:

$$t_a = t_a(h) = \min\{n \geq 1 : G_{ij}^{nT} \geq h\} \tag{5}$$

Where:

$$G_{ij}^{nT} = S_{ij}^{nT} - m_{ij}^{nT} \quad \text{and} \quad m_{ij}^{nT} = \min_{\substack{1 \leq i \leq d \\ 1 \leq j \leq w}} S_{ij}^{nT} \tag{6}$$

The statistic function $G_{ij}^{nT}$ obeys the recursion:

$$G_{ij}^{nT} = \left\{ G_{ij}^{(n-1)T} + \ln\left(\frac{pr(X_{ij}^{nT}|\gamma_1)}{pr(X_{ij}^{nT}|\gamma_0)}\right) \right\}^{+} \wedge G_{ij}^{0} = 0 \tag{7}$$

Where $\{y\}^+ = \max(0, y)$. We consider that $X_{ij}^{nT}$ follows a Gaussian distribution with known variance $\sigma_{ij}^2$ that remains unchanged after attack, and $\mu_0$ and $\mu_1$ are the mean before and after the attack, with a value of $\mu_1 = \alpha.\mu_0$, by following the same analysis and assumptions in [20], where authors have conclude to more false alarm and more miss detection when replacing the Gaussian distribution assumption in CUSUM by its non-parametric version. With the assumption of Gaussian distribution, eq. (7) takes the following form:

$$G_{ij}^{nT} = \left\{ G_{ij}^{(n-1)T} + \frac{\mu_{ij,1} - \mu_{ij,0}}{\sigma_{ij}^2} \left( X_{ij}^{nT} - \frac{\mu_{ij,1} + \mu_{ij,0}}{2} \right) \right\}^+ \wedge\ G_{ij}^0 = 0 \tag{8}$$

After the substitution of $\mu_{ij,1}$ by $\mu_{ij,1} = \alpha.\mu_{ij,0}$, eq. (8) becomes:

$$G_{ij}^{nT} = \left\{ G_{ij}^{(n-1)T} + \frac{(\alpha - 1)\mu_{ij,0}}{\sigma_{ij}^2} \left( X_{ij}^{nT} - \frac{(\alpha + 1)\mu_{ij,0}}{2} \right) \right\}^+ \wedge\ G_{ij}^0 = 0 \tag{9}$$

The detection of anomaly is given by testing the value of M-CUSUM function $G_{ij}^{nT}$ (if $G_{ij}^{nT} > h$ then alarm is raised). The values of $\mu_{ij,0}$ and $\sigma_{ij}^2$ can be estimated in a self learning phase and updated dynamically using *EWMA* (Exponential Weighted Moving Average) formulas given in eq. (10) and eq. (11):

$$\mu_{ij}^{nT} = \beta\mu_{ij}^{(n-1)T} + (1-\beta)X_{ij}^{nT} \tag{10}$$

$$\sigma_{ij}^2(nT) = \beta\sigma_{ij}^2\,((n-1)T) + (1-\beta)(X_{ij}^{nT} - \mu_{ij}^{nT})^2 \tag{11}$$

In fact, with the large fluctuations and variations in traffic characteristics (heavy tailed distributions for packets length, memory-less inter-arrival, self-similarity and long-range dependence, etc.), and with the lack of consensus about distributions of traffic characteristic parameters, one may wonder about the efficiency of Gaussian distribution assumption for CUSUM. The M-CUSUM [10] function in non parametric version is updated using the following formula:

$$G_{ij}^{nT} = \left\{ G_{ij}^{(n-1)T} + c_{ij}(X_{ij}^{nT} - \mu_{ij,0} - \varepsilon_{ij}\mu_{ij,1}) \right\}^+ \wedge\ G_{ij}^0 = 0 \tag{12}$$

However, the problem is in choosing the parameters ($c_{ij}$, $\varepsilon_{ij}$, and $\mu_{ij,1}$) that control the sensitivity of attack detection, which is not a straightforward task, and left to user. $c_{ij}$ a positive weight which is set to 1 in [24]. Parameter $\varepsilon_{ij}$ is a tuning parameter chosen from $[0, 1]$ due to the average delay detection [10] which must be a positive number:

$$ADD_{t_0}(t_a) = \frac{h}{(1-\varepsilon_{ij})\mu_{ij,1} - \mu_{ij,0}} > 0 \Rightarrow \varepsilon_{ij} < 1 - \mu_{ij,0}/\mu_{ij,1} \tag{13}$$

