High-Potency Models of LDoS Attack Against CUBIC + RED

Meng Yue, Jing Li, Zhijun Wu, and Minxiao Wang

***Abstract*— A TCP-targeted Low-rate denial of service (LDoS) attack exploits the vulnerabilities of TCP congestion control mechanism. CUBIC is the most widely used TCP congestion control algorithm today. CUBIC TCP increases the resilience against LDoS over traditional TCP. This paper explores high-potency patterns of the LDoS attack against CUBIC TCP under RED queue management scenario, and accordingly develops two attack models referring to maximizing attack potency. Theoretical analyses and extensive experiments are conducted to validate the proper function of the two models and evaluate their performance. Test results show that the two attack models can effectively throttle CUBIC TCP throughput. Under standard-configured network parameters, the number of TCP units damaged by one attack unit are up to about 21 and 26 respectively for our proposed two models, which represents an increase in attack potency of about 20%. And, our proposed models outperform the traditional attack model in terms of attack potency by at least 250%. In addition, with variations in different network parameters, these two models are still efficient and alternatively maximize the attack potency. Finally, an outline for the attack countermeasure is discussed. The present study offers a basis to explore new attack manners which may be exploited by attackers and excites defenders to develop new measurements against such attack.**

***Index Terms*— Low-rate denial of service, mathematical model, congestion control, attack performance.**

TABLE I
DEFINITIONS OF COMMONLY USED NOTATIONS

Symbol	Definition
δ	The attack pulse rate, given in bits per second [bps]
L	The attack pulse width, given in seconds
T	The attack pulse period, given in seconds
$cwnd$	The current window size of CUBIC TCP
$cwnd_{tcp}$	The current window size of standard TCP
$cwnd_a$	The window size when triggering packet loss
$cwnd_o$	The optimal $cwnd_a$ at which the attack model can maximize the potency
$Damage$	The damaged TCP throughput, given in bits
$Cost$	The attack traffic cost, given in bits
$Potency$	The attack potency, defined as $Damage/Cost$

I. INTRODUCTION

THE low-rate denial of service (LDoS) attack greatly degrades network services by precisely attacking system vulnerabilities. Commonly, such attack employs low average volume traffic to achieve the goal of making greatest damage at least cost, and hide its malicious behaviors. Up to now, researchers have exposed many sorts of LDoS attacks, such as TCP-targeted Shrew attacks [1], HTTP-oriented Tail attacks [2], Economic denial of sustainability (EDoS) attacks in cloud computing [3], [4], Slow TCAM (ternary content addressable memory) attacks against software defined network (SDN) [5]. This paper focuses on the TCP-targeted LDoS attacks. The traditional LDoS attacks exploit the vulnerabilities of the retransmission timeout (RTO) mechanism or additive increase multiplicative decrease (AIMD) mechanism. Attackers periodically send high-rate pulses to induce the packet loss, consequently, frequently trigger the TCP congestion control. As such, LDