Telecommun Syst (2006) 33:47–65
DOI 10.1007/s11235-006-9006-0

Throttling spoofed SYN flooding traffic at the source

Wei Chen · Dit-Yan Yeung

Published online: 2 November 2006

Abstract TCP-based flooding attacks are a common form of Distributed Denial-of-Service (DDoS) attacks which abuse network resources and can bring about serious threats to the Internet. Incorporating IP spoofing makes it even more difficult to defend against such attacks. Among different IP spoofing techniques, which include random spoofing, subnet spoofing and fixed spoofing, subnet spoofing is the most difficult type to fight against. In this paper, we propose a simple and efficient method to detect and defend against TCP SYN flooding attacks under different IP spoofing types, including subnet spoofing. The method makes use of a storage-efficient data structure and a change-point detection method to distinguish complete three-way TCP handshakes from incomplete ones. This lightweight approach makes it relatively easy to deploy the scheme as its resource requirement is reasonably low. Simulation experiments consistently show that our method is both efficient and effective in defending against TCP-based flooding attacks under different IP spoofing types. Specifically, our method outperforms others in achieving a higher detection rate yet with lower storage and computation costs.

Keywords DDoS · SYN flooding · IP spoofing · Hash-based detection · Bloom filter · CUSUM

The research presented in this paper has been supported by a research grant from the Research Grants Council of the Hong Kong Special Administrative Region, China under the Area of Excellence (AoE) Scheme (Project No. AoE/E-01/99).

W. Chen (✉) · D.-Y. Yeung
Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong, China
e-mail: wchen@cse.ust.hk

D.-Y. Yeung
e-mail: dyyeung@cse.ust.hk

W. Chen
College of Computer, Nanjing University of Posts and Telecommunications, Nanjing 210003, Jiangsu, China

1 Introduction

Distributed Denial-of-Service (DDoS) attacks are large-scale cooperative attacks typically launched from a large number of compromised hosts. DDoS attacks are bringing about growing threats to businesses worldwide. Designed to elude detection by the most popular tools available today, these attacks can quickly incapacitate a targeted business, costing the victims a great deal in lost revenue and productivity. While many methods have been proposed to counter such attacks, they are either not efficient or not effective enough. Recent DDoS incidents show that such attacks continue to cause serious threats to the Internet.

DDoS attacks are even more difficult to fight against if IP spoofing is incorporated into such attacks. IP spoofing, or source IP address spoofing, refers to the technique of lying about the return address (i.e., source address) of a packet. With IP spoofing, attackers can gain unauthorized access to a computer or a network by making it appear that a message has come from a certain trusted machine by "spoofing" the IP address of that machine. This technique has been used by attackers for years, and is commonly used in DDoS attacks launched against commercial servers. Strictly speaking, IP spoofing is not an attack by itself; it is merely a scheme used with DDoS attacks. Since the attackers are mainly concerned with consuming network bandwidth and resources, they usually do not care about properly completing handshakes and transactions. Rather, they simply want to flood the victim with as many packets as possible within a short period of time. In order to prolong the effects of an attack, they spoof the source IP addresses to make tracing and stopping the DDoS as difficult as possible.

Spoofing techniques can be categorized into different types according to what spoofed source addresses are used in the attacking packets. The three common IP spoofing types are random spoofing, subnet spoofing, and fixed spoofing [11]. In random spoofing, the attacker randomly generates 32-bit numbers for use as source addresses of the attacking packets. In subnet spoofing, the addresses are generated from the address space corresponding to the subnet in which the agent machine resides. For example, a machine which is part of the 143.89.124.0/24 network may spoof any address in the range from 143.89.124.1 to 143.89.124.254. Compared to random spoofing, subnet spoofing has a much narrower range of IP addresses. Another type of IP spoofing, called fixed spoofing, chooses source addresses from a given list. In this case, the attacker typically wants to perform a reflector attack or impose a blame for attack on several specific machines.

Most existing defense schemes against IP spoofing focus on random spoofing. One example is to limit the chance of random IP spoofing by filtering at the routers. Implementing ingress and egress filtering at the border routers is one specific realization of this approach. At the upstream interface, the ingress filter should only allow source addresses within a valid range (usually corresponding to the same subnet), thus preventing spoofed traffic from being sent to the Internet. Unfortunately, this method does not work for subnet spoofing. Implementing encryption and authentication can also reduce the spoofing threats. In fact, both measures are already included in IPv6 to eliminate the spoofing threats. However, the current Internet still uses IPv4 as the dominant communication protocol.

To defend against spoofed flooding traffic, especially that with subnet spoofing, we propose a scheme that is based on a storage-efficient data structure and a change-point detection method. The storage-efficient data structure, which is a variant of Bloom filter [2], is used to generate a hash digest of the traffic. The change-point detection method is based on the CUSUM algorithm [3], which is a nonparametric change-point detection method. CUSUM enjoys the virtues of sequential and nonparametric test and its computational requirement is quite low. After some information about the traffic is extracted and stored in the Bloom filter, CUSUM is then applied to detect abnormal changes in the digested traffic.

The major contributions of this paper are summarized below:

- A modified Bloom filter is employed for the detection of spoofed flooding traffic. Since only fixed-size memory is required for the modified Bloom filter, it can avoid the potential threat of most existing DDoS defense schemes that are based on dynamic memory allocation. Moreover, the data structure is storage efficient, making it easier for Internet service providers (ISP) to adopt and deploy this source-side defense scheme.
- We address the issue of how to define the hash functions for use in the Bloom filter. Analysis shows that our proposed scheme has very low collision rate (and hence low false positive rate for our DDoS defense scheme).
- A simple and efficient change-point detection method can accurately detect spoofed flooding traffic, including that with the subtle IP spoofing type based on subnet spoofing. After detection, the traffic can be classified into three categories: random spoofing, subnet spoofing and fixed spoofing.

The remainder of this paper is organized as follows. We first review some basics about TCP handshakes in Section 2. In Section 3, a space-efficient data structure called Bloom filter is reviewed and the design of the hash functions is discussed. Our method makes use of a modified Bloom filter to store a hash digest of the relevant portions of a packet. The CUSUM based change-point detection scheme is presented in Section 4. We have performed some simulation experiments. The results reported in Section 5 show that our method can accurately detect spoofed flooding traffic under different IP spoofing types. We compare our method with two other DDoS detection methods, IDR and TOPS, which are also based on some hashing scheme. Finally, Section 6 presents some related work in the area of DDoS research and Section 7 concludes the work and outlines some future research issues.

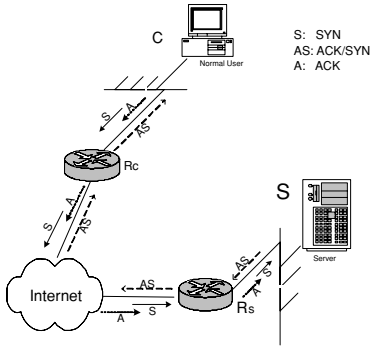
2 TCP handshakes

Most existing DDoS attacks exploit the Transmission Control Protocol (TCP) [13]. It has been reported that more than 90% of the existing DDoS attacks are TCP based [17], although we do expect that attacks of other types, such as UDP based, will increase in the future as more applications based on them are available in the Internet. It is well known that IP spoofing is one of the security problems in the TCP/IP protocol suite, and hence it is commonly exploited by attackers. In what follows, we first look at TCP handshakes for normal transactions and then those for spoofed ones.

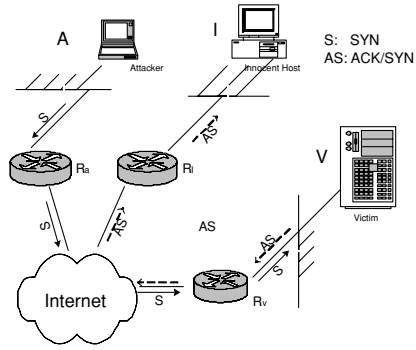
The normal three-way handshake sequence is depicted in Fig. 1(a). The client C first sends a *SYN* request to the server S. After receiving the request, server S replies with a packet, which contains both the acknowledgement *ACK* and the synchronization request *SYN* (denoted as *ACK/SYN* hereinafter). Then client C sends an *ACK* back to S to complete the establishment of the connection.

As mentioned above, there are three common types of IP spoofing techniques for faking the source IP addresses: random, subnet, and fixed. In random spoofing, the attacker randomly generates 32-bit numbers for use as source addresses of the attacking packets. Since the addresses are randomly generated, these spoofed IPs belong to different subnets and most of them are not from the subnet where the attack source is located. On the other hand, the addresses generated for subnet spoofing must be from the same subnet as the attack source. For fixed spoofing, source addresses are chosen from a given list of IPs.

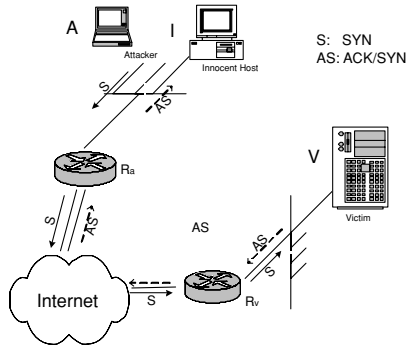
Under IP spoofing, the three-way handshake will be very different from that of the normal case as shown in Fig. 1(a). Attackers usually use unreachable spoofed source IPs in the attacking



(a) Normal three-way handshake



(b) Abnormal handshake caused by SYN flooding attack under random spoofing



(c) Abnormal handshake caused by SYN flooding attack under subnet spoofing

Fig. 1 Three-way handshakes for a complete TCP connection and half-open connections caused by SYN flooding attacks

packets to improve the attack efficiency [14]. These packets will not trigger the third round of a handshake. Figure 1(b) shows the scenario under random IP spoofing. Most connections will not receive the second round of each handshake because the *ACK/SYN* packets are sent to other subnets. Under subnet spoofing as shown in Fig. 1(c), however, *ACK/SYN* packets are sent to the correct subnet but destined to an incorrect host. The third round of a handshake is not successful. Thus, a major difference between random spoofing and subnet spoofing is the different return paths of the *ACK/SYN* packets. The figure for fixed spoofing is ignored since the effect is similar to that of subnet spoofing.

Another difference between random spoofing and subnet spoofing is that random spoofing has a much wider range of IPs than subnet spoofing. Random spoofing generates 32-bit IP addresses randomly, and hence the probability of generating the same IP address twice is very small. On the other hand, since the IP addresses in subnet spoofing are restricted to the range of a subnet, it is not unusual to find that quite a large number of attacking packets have the same spoofed source IP address. As a result, although both random spoofing and subnet spoofing give rise to many incomplete handshakes, the incomplete handshakes caused by subnet spoofing have more repeated source IPs.

Our method tries to detect the incomplete handshakes by monitoring the first and third rounds of each handshake. If any one of the rounds is lost, it is regarded as an incomplete handshake. We use the first and third rounds because both of them belong to the outgoing traffic. A study on the stability of end-to-end routing [12] shows that the paths from a source to a destination in the Internet are strongly dominated by a single route, with about two-thirds of the Internet paths having routes that persist for days or even weeks. Since the first and third rounds of a handshake are expected to occur within a very short period of time, it is reasonable to assume that they go through the same routing path and hence should be observed at the same router. This makes it easier to perform traffic monitoring. We do not consider the second round of a handshake because it may go through a path that is different from that of the first or third round.

Since both the first and third rounds of a handshake belong to the outgoing traffic, our method which requires only one-way traffic monitoring has the advantage of being flexible and hence can easily be deployed at the source side, the intermediate network or the victim side. Nevertheless, source-side deployment is preferred because malicious traffic can be mitigated before they enter the Internet. It is stated in [11] that existing methods can only detect subnet spoofing within an exit router and hence they cannot detect subnet spoofing anywhere between the exit router and the victim. However, our method can perform detection successfully with little restriction on the deployment location.

Recording the handshakes for all connections has very high computational and storage requirements especially in high-speed networks. To keep the overhead reasonably low, we propose a storage-efficient data structure in the next section.

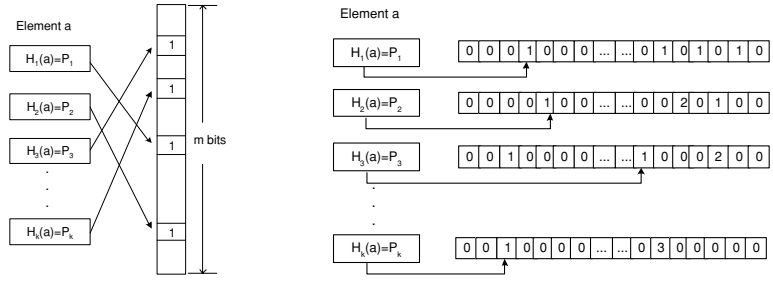
3 Traffic information digest

Accurate detection of anomalies has to depend on detailed information analysis. However, storing detailed traffic information is an expensive task. In order to extract useful information about abnormal (incomplete) handshakes, we only record information about TCP handshakes. Moreover, we use a storage-efficient data structure for this purpose. In this section, we first give a brief overview of such a data structure. We then discuss the design of hash functions that can guarantee sufficiently low false positive rates.

3.1 Bloom filter

Bloom filter was first proposed by Bloom [2] in 1970. It was originally used to reduce the disk access time to different files and other applications, such as spell checkers. Recently, it has been adapted for use by some methods for defending against DDoS attacks [1, 4, 15]. A Bloom filter is composed of a vector v of m bits, initially all set to 0. We have k independent hash functions, $h_1, h_2, \ldots, h_k$, each with a range $0, 1, \ldots, m-1$. The vector v can show the existence of an element from some address space A. Given an element $a \in A$, the bits at positions $h_1(a), h_2(a), \ldots, h_k(a)$ in v are set to 1 (Fig. 2(a)). Note that a particular bit may be set to 1 multiple times and hence may potentially lead to inaccurate results. Given a query of the existence of b in A, we check the bits at positions $h_1(b), h_2(b), \ldots, h_k(b)$. If any one of them is 0, then certainly b is not in A. Otherwise, we conjecture that b is in it. There is, however, a certain probability that the Bloom filter will give a false result. This probability is referred to as the false positive rate. The parameters k and m should be chosen carefully so that the false positive rate can be kept low enough.

A variant of the original Bloom filter, called *counting Bloom filter*, uses a table of counters to replace the n bits, as shown in Fig. 2(b). The table is composed of k rows with n counters in



(a) Original Bloom filter (b) Counting Bloom filter

Fig. 2 A Bloom filter uses independent hash functions to map an input into multiple locations

each row. The rows are independent of each other and each row corresponds to one hash function h_i, $1 \leq i \leq k$. The n counters in each row correspond to addresses from 0 to $n-1$. All counters are initialized to 0. Each counter represents how many times the corresponding location has been hit. When a key a (such as an IP address) is inserted or deleted, the value of the corresponding counter in each row is increased or decreased by 1, according to $h_i(a)$ for all k rows. If an IP address b is already stored in the modified bloom filter, the counters at locations $h_i(b)$, $1 \leq i \leq k$, in the table should all be nonzero.

3.2 Traffic digest

Two hash tables, which are based on the counting Bloom filter, are constructed to record information about TCP handshakes. One table, called *destination table* and denoted as T_d, is used to record destination IP information. The other one, called *source table* and denoted as T_s, is used to record source IP information.

When a *SYN* request packet, corresponding to the first round of a handshake, is captured in the outgoing traffic, the destination IP of the *SYN* packet is hashed using the k independent hash functions and the corresponding counters in T_d hit by the k hash functions are incremented by α where $0 < \alpha \leq 1$. Meanwhile, another hash table T_s works in the same way but records the source IP information. The source IP of the *SYN* packet is hashed into the source table T_s and the corresponding counters are updated.

When an *ACK* packet, corresponding to the third round of a handshake, is captured in the outgoing traffic, both the destination and source IPs are extracted again and hashed into T_d and T_s, respectively. This time the corresponding counters are decremented by α where $0 < \alpha \leq 1$. For a normal TCP handshake, both *SYN* and *ACK* are observed and hence the corresponding counters are first incremented and then decremented by α, leading to no resulting changes. Figure 3 summarizes how tables T_d and T_s are used.

All entries in both tables T_d and T_s are reset periodically to prevent them from growing indefinitely when it is under attack. Suppose R_{t-1} is the value of a counter at time $t-1$. Its value is reset to R_t at time t as follows:

$$R_t = (1-\alpha)R_{t-1}, \qquad 0 < \alpha \leq 1.$$

Although it is possible that two different IP addresses are mapped to the same counter in a row, the probability that they get mapped to the same counters in all k ($k \geq 1$) rows is very low even for a small value of k. As a result, the false positive rate caused by hash collision is rather

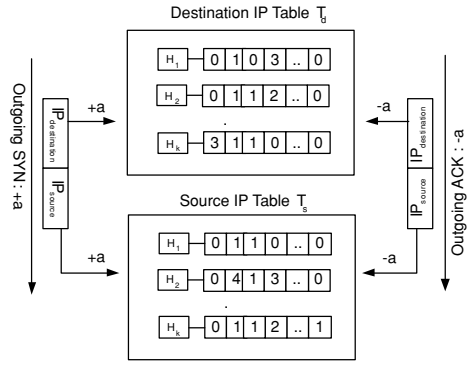


Fig. 3 T_d and T_s together monitor TCP *SYN* and *ACK* packets

low. Assume that there are l_i counters with suspicious values in row i. The probability that a legitimate packet hits the suspicious counters in all k rows is equal to

$$P = \prod_{i=1}^{k} \left(\frac{l_i}{n} \right) = \frac{\prod_{i=1}^{k} l_i}{n^k}$$

Since $l_i \ll n$, the probability P that a legitimate packet is misclassified as a suspicious one is rather low. For example, if $k = 4$, $n = 256$ and $l_i = 12$, $1 \leq i \leq 4$, then $P \approx 4.828 \times 10^{-6}$ which is a very acceptable false positive rate.

3.3 Construction of hash functions

Having a good set of independent hash functions is essential for good hash table performance. Ideally, each hash function in the Bloom filter should hash the keys to the table uniformly and two different keys should have low probability of collision. Moreover, the k hash functions are independent.

In practice, it is not easy to design a good hash function that distributes the keys uniformly and yet has low computational cost. Moreover, the distribution of input keys affects the distribution of the counter usage. The design of such hash functions will be studied in our future work.

In this paper, we focus on the design of independent hash functions that have low probability of collision. We use the 32-bit IP address *IP* as the key of the hash functions. The hash functions are defined as follows:

$$h_i(IP) = (IP + IP \bmod p_i) \bmod n, \quad 1 \leq i \leq k,$$

where mod denotes the modulus operation, n is the row length of the hash table, and p_i is a prime number less than n.

We now discuss the condition that makes two different keys collide in all k hash functions, i.e., for $IP_1 \neq IP_2$, $h_i(IP_1) = h_i(IP_2)$, $1 \leq i \leq k$. If $h_i(IP_1) = h_i(IP_2)$ and $h_j(IP_1) = h_j(IP_2)$ for $i \neq j$, that means

$$IP_1 + IP_1 \bmod p_i = IP_2 + IP_2 \bmod p_i + nk$$
$$IP_1 + IP_1 \bmod p_j = IP_2 + IP_2 \bmod p_j + nl,$$

for some integers k and l. Hence we have

$$IP_1 \bmod p_i - IP_2 \bmod p_i = IP_1 \bmod p_j - IP_2 \bmod p_j + n(k-l).$$

This condition is strict for two keys $IP_1 \neq IP_2$ to satisfy $h_i(IP_1) = h_i(IP_2)$ for all k hash functions. Thus we can conclude that the false positive rate should be very low.

The row length n of the hash table is chosen to be a power of 2. This choice allows the modulus operation to be applied by simple bit masking which is computationally much cheaper than performing the division operation.

4 Detection mechanism

During a spoofed flooding attack, it is expected that many incomplete handshakes will be observed and hence there are many more *SYN* packets than the corresponding *ACK* packets. As a result, some counters in T_d will have an abnormally high counter value. Under normal operation, these counter values should be close to 0. So there should exist a change point from low to high counter values when an attack is launched.

In this section, we address the problem of detecting change points in the probabilistic characteristics of random sequences. Cumulative Sum (CUSUM) is a sequential detection method which assumes that the mean value of some variable under surveillance will change from negative to positive value whenever a change occurs. In our case, CUSUM is applied to detect abrupt changes in T_d. After a packet is identified as suspicious, we analyze its source IP and classify it into one of three categories: random, subnet or fixed spoofing.

4.1 Change-point detection method

There exist two major change-point detection methods: posteriori change-point detection method and sequential change-point detection method. When the process of data acquisition is completed at the moment of checking, it is called a posteriori change-point method. On the other hand, the sequential method checks for change points online with observations, i.e., simultaneously with the process of data acquisition. The sequential method is preferred in spoofed attacking traffic since it works in an online manner.

The essence of sequential change-point detection is as follows. Suppose the observations of a random process X_t (with discrete or continuous time) are received sequentially. At a certain moment (random or not, but unknown), some probabilistic characteristics of this process change. An observer must make a decision as quickly as possible as to whether a change-point has happened or not, while keeping the false alarm rate to be as low as possible.

Suppose that a sequence $X_1, \ldots, X_r$ of independent random variables is observed. For each $1 \leq v \leq r$, consider the hypothesis H_v that $x_1, \ldots, x_{v-1}$ have the same density function $f_0(\cdot)$ and $x_v, \ldots, x_r$ have another density function $f_1(\cdot)$. Denote by H_0 a hypothesis of stochastic homogeneity of the sample. Then the likelihood ratio statistic for testing the composite hypothesis H_v $(1 \leq v \leq r)$ against H_0 is:

$$\max_{0 \leq k \leq r} (S_r - S_k) = S_r - \min_{0 \leq k \leq r} S_k,$$

where

$$S_0 = 0,\ \ S_k = \sum_{j=1}^{k} \log \frac{f_1(x_j)}{f_0(x_j)}.$$

The CUSUM statistic $g_r = S_r - \min_{0 \le j \le r} S_j$ can be written in the following recurrent form:

$$g_r = \left(g_{r-1} + \log \frac{f_1(x_r)}{f_0(x_r)} \right)^+, \quad g_0 = 0,$$

where $a^+ = a\, I(A)$ for some $a \ge 0$ and $I(A)$ is the characteristic function of the set A. The effectiveness of this statistic is easy to understand. The mathematical expectation of $y_r = \log(f_1(x_r)/f_0(x_r))$ is negative before and positive after the change-point.

The stopping rule for change-point detection is:

$$\tau = \inf \left\{ r \ge 1 : S_r - \min_{0 \le j \le r} S_j \ge b \right\},$$

where $b > 0$ is the alarm threshold.

There is a nonparametric version of the CUSUM statistic:

$$y_r = (y_{r-1} + x_r)^+, \ \ y_0 = 0,$$

and the corresponding decision rule is

$$d_N(\cdot) = d(y_r) = I(y_r > N),$$

where $I(\cdot)$ is the indicator function and N is the threshold. d_N is the decision at time r, which gives a value of 1 to indicate an attack and 0 to indicate a normal condition.

In general, $E(X_r) = c$. We choose a parameter a as the upper bound of c, i.e., $a > c$. Then we define $x_r = X_r - a$ so that it has a negative value during normal operation. When an attack takes place, the increase rate will suddenly become larger and the value $x_r = X_r - a$ will be positive.

4.2 Sequence model

Modeling of TCP connection requests is a difficult problem and the sequence X_t is not easily defined. It is not easy to devise a simple parametric model for TCP traffic due to its complicated characteristics. However, we assume that when an attack occurs, the traffic distribution will be different from that of normal traffic. We select the nonparametric version of CUSUM to perform online change-point detection, which has low computational cost.

Since the k rows in T_d are independent of each other, we choose one row to discuss for clarity. There are n counters in each row of table T_d and these counters are denoted by $C_1, C_2, \ldots, C_n$. The hash table is refreshed periodically. The value of counter C_i $(1 \le i \le n)$ forms a sequence $\{C_i^t\}$ according to the refresh period t. For each counter, we analyze the change-point of the sequence $\{C_i^t\}$.

Normal TCP traffic has symmetric *SYN* and *ACK* pairs and hence the counters in T_d should be close to 0. When spoofed TCP handshake packets are sent towards the victim, there will be more *SYN* packets than *ACK* packets. The corresponding counter value in T_d will grow rapidly.

The density function of sequence $\{C_i^t\}$ will change and CUSUM will detect this change. We define $x_i^t = C_i^t - a$, where $a > 0$, as the sequence of time t for the CUSUM method. The two parameters are set according to the network conditions. Generally, the parameter a is close to 0 since the counter value should be close to 0 when there exist no spoofed flooding attack and network error. Thus x_i^t has a negative value under normal operation. When a spoofed flooding attack occurs, it will change to a positive value. As for the parameter N which defines the alarm threshold, a large value leads to longer detection delay but lower false positive rate.

When a spoofed flooding attack occurs, each row is expected to have a counter with an abnormally high value. These abnormal counters are referred to as *heavy counters*. If a packet is hashed into T_d and hits all the k heavy counters in all k rows, this packet is regarded as a suspicious spoofed packet and hence an alarm will be launched.

4.3 Spoofing type classification and response

No matter what type of IP spoofing is used, packets can be detected with our change-point detection method. We can then identify the spoofing type by analyzing its source IP.

Since random spoofing generates a wide range of IP addresses for the source IPs of the packets, the probability that two packets have the same source IP is very low. On the other hand, subnet spoofing has a much narrower range than random spoofing. During a spoofed flooding attack, the attacking source typically generates a large number of packets. This number is much larger than the number of candidate IP addresses used for subnet spoofing. For example, in order to bring down a victim server for 10 minutes, the attacker needs to inject at least 300,000 *SYN* packets [17]. However, a class C subnet only has 254 IP addresses which are available for a subnet spoofing attack. Therefore, quite a number of subnet spoofed packets are expected to have identical source IPs during the attack period.

We define two thresholds, θ_1 and θ_2 ($1 < \theta_1 < \theta_2$), for T_s. When a packet hits k heavy counters in T_d, its source IP is checked in T_s. If all counters in T_s hit by this packet have values larger than θ_1 but smaller than θ_2, it is regarded as subnet spoofing. If the value is much larger than θ_2, it may be caused by fixed spoofing. Otherwise it is regarded as random spoofing. Another difference between random spoofing and subnet spoofing is the degree of overlap of the IP addresses. With random spoofing, the IP addresses hashed into the same counter are typically different as a result of hash collisions. With subnet spoofing, however, the IP addresses hashed into the same counter are usually the same. This difference can be used for distinguishing random spoofing from subnet spoofing.

Random spoofing may be throttled by ingress filters deployed at the edge routers. However, there has been a lack of efficient methods for fighting against subnet spoofing. To defend against attacks caused by fixed spoofing and subnet spoofing, we propose here a soft rate-limiting scheme. Specifically, the percentage of traffic to go through is equal to

$$R_{\mathtt{pass}} = \gamma^{-\varepsilon C_s},$$

where C_s is the value of a counter in T_s and γ, ε are parameters. The legitimate value of C_s is 0. Thus if C_s is close to 0, we essentially allow almost all traffic to pass through. According to the classification scheme above, a higher value of C_s means that the traffic has a higher chance of being packets caused by fixed spoofing. The larger the value of C_s is, the more traffic will be blocked. A lower value means that the traffic is more likely due to subnet spoofing. A small rate-limiting value is thus used to mitigate collateral damages since the traffic is more distributed and more legitimate traffic may hit these counters.

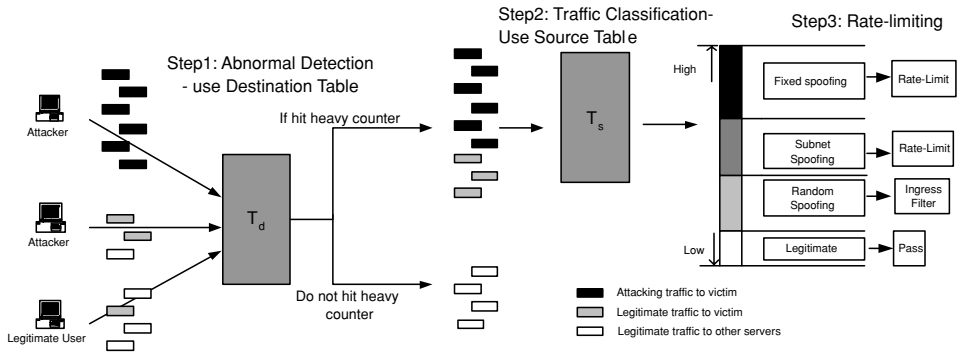


Fig. 4 Architecture of the detection and response scheme

Figure 4 shows the architecture of our detection and response scheme. The scheme consists of three main steps. In the first step, only the destination table T_d is used to make a traffic digest. The CUSUM method is then applied to detect SYN flooding attacks by detecting abnormal change-points in the traffic digest. In the second step, suspicious traffic is identified as either one of the spoofing types: random, subnet or fixed spoofing. The source table T_s helps to perform spoofing type classification. In the final step, different response methods are applied to defend against different types of IP spoofing. While ingress filters [6] can effectively filter packets caused by random spoofing, a flexible rate-limiting scheme is used to defend against the other two spoofing types according to the source IP distribution.

5 Performance evaluation

We have carried out some simulation experiments to evaluate the performance of our proposed method. We first evaluate the performance of our hash function and compare it with other hash functions with respect to the collision rate. We then use the DARPA off-line intrusion detection benchmark [9] for experiments on the detection of SYN flooding attacks with spoofing, which are commonly found on the current Internet. We use the NS2 network simulator to simulate the three IP spoofing types. Detection results are then separately analyzed for these different attacking scenarios, showing that our method is effective for all cases including the more subtle subnet spoofing type.

5.1 Hash function performance

Our hash function described in Section 3.3 has the advantages of low collision rate and high execution efficiency, which are particularly suitable for use in the Bloom filter. The proposed hash function is compared with Knuth's multiplicative hash function [8], Robert's 32- and 96-bit mixing hash functions [7] and a hybrid hash function. The hybrid hash function is composed of Knuth's hash function and Robert's hash function with different parameters, which are used as the k different independent hash functions in the Bloom filter. In our comparative study, we set n to 1024, k to 4, and the 100,000 IP addresses are randomly generated. These IP addresses are inserted into the hash tables for different hash functions.

The execution time and the number of collisions for each hash function are shown in Table 1. It can be seen that our hash function not only has the lowest (zero) collision rate like Robert's

Table 1 Comparison of hash functions

Hash function	Time consumed (sec)	Number of collisions
Our hash function	0.187	0
Robert's 32-bit hash	0.188	3838
Robert's 96-bit hash	0.250	0
Knuth's hash	0.031	977
Hybrid hash	0.328	0

96-bit hash function and the hybrid hash function, it is also the most efficient among these three methods. Although Knuth's hash function is even more efficient than ours, it has a higher collision rate and hence is less favorable. Our method, being both simple and efficient, is most suitable for networking applications as that studied here.

5.2 Attack traffic detection

Two datasets from the DARPA benchmark are used to evaluate the detection performance of our method. Specifically, one dataset contains *SYN* flood attack packets and the other dataset is free of flooding attack. The original DARPA datasets are in `tcpdump` format and TCP *SYN*, *ACK* and *ACK/SYN* packets are extracted from the datasets.

In all the experiments, the number of rows k in each hash table is set to 4 and the number of counters n in each row is set to 1024. For attack detection, we only need to monitor the destination table T_d. As before, since each row in T_d corresponds to an independent hash function, we only discuss any one row here as other rows are the same.

We first observe the status of T_d during normal operation. The parameters a and N are set to different values for our testing. Normal traffic begins at 21:00 on one day and ends at 18:32 on the following day. There are a total of 27877 TCP connections during this period. The numbers of false positives observed for different combinations of the parameters a and N are shown in Table 2. Compared to a total of 27877 connections, the number of false positives is actually quite small and tolerable. We observe that most of the counters have a zero value and only two counters (out of 1024 counters in the row) have nonzero values. The nonzero values are triggered by incomplete handshakes caused by occasional network errors. We select a counter with zero value and another counter with nonzero values to show the CUSUM change-point detection results. The results are depicted in Fig. 5. Since the traffic is very low around midnight, the result for this period is not shown for clarity.

In another experiment, the dataset with *SYN* flood attack packets is used. During the attack, there are many more *SYN* packets than *ACK* packets. The counters corresponding to the hashed values of the victim IP addresses grow substantially and hence become heavy counters. The detection rates and false positive rates under different parameter settings are shown in Table 3. When N is set to 2 and a to 1, the CUSUM detection rate is the highest. When N and a are set to other values, the detection rate is lower but remains almost the same for different settings. The reason is that the destination table T_d is refreshed periodically and change-point is checked

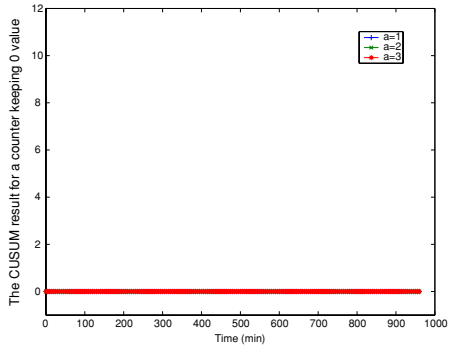
Table 2 Number of false positives

N	$a = 1$	$a = 2$	$a = 3$
2	13	4	2
3	4	2	0
4	2	0	0

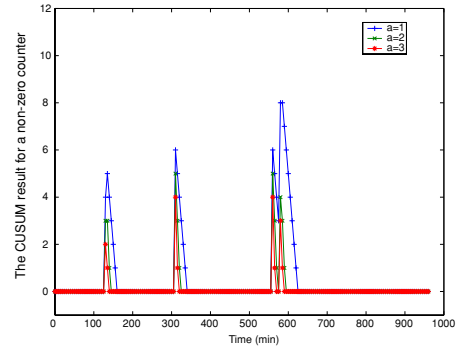
Table 3 Detection rate and false positive rate

N	$a = 1$	$a = 2$	$a = 3$
2	99.9%/1.21%	93.7%/0.95%	93.7%/0.76%
3	93.7%/1.19%	93.7%/0.84%	93.7%/0.69%
4	93.7%/1.09%	93.7%/0.76%	93.7%/0.65%

d/f refers to detection rate d and false positive rate f.



(a) CUSUM result of counter with zero value

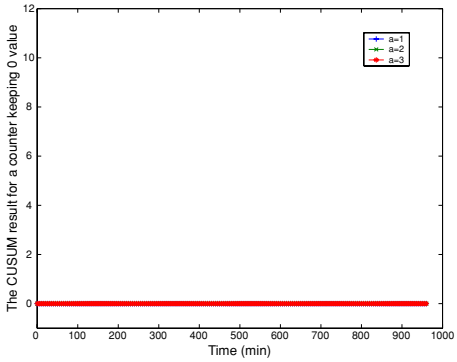


(b) CUSUM result of counter with nonzero values

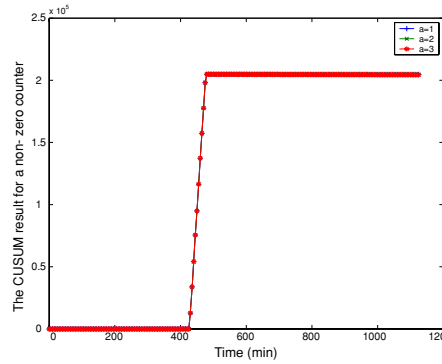
Fig. 5 CUSUM results of two counters when there is no attack

by CUSUM after each refresh. After a special refresh period, the CUSUM method can detect malicious *SYN* packets at almost the same time even for different N and a values. On the other hand, the false positive rate generally decreases as N or a increases.

While most of the counters still have a zero value, there are a few heavy counters with values significantly higher than those of the nonzero counters during normal operation as observed above. The CUSUM detection method can successfully detect these heavy counters. Figure 6

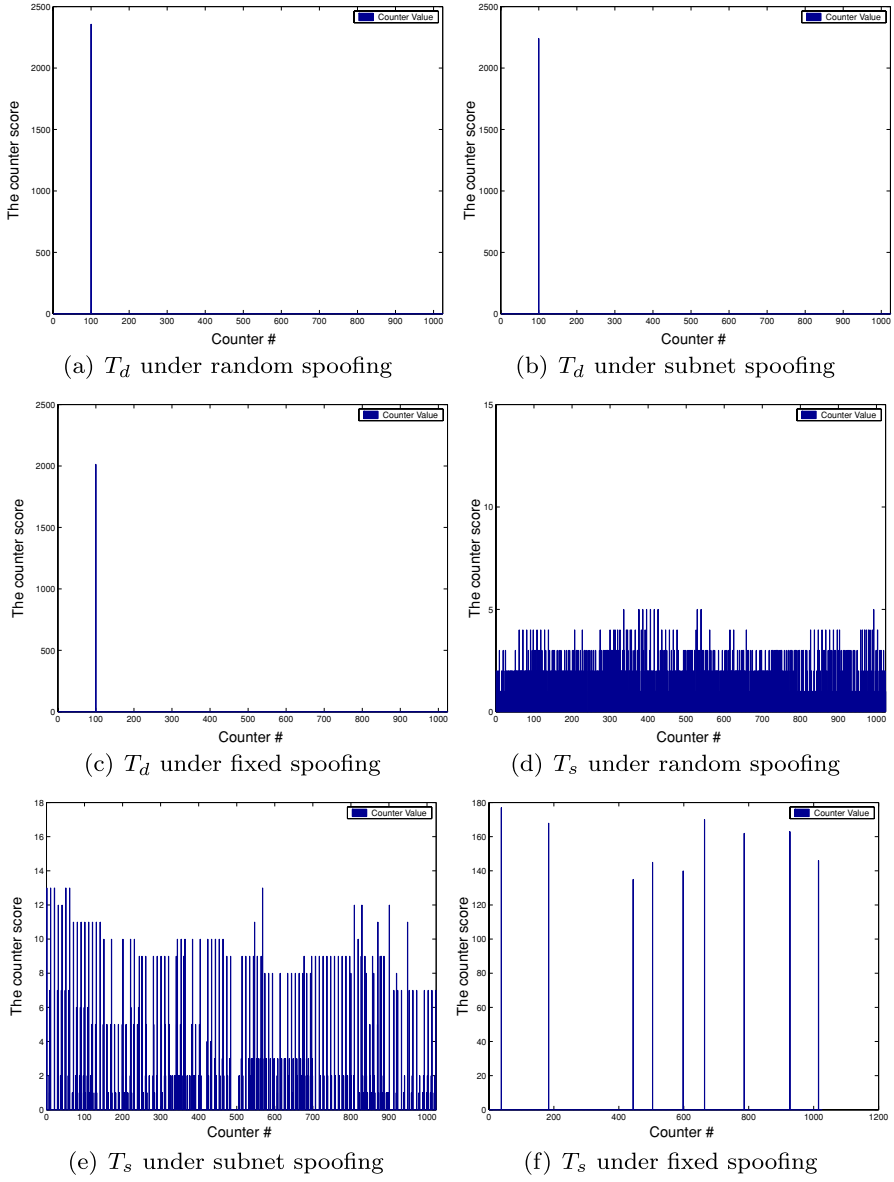


(a) CUSUM result of counter with zero value



(b) CUSUM result of counter with nonzero values

Fig. 6 CUSUM results of two counters during a SYN flooding attack



(a) T_d under random spoofing (b) T_d under subnet spoofing

(c) T_d under fixed spoofing (d) T_s under random spoofing

(e) T_s under subnet spoofing (f) T_s under fixed spoofing

Fig. 7 Counter values in one row each of tables T_d and T_s under different spoofing types

shows the CUSUM results of two counters when an attack is launched. Note that the scale of the vertical axis in Fig. 6(b) is much larger than that of Fig. 5(b).

5.3 Spoofing type classification

We use NS2 to simulate the random spoofing, subnet spoofing and fixed spoofing attack scenarios. In the simulations, there are 10 server nodes and 1000 client nodes. One of the server nodes is

selected as the victim. The client nodes randomly select one or more server nodes as destinations for establishing connections. Regardless of what spoofing type is used, we observe that the flooding traffic can trigger one counter in each row of table T_d to have an abnormally large value as shown in Fig. 7(a)–(c).

To classify the spoofing into one of the three types, we have to monitor the counter values in table T_s. Figure 7(d) shows the values of the counters in one row of T_s under random spoofing. The counters have relatively small values which are distributed somewhat uniformly across different counters because the spoofed IPs are generated randomly. For subnet spoofing as shown in Fig. 7(e), since a much narrower range of IP addresses is used, we can see that only a portion of the counters have nonzero values. Moreover, these counter values are generally higher than those in Fig. 7(d). If we examine more closely the two tables in Figs. 7(d) and 7(e), we can see that many different IP addresses are hashed into the same counter in Fig. 7(d) due to hash table collisions, but a counter in Fig. 7(e) almost always corresponds to the same IP address. Figure 7(f) shows the counter values under fixed spoofing. This type of spoofing is much easier to identify since only a few counters have nonzero values and they are heavy counters with extremely high values.

5.4 Comparison with other methods

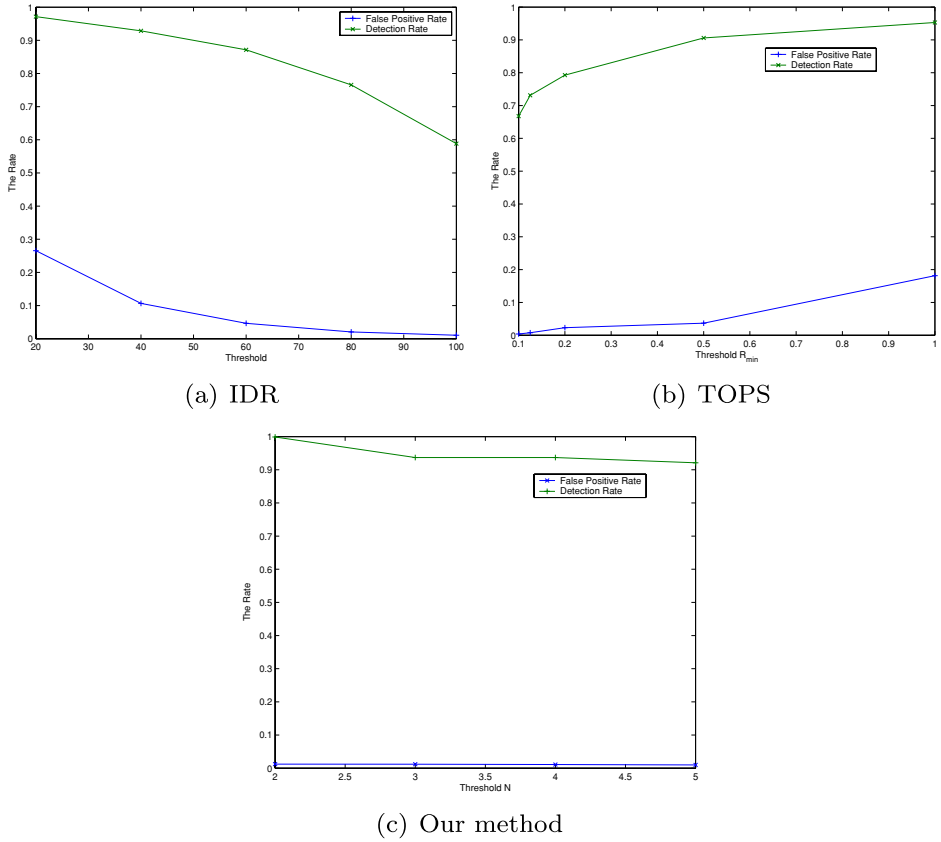
Our work bears resemblance to IDR [4] and TOPS [1], in the sense that these two methods also detect DDoS attacks based on a hashing scheme.

IDR (Intrusion Detection Router) monitors the traffic passing through a router and detects the occurrence of any exceptionally heavy volume of packets going to the same destination. IDR also employs a Bloom filter to detect malicious packets. A two-dimensional table of k levels by m bins with k independent hash functions is used to keep track of the recent arrival rates of packets of different destination IP addresses passing through a router within a sampling period t.

There exist some crucial differences between IDR and our method. First, IDR does not extract DDoS features from the network traffic but it records the destination IP information of all packets passing through the router. Our method only records symmetric packets using the TCP protocol, such as *SYN* and *ACK/SYN*. The fact that IDR records more information than our method implies that IDR requires more hash updating effort. Second, IDR identifies a counter as being suspicious simply by its counter value, making it difficult to set the detection thresholds and difficult to obtain very accurate results. On the other hand, our method analyzes the counter values with CUSUM and hence the results are more accurate. Moreover, IDR is only a detection method but ours performs both detection and response.

TOPS is a system based on the flow balance heuristic to detect and filter DoS bandwidth attacks. In particular, TOPS applies a form of Bloom filter based on a static tabular memory structure to detect attacks.

There also exist some crucial differences between TOPS and our method. First, like IDR, TOPS also does not perform feature extraction and hence incurs higher storage and hash updating costs. Second, if the attacker incorporates IP spoofing into the attacking packets, TOPS can no longer compute the ratio of incoming to outgoing traffic for each IP address. Third, TOPS divides an IP address into four octets and uses the octets as input for the hash functions. This scheme can hardly distribute the entries uniformly in the hash tables. For example, the packets from subnet 143.89.0.0/16 have the same octets 143 and 89. Therefore, octets 143 and 89 of all these packets are hashed into the same table.



(a) IDR (b) TOPS

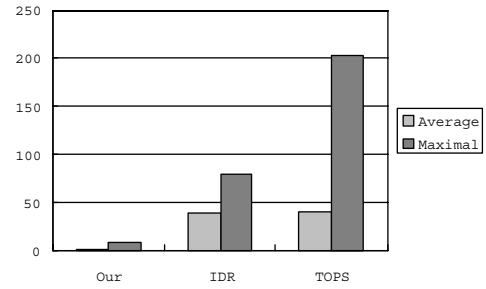
(c) Our method

Fig. 8 Detection rates and false positive rates for IDR, TOPS and our method for different thresholds

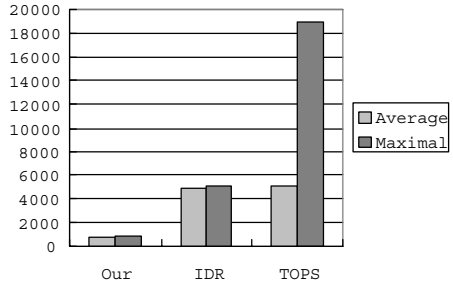
In our comparison, we use the DARPA off-line intrusion detection benchmark to evaluate the detection performance of IDR, TOPS and our method under different parameter settings. We use two datasets: one is free of any attack and the other contains SYN flooding attack packets.

Figure 8(a) shows the detection rates and false positive rates for different thresholds. For IDR, parameter setting follows the recommendation in [4], with the thresholds for all counters being the same. From Fig. 8(a), we can see that it is difficult for IDR to obtain high detection rate and low false positive rate simultaneously. When the threshold is set somewhat low, some packets sent to a popularly visited destination will be incorrectly recognized as suspicious ones. On the other hand, if the threshold is set too high, IDR will fail to detect some relatively low attacking traffic.

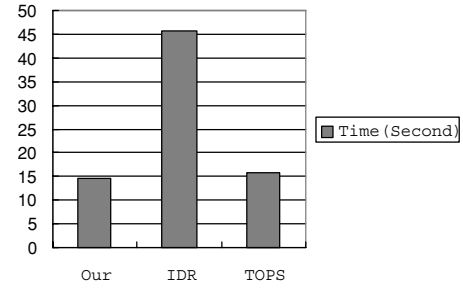
TOPS can hardly detect SYN flooding attacks near the victim side since the victim tries to respond to all SYN requests, including malicious ones. Therefore, the ratio of incoming to outgoing traffic cannot indicate abnormal traffic characteristics. In our experiments, TOPS is deployed near the attacking source and $R_{\min}$ is used as the threshold according to the recommendation in [1]. Figure 8(b) shows that TOPS, like IDR, cannot achieve high detection rate and low false positive rate simultaneously. Adjusting $R_{\min}$ can only improve one measure at the expense of the other measure. Compared to IDR, TOPS can give more accurate results as the ratio of incoming to outgoing traffic is a more reliable indicator for detection than the absolute counter values



(a) Counter value statistics under no attack



(b) Counter value statistics during attack



(c) Computational cost

Fig. 9 Storage and computational costs for IDR, TOPS and our method

especially when the traffic volume is high. We observe that many false positives are caused by some small counter values. The reason is that the ratio of incoming to outgoing traffic for these counters is very sensitive to changes in the traffic characteristics. However, since these counter values are small, their effect on the final results is relatively small.

Figure 8(c) shows the results of our method. We set a to 1 and N to different values as shown in x-axis. From the experimental results, our method clearly outperforms IDR and TOPS in that it can give very high detection rate even when the false positive rate is kept very low. Moreover, it can be seen that the performance is not very sensitive to the parameter setting, making it relatively easy to deploy.

We compare the storage costs of the three methods. Since all three methods use the same number of counters, the size of the hash table depends on the size of each counter. In order to allocate an appropriate memory size to each counter, let us first take a look at the counter value statistics of the three methods both under no attack and during attack.

Figure 9(a) shows that the counters in our method have values close to zero since the TCP traffic is mostly symmetric when there is no attack. The average counter value for TOPS is close to that for IDR, but its maximum counter value is significantly larger because the hash function of TOPS makes the distribution of counter values highly non-uniform. During an attack, the counter values for all three methods increase significantly but the maximum value for TOPS is much larger, as shown in Fig. 9(b). Our method can accurately detect DDoS attacks even with a small threshold value. Therefore, it suffices to allow for a relatively small range of values for each counter, implying that the memory size can be kept small. IDR can also use a small threshold value. However, since it does not extract any traffic feature but stores all packet information, it requires more storage than our method. As for TOPS, it needs to collect all the information in order to calculate the ratio of incoming to outgoing traffic. Moreover, the distribution of counter

values is highly non-uniform and hence the maximum counter value can be very large. As a result, TOPS has a higher storage requirement than IDR and our method.

We also compare the computational cost of the three methods. A dataset containing about 200,000 entries is used for evaluation. Figure 9(c) shows that our method has the lowest computational cost since it extracts DDoS features from the traffic and it does not need to update the hash table frequently. Besides the hash operation, our method only requires simple increment and decrement operations. IDR has the highest computational cost since it has to perform the hash operation for every packet. While TOPS also performs the hash operation for every packet, the hash function used is simple and hence the computational cost of TOPS is still low.

To summarize, our method can perform accurate detection yet with low storage and computational costs. Moreover, it is relatively insensitive to the parameter setting.

6 Related work

Hash tables are high-performance data structures that can be used for efficient table lookup and hence are particularly useful for many network packet processing applications. The Bloom filter is a kind of space-efficient hash data structure, which was first proposed by Bloom [2]. It has been used for network packet processing. For example, Song [16] presented a hash table data structure and a lookup algorithm based on an extended Bloom filter which can support better throughput for router applications using hash tables. Also, NetFlow [5] maintains a hash table of connection records in DRAM and monitors the network traffic. The concept of multiple hashing, which is similar to Bloom filter, is used to track large flows in the network traffic.

Hash tables have also been used for defending against DDoS attacks. Snoeren [15] presented a technique based on hash table for IP traceback, which generates audit trails for the network traffic so that the origin of an IP packet delivered by the network in the recent past can be traced. Hash table has also been employed to look for imbalance between the incoming and outgoing traffic flows to or from an IP address [1]. More recently, a router equipped with DDoS protection capability called IDR [4] was proposed to detect DDoS attacks using a Bloom filter.

Change-point detection methods have been applied to DDoS detection due to their simplicity and effectiveness. Wang et al. [17, 18] proposed a method for detecting SYN flood attacks at leaf routers that connect end hosts to the Internet. Based on the observation that *SYN* and *FIN* packets form pairs in normal network traffic, they proposed using a nonparametric CUSUM method to accumulate the pairs. Luo and Chang [10] proposed a two-stage scheme to detect the so-called pulsing DoS attacks. The first stage uses wavelet transform to extract the desired frequency components of the traffic data and the second stage tries to detect change points in the extracted components.

7 Conclusion

IP spoofing is a problem without any easy solution because it is inherent to the design of the TCP/IP suite. Although IP spoofing is not an attack in itself, it is commonly used with real TCP-based attacks by exploiting the characteristics of TCP/IP.

To defend against spoofed flooding attacks, we propose in this paper an efficient method that can detect all three types of spoofing: random, subnet and fixed spoofing. Based on the Bloom filter, we propose a storage-efficient data structure which only requires a fixed-length table for recording relevant traffic information. A change-point detection method, CUSUM, is then applied to detect abrupt changes in the traffic characteristics which correspond to the occurrence

of flooding attacks. When malicious events are detected, they can further be classified into random spoofing, subnet spoofing or fixed spoofing type by analyzing a hash table for the source IP characteristics. Simulation experiments show that our proposed method yields very accurate detection and classification results yet with low computational cost.

There are some parameters in our method. Currently these parameters are set manually based on experience. A future extension is to devise an automated scheme for setting or adapting these parameters. Another interesting direction to pursue is to design adaptive hash functions that maximize the utilization of the hash table entries and hence further reduce the false positive rate. Moreover, we plan to evaluate our method in a reasonably large real network.

References

1. S. Abdelsayed, D. Glimsholt, C. Leckie, S. Ryan and S. Shami, An efficient filter for denial-of-service bandwidth attacks, in: *IEEE Global Telecommunications Conference (GLOBECOM'03)* (2003), Vol. 3, pp. 1353–1357.
2. B.H. Bloom, Space/time trade-offs in hash coding with allowable errors, Communications of the ACM 13(7) (1970) 422–426.
3. B. Brodsky, *Nonparametric Methods in Change-Point Problems* (Kluwer Academic Publishers, The Netherlands, 1993).
4. E. Chan, H. Chan, K. Chan, V. Chan, S. Chanson and etc, IDR: An intrusion detection router for defending against distributed denial-of-service(DDoS) attacks, in: *Proceedings of the 7th International Symposium on Parallel Architectures, Algorithms and Networks 2004(ISPAN'04)* (2004), pp. 581–586.
5. C. Estan, K. Keys, D. Moore and G. Vargese, Building a better NetFlow, in: *ACM SIGCOMM* (2004), pp. 39–42.
6. P. Ferguson and D. Senie, Network ingress filtering: Defeating denial of service attacks which employ ip source address spoofing (2000).
7. B. Jenkins, A new hash functions for hash table lookup (1997).
8. D. Knuth, *The Art of Computer Programming*, volume 3 of *Sorting and Searching* (Addison-Wesley, 1975).
9. R. Lippmann, D. Fried, I. Graf, J. Haines, K. Kendall, D. McClung and other, Evaluating intrusion detection systems: The 1998 DARPA off-line intrusion detection evaluation, in: *the 2000 DARPA Information Survivability Conference and Exposition* (2000).
10. X. Luo and R.K.C. Chang, On a new class of pulsing denial-of-service attacks and the defense, in: *Network and Distributed System Security Symposium 2005(NDSS2005)* (San Diego, California, 2005).
11. J. Mirkovic and P. Reiher, A taxonomy of DDoS attack and DDoS defense mechanisms, ACM SIGCOMM Computer Communications Review 34(2) (2004) 39–54.
12. V. Paxson, End-to-end routing behavior in the Internet, IEEE/ACM Transactions on Networking 5(5) (1997) 601–615.
13. J. Postel, Transmission control protocol: DARPA internet program protocol specification, RFC 793 (1981).
14. C.L. Schuba, I. Krsul, M. Kuhn, E.H. Spafford, A. Sundaram and D. Zamboni, Analysis of a denial of service attack on TCP, in: *Proceedings of the 1997 IEEE Symposium on Security and Privacy*, IEEE Computer Society (1997), pp. 208–223.
15. A.C. Snoeren, Hash-based IP traceback, in: *Proceedings of the ACM SIGCOMM Conference* (ACM Press, 2001), pp. 3–14.
16. D.X. Song and A. Perrig, Advanced and authenticated marking schemes for IP traceback, in: *Proceeding of Annual Joint Conference of the IEEE Computer and Communications Societies(INFOCOM)* (2001), pp. 878–886.
17. H. Wang, D. Zhang and K.G. Shin, Detecting SYN flooding attacks, in: *Proceedings of Annual Joint Conference of the IEEE Computer and Communications Societies(INFOCOM)* (2002), Vol. 3, pp. 1530–1539.
18. H. Wang, D. Zhang and K.G. Shin, Change-point monitoring for the detection of dos attack, IEEE Transactions on Dependable and Secure Computing 1(4) (2004) 193–208.