In fact, we can get the Gaussian parametric version of CUSUM (given in eq. (9)) from the non-parametric version by substituting parameters $\varepsilon_{ij}$ and $c_{ij}$ in eq. (12) by:

$$c_{ij} = \frac{2\varepsilon_{ij}\mu_{ij,1}}{\sigma_{ij}^2} \quad \text{and} \quad \varepsilon_{\mathrm{ij}} = \frac{\mu_{\mathrm{ij},1} - \mu_{\mathrm{ij},0}}{2\mu_{\mathrm{ij},1}} \tag{14}$$

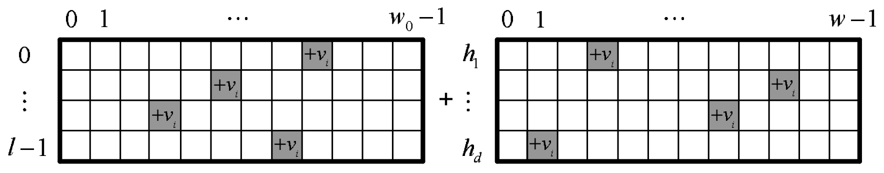


**Fig. 2.** *MLRS* and *CMS* sketch

Therefore, parametric version of CUSUM is not more than special case of non-parametric version, and choosing normal distribution becomes parameters adjustment choice for CUSUM detection algorithm.

A detection procedure should have a low false alarm rate $FAR$ and small delay for attack detection. In [22], it was proven that $CUSUM$ is asymptotically optimal. It minimizes the average delay detection $ADD_{t_0}(t_a)$ for a given false alarm rate $\overline{FAR}$, i.e $FAR(t_a) \leq \overline{FAR}$. The $FAR$ increases by decreasing the speed of detection, and a trade-off between low $FAR$ and minimum delay detection is required. The threshold value should be chosen from the condition $E_0 t_a(h) = 1/\overline{FAR}$ to minimize delay detection given a $\overline{FAR}$. It is worth noting that value of threshold $h$ controls the sensitivity of the attack detection, hence large value of $h$ decreases the $FAR$, but true attacks may also completely missed. We refer to [10, 20–22] for a complete reference about CUSUM parameters and implementation details.

## 4 Proposed approach

Our proposed framework is based on 2 data summary architecture: a Multi-Layer Reversible Sketch (*MLRS*) and a Count-Min Sketch (*CMS*) as shown in Fig. 2. Operations of the proposed framework are performed by two steps. Firstly, it continuously updates the two sketches (*MLRS* and *CMS*) counters from input data stream $(\kappa_i, v_i)$ for a fixed time interval $T$. Secondly, it applies M-CUSUM in the background at each bucket to detect anomalies. Afterward we identify and output keys that mapped to buckets with a raised alarm by CUSUM.

In this paper, we seek to detect victim servers of TCP SYN flooding, as it was widely shown in the literature that more than 90% of the DoS attacks use TCP [15], and TCP SYN flooding dominates in available attacking tools. Ongoing packets are classified by $DIP$ as key for flow identifier, and only packets with bit SYN set to 1 in TCP flag are considered. We associate with the key $\kappa_i$ the $DIP$, and with $v_i$ the value of bit SYN in packet header $((\kappa_i, v_i) = (DIP, SYN))$. We can also monitor another kind of flooding with TCP (ACK, RST, FIN, etc.) or with other protocol (flooding UDP, SMURF, etc.) by changing the associated reward $v_i$, but we will restrict our analysis to TCP SYN flooding in this paper.

However, M-CUSUM only raises alarm in buckets after the detection of abrupt change, and due to random aggregations and collisions occurrences, reversing sketch is a difficult operation to uncover responsible flow of anomaly.

There is only two existing approaches in the literature. The first [12] is based on intuitive idea by storing all keys in $T$ time interval, and achieve verification by hashing for the second time the set of stored keys at the end of each interval. This strategy is inefficient in term of storage space and update speed for the list of keys.

The second approach [6, 19] is based on modular hashing and mangling via Galois Field $GF(2^n)$ operators, which is complex and more efficient for hardware implementation, as it was done in [19].

Our idea to reverse sketch is based on exploiting index in an additional multi-layer reversible sketch (Fig. 2), where indexes are used to store keys. In fact, the *MLRS* is used in the same way of *CMS* sketch, where the arrival of each key increments its counter. However, each key has $l$ counter (one by layer), where we split the key of $N$ bit into $l \times w_0$ bit, with $w_0 = 2^P$, and $l = \lceil N/P \rceil$. $P$ is the number of bits used to split the key, and $w_0$ is used as layer width in *MLRS*. The update procedure is summarized in algorithm 1.

**Algorithm 1** Sketch Update procedure

```
1: $Mkey = crypt\_optimsed\_RC4(key)$;
2: for $i = 0$ to $d - 1$ do
3:     $j = universal\_hash_i(Mkey)$;
4:     $CMS[i][j].counter \mathrel{+}= v_i$;
5: end for
6: for $j = 0$ to $l - 1$ do
7:     $MLRS[j][Mkey\&(2^P - 1)].counter \mathrel{+}= v_i$;
8:     $Mkey >>= P$;
9: end for
```

If we seek to search for victim $DIP$ (or keys that map to buckets with raised alarm by M-CUSUM), we can release hierarchical search procedure in *MLRS*. If we don't find at least one bucket with raised alarm in each of the $i^{th}$ $(i \leq l - 1)$ first layers of *MLRS*, there is no need to continue searching in other deep layers or through the second *CMS* sketch. Malicious flows must have one alarmed bucket in each layer.

We will begin by the simple case, where we assume that there is at most one bucket with CUSUM raised alarm in each layer as shown in Fig. 2. To recover key, we concatenate the $l$ index in *MLRS* and we get the value of suspect key (e.g. $DIP$). We can not be sure of suspect before verification, where due to collision with other IP prefix, their value becomes large. The suspect key is verified through hashing and verification (by count-min query of CUSUM function) in the *CMS* for confirmation.

In general, even with a different value of width (e.g. $2^{12}$ or $2^{14}$) for the *MLRS*, many buckets in different layers will be subject to collision occurrence, and in some case, we will be found with a bigger set of keys to verify through *CMS* than the original one. Nevertheless, it is important to notify that even if the set of suspect key is larger than departure one, it requires a small memory and fast update time with respect to original list.

To resolve this problem and reduce collision in *MLRS*, we use the idea of IP-mangling technique presented in [19], but with an optimized version of RC4 (Ron's Code [7]) ciphering algorithm rather than Galois Field $GF(2^n)$. The optimized RC4 code is available from [8].

IP mangling is a reversible procedure, which randomizes the input data in an attempt to destroy correlation between keys, to disperse adjacent keys uniformly at all available buckets. Mangled key is denoted by Mkey in this paper. This technique is a bijective function that maps keys in a universe $U$ to $U$. Each key $\kappa_i$ is mapped to $y_i = f(\kappa_i)$, with the function $f$ chosen in a way to destroy any correlation between keys, as show in table 1. Any bijective function able to destroy correlation between keys, and return a completely random set of keys, can be used. Afterward, we use $f^{-1}(y_i)$ to recover suspect key $\kappa_i$ from *MLRS*.

In [19], the function $f(\kappa_i) = a \otimes \kappa_i \oplus b$ is used for mangling, where $\otimes$ is the multiplication operation defined on $GF(2^n)$, $\oplus$ is the bit-wise XOR operator, and $a$ and $b$ are two random number uniformly chosen from the universe $U$. The reverse of a mangled key is obtained from $f^{-1}(y_i) = a^{-1} \otimes (y_i \oplus b)$ by precomputing the value of $a^{-1}$. In the other hand, the authors of [19] explain clearly that the direct calculation of $a \otimes x$ is very expensive, as it requires multiplying two polynomials (of degree $n-1$) modulo an irreducible polynomial (of degree $n$). Therefore, they use tabulation and many additional precomputing tables to reduce complexity. In fact, Galois field is based on bit by bit operations which is hardware compliant, and requires additional memory for fast calculation of polynomial product. In contrast, the optimized RC4 bloc cipher algorithm is ideal for software implementation, as it requires only byte manipulations and its implementation is based on few lines of code. It has been proven to be powerful in our experimentations for mangling and destroying any correlation between adjacent keys, in terms of random Hamming distance between adjacent keys, as shown binary values in table 1.

**Table 1.** Mangling *DIP* by optimized RC4

| *DIP* | Mangled key |
|---|---|
| 192.168.92.40 | 10010100101001011110100010011011 |
| 192.168.92.41 | 10101011011001000011001000100110 |
| 192.168.92.42 | 10010110111011000010010010101110 |
| 192.168.92.43 | 00100000001101001000000001101101 |
| . . . | . . . |

In Fig. 3, we show the distribution of collision when using direct mapping and mangling to update multi-layer sketch with $P = 8$ for a universe size of 32-bit ($\kappa_i = DIP$). Data traces from 1 minute real bidirectional Internet traffic of 1776 flows (here flows are classified by $DIP$). In fact, used mangling technique allows

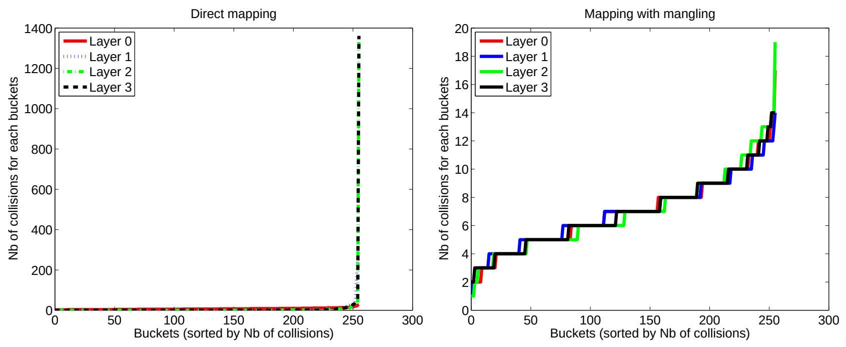


**Fig. 3.** Distribution of collisions for each bucket

to uniformly distributing key over buckets, and prevent collision over IP with same prefix. It is worth noting, that we proved by experiments that the number of collision reduce significantly by increasing the width of multi-layer sketch.

At the end of each time interval $T$, and after updating counters of *MLRS* and *CMS* continuously in online manner, M-CUSUM anomaly detection algorithm run in the background, to update CUSUM function in each bucket, and to raise alarm in bucket where the value of function $G_{ij}^{nT}$ exceeds the threshold. Afterward, we scan *MLRS* for identification of all possible sequence of $l$ bucket (one per layer) with triggered alarm and we realize verification through count-min query over the *CMS*, to ensure that the corresponding buckets with the $d$ universal hash functions have a triggered alarm by CUSUM. However, we don't store the set of suspect keys, but once we have a suspect, we realize verification through the *CMS* before integrating it in alert message.

The hierarchical search procedure for alarmed bucket in *MLRS*, and the verification through *CMS* sketch are given in algorithm 2, for a universe of size $2^n$, and a width of $2^P$ for *MLRS*, $P = n/3$ and $l = 3$. Boolean alarm variable is used to indicate the state of CUSUM function.

## 5 Experiments results

In this section, we present performance analysis results of juxtaposing M-CUSUM detection algorithm over reversible sketch, for detecting victims of TCP SYN flooding attacks. We have implemented M-CUSUM over sketch in C using the code of CMS available from [14]. We applied the proposed algorithm over many public traces (LBL-TCP-3, Abilene, Auckland, etc.) available from [16], and other traces used in OSCAR RNRT French Research project (OTIP, ADSL). Online implementation over Endace DAG 3.6ET is realized, and many experiments have been conducted for accuracy analysis. Our results are encouraging in terms of accuracy and response time.

**Algorithm 2** Hierarchical search and verification

```
1: for $i = 0$ to $2^P - 1$ do
2:   if $(MLRS[0][i].Alarm)$ then
3:     for $j = 0$ to $2^P - 1$ do
4:       if $(MLRS[1][j].Alarm)$ then
5:         for $k = 0$ to $2^P - 1$ do
6:           if $(MLRS[2][K].Alarm)$ then
7:             $Mkey = (k << 2 * P) \mid (j << P) \mid i;$
8:             $Alarm = cms\_query(CMS, Mkey);$
9:             if $(Alarm)$ then
10:              $DIP = decrypt\_ORC4(Mkey);$
11:              $output(DIP)$
12:            end if
13:          end if
14:        end for
15:      end if
16:    end for
17:  end if
18: end for
```

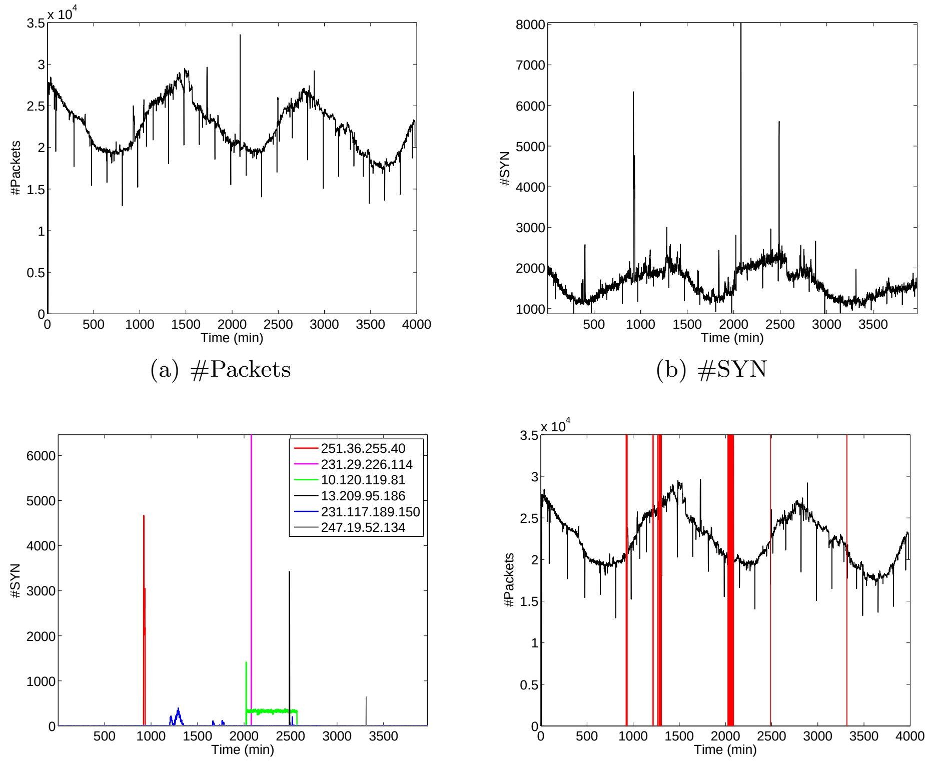
In this paper, we present the result of our experiments over the set of traces used in OSCAR project, and extensively studied in this project. We are interested in detecting victim of DoS/DDoS SYN Flooding in these traces. Afterward, we conduct performance analysis to study the influence of parameters at true positive and false detection.

The parameters we considered for the M-CUSUM algorithm were: threshold $h = 5$, $\alpha = 1.5$, $\beta = 0.9$ as in [20]. For sketch parameters: $P = 8$ unless otherwise noted, $w_0 = 256$, $l = 4$, $d = 4$ hashing functions from the set of 2-universal hash function, and with the use of tabulation [23].

First, we present our analysis result over anonymized OTIP traces: 3 days of bidirectional traces collected by France Telecom ISP with Netflow format ($\sim 6.9GB$) and contains $\sim 896.10^5$ flows.

Fig. 4(a) and Fig. 4(b) show the variation of the number of packets during a time interval $T = 1min$, as well as the variation of number of SYN during the 3 days. We have applied our proposed framework over traces to uncover attacks, and we isolate the number of connection request received by each of identified victim as shown in Fig. 4(c). The separation of received SYN by each destination is realized for additional information about the false alarm rate, and have been used for manual verification. The raised alarms by M-CUSUM for detected victims and their IP addresses are presented in Fig. 4(d). It is worth noting that response time for analyzing the whole 3 days OTIP trace is less than 2 minutes over a Pentium 1.72 $Ghz$ with 1 $GB$ of RAM memory.

Our second experiment considers two unidirectional anonymized traces ADSL (up and down) during during 3 hours of capture in pcap format (contains $\sim 825.10^5$ packets). We realize the same analysis study and manual verification as



(a) #Packets (b) #SYN

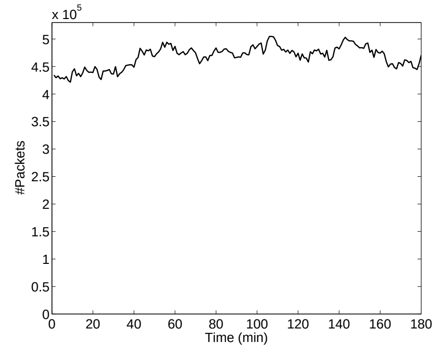
(c) Detected malicious flows (d) M-CUSUM raised alarms

**Fig. 4.** Analysis results for OTIP trace

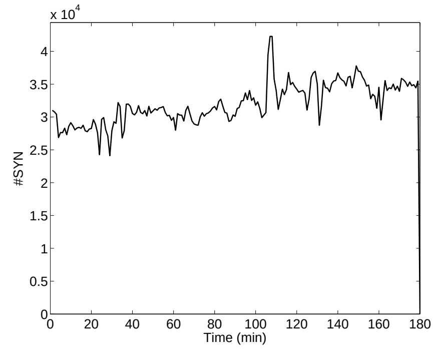
in the first experiment. First, we show the result obtained over upload traffic, and afterward we present the analysis result for the down traffic.

Fig. 5(a) and Fig. 5(b) show the variation of the total number of packets and the number of SYN during a time interval of $1min$. Fig. 5(c) and Fig. 5(d) show the number of SYN received by the only existing victim in this trace (with $\sim 841.10^5$ packets), and the raised alarms by *M-CUSUM*. Similarly, Fig. 6(a) and Fig. 6(b) show the variation of the total number of packets and the number of SYN during a time interval of 1 min. Fig. 6(c) and Fig. 6(d) show the number of SYN of two detected victims and the raised alarms by CUSUM. However, deep manual investigations show that the $DIP$ address (97.68.23.88) is not victim of SYN flooding, but of PortScan attack.

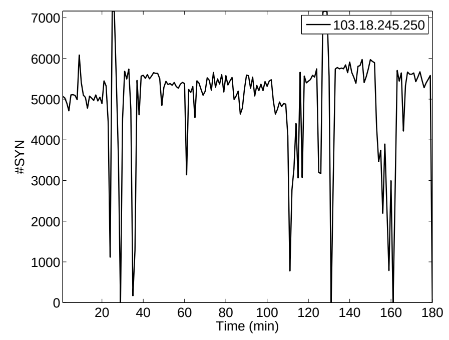
In the third set of our experiments, we conduct performance analysis study via Receiver Operational Characteristics (ROC) curve, to study the accuracy of the proposed framework. Our analysis verify the false positive and true positive probability, with the variation of the value of threshold $h$ and the $MLRS$ sketch width. However, due to the lack of public well documented traces with well known



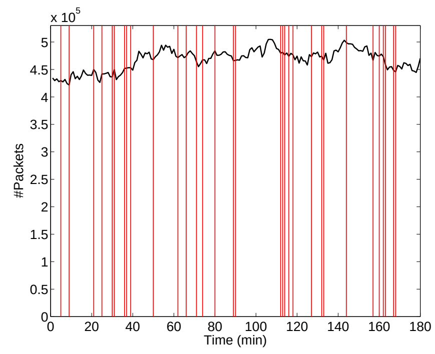
(a) #Packets



(b) #SYN



(c) Detected malicious flows

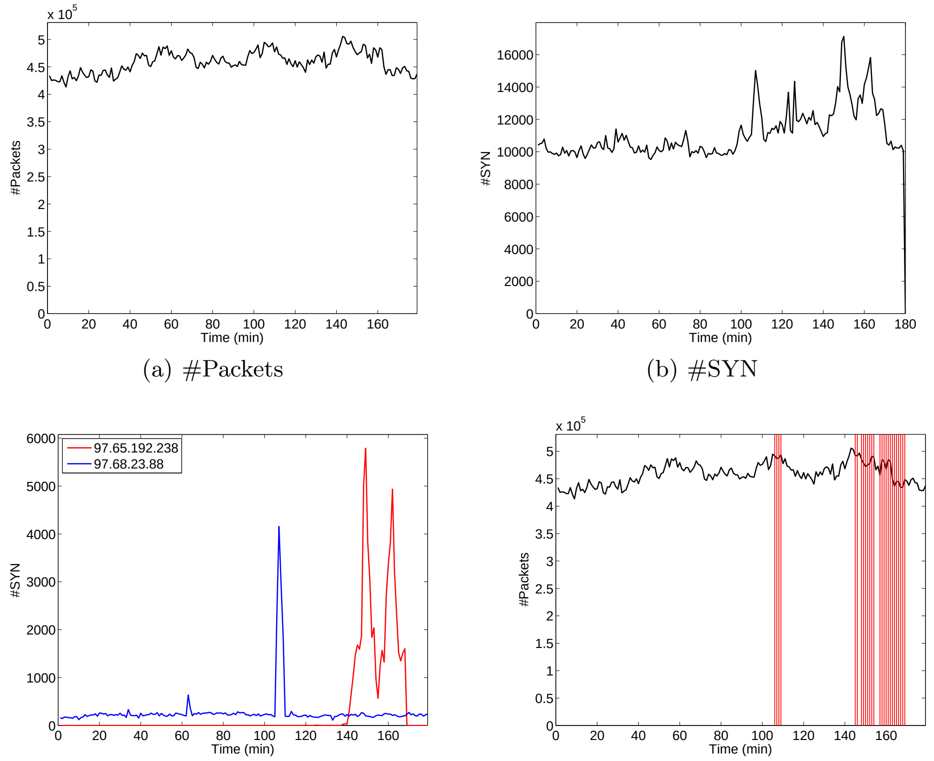


(d) M-CUSUM raised alarms

**Fig. 5.** Analysis results for ADSL up trace.

attacks, we use the overall attacks uncovered by other research laboratory that have analyzed OTIP traces, as a complete set of existing one. Therefore, true positive and false positive probabilities are easily verified because we know in advance the IP address of victim servers, and the number of existing attacks. $P_{TD}$ is the number of detected attacks divided by the total number of existing ones. $P_{FP}$ is the percentage of raised alarm that did not correspond to real attack.

Fig. 7 illustrates the relation between true positive and false positive, as well as $P_{TD} = f(h)$ and $P_{FP} = f(h)$, where $FP$ decreases as the threshold value increases, and true attacks may also completely missed. Hence, a tradeoff between false alarm and true positive detection is required to control sensitivity and prevent miss detection. We also notice that large sketch width decreases the false positive and increase the detection rate.



(a) #Packets (b) #SYN

(c) Detected malicious flows (d) M-CUSUM raised alarms

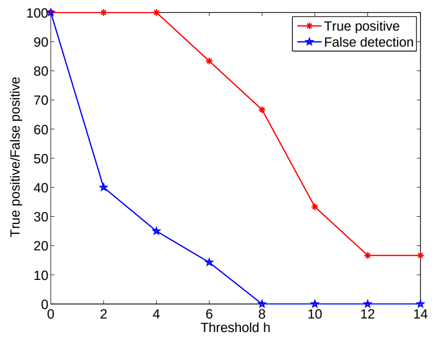
**Fig. 6.** Analysis results for ADSL down trace

## 6 Conclusion

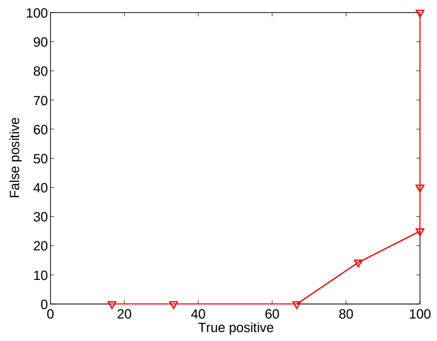
In this paper, we propose a new framework that integrates sketch and CUSUM for online anomalies detection at high speed link. Proposed framework is able to automatically pinpoint the malicious IP flows responsible of anomaly, through exploiting bucket index in an additional multi-layer sketch.

We proved the effectiveness of the proposed approach through implementation and testing at real traces with DoS/DDoS via SYN flooding. Results of our experimentations have proved the capacity of early detection even for low intensity of DoS/DDoS attacks.

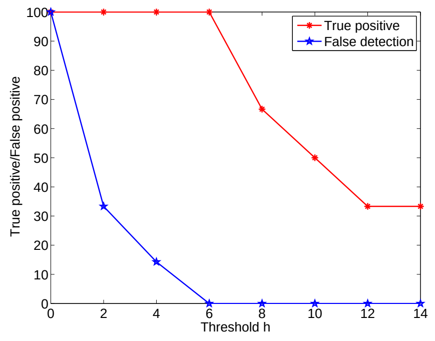
The proposed method is easily decentralized due to linear property of sketch with respect to addition operator. Ongoing work will be converged toward the hierarchical distribution of the proposed approach, and the reduction of the size of exchanged sketch information between different monitoring nodes in different layers.



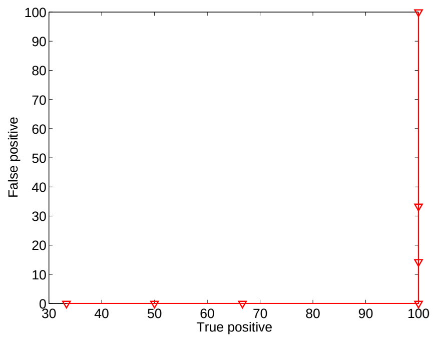
(a) $P_{TD} = f(h) \wedge P_{FA} = f(h)$ for $P = 8$



(b) $P_{FP} = f(P_{TP})$ for $P = 8$



(c) $P_{TD} = f(h) \wedge P_{FA} = f(h)$ for $P = 14$



(d) $P_{FP} = f(P_{TP})$ for $P = 14$

**Fig. 7.** $P_{TD}$ and $P_{FP}$ for $P = 8$ and $P = 14$.

## Acknowledgments

This work has been partially funded by the French National Research Agency through the OSCAR project.

## References

1. Charikar, M., Chen, K., Farach-Colton, M.: 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) 693–703
2. Cisco Systems Inc: Cisco netflow. http://www.cisco.com/wrap/public/732/Tech/netflow
3. Cormode, G., Korn, F., Muthukrishnan, S., Srivastava, D.: Diamond in the rough: Finding hierarchical heavy hitters in multi-dimensional data. In: Proceedings of the 23rd ACM SIGMOD. (2004) 155–166

4. Cormode, G., Muthukrishnan, S.: What's new: Finding significant differences in network data streams. In: Proceedings of IEEE Infocom. (2004) 1534–1545
5. Cormode, G., Muthukrishnan, S.: An improved data stream summary: The count-min sketch and its applications. Journal of Algorithms **55**(1) (2005) 58–75
6. Feng, W., Zhang, Z., Jia, Z., Fu, Z.: Reversible sketch based on the xor-based hashing. In: Proceedings of the Asia-Pacific Conference on Services Computing (APSCC '06), Guangzhou, Guangdong, China (2006) 93–98
7. Fluhrer, S., McGrew, D.: Statistical analysis of the alleged rc4 keystream generator. In: Proceedings of the 7th International Workshop on Fast Software Encryption (FSE '00), London, UK, Springer-Verlag (2001) 19–30
8. Gutmann, P.: Optimized rc4 code. http://www.zengl.net/freeswan/
9. Jung, J., Paxson, V., Berger, A.W., Balakrishnan, H.: Fast Portscan Detection Using Sequential Hypothesis Testing. In: IEEE Symposium on Security and Privacy 2004, Oakland, CA (2004)
10. Kim, H., Rozovskii, B., Tartakovsky, A.: A nonparametric multichart cusum test for rapid intrusion detection. International Journal of Computing and Information Science **2**(3) (2004) 149–158
11. Krishnamurthy, B., Sen, S., Zhang, Y., Chen, Y.: 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) 234–247
12. Li, X., Bian, F., Crovella, M., Diot, C., Govindan, R., Iannaccone, G., Lakhina, A.: Detection and identification of network anomalies using sketch subspaces. In: Proceedings of the 6th ACM SIGCOMM on Internet measurement (IMC '06), New York, NY, USA, ACM Press (2006) 147–152
13. Li, Y., Yang, J., An, C., Zhang, H.: Finding hierarchical heavy hitters in network measurement system. In: Proceedings of the 2007 ACM symposium on Applied computing (SAC '07), New York, NY, USA, ACM Press (2007) 232–236
14. Massive Data Analysis Lab: MassDal: Count-min sketch source code. http://www.cs.rutgers.edu/7Emuthu/massdal-code-index.html
15. Moore, D., Voelker, G., , Savage, S.: Inferring internet denial-of-service activity. In: proceedings of the 2001 USENIX Security Symposium. (2001) 9–22
16. National Laboratory of Applied Network Research: NLANR: Traces archive. http://pma.nlanr.net/Special/
17. Paxson, V.: BRO: A system for detecting network intruders in real-time. Computer Networks **31**(23–24) (1999) 2435–2463
18. Roesch, M.: Snort - lightweight intrusion detection for networks. In: LISA '99: Proceedings of the 13th USENIX conference on System administration, Berkeley, CA, USA, USENIX Association (1999) 229–238
19. Schweller, R., Li, Z., Chen, Y., Gao, Y., Gupta, A., Parsons, E., Zhang, Y., Dinda, P., Kao, M.Y., Memik, G.: Reverse hashing for high-speed network monitoring: Algorithms, evaluation, and applications. In: Proceedings of IEEE International Conference on Computer Communications (INFOCOM 06). (2006) 1–12
20. Siris, V.A., Papagalou, F.: Application of anomaly detection algorithms for detecting syn flooding attacks. In: Proceedings of IEEE Global Telecommunications Conference (GLOBECOM '04). Volume 4., Dallas, USA (2004) 2050–2054
21. Tartakovsky, A.: Asymptotic performance of a multichart cusum test under false alarm probability constraint. In: Proceedings of the 44th IEEE Conference on Decision and Control and the European Control Conference, Seville, Spain (2005) 320–325

22. Tartakovsky, A., Rozovskii, B., Blazek, R., Kim, H.: A novel approach to detection of intrusions in computer networks via adaptive sequential and batch-sequential change-point detection methods. IEEE Transactions on Signal Processing **54**(9) (2006) 3372–3382
23. Thorup, M., Zhang, Y.: Tabulation based 4-universal hashing with applications to second moment estimation. In: Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA '04), New Orleans, Louisiana, USA (2004)
24. Wang, H., Zhang, D., Shin, K.G.: Syn-dog: Sniffing syn flooding sources. In: Proceedings of the 22 nd International Conference on Distributed Computing Systems (ICDCS'02), Washington, DC, USA, IEEE Computer Society (2002) 421–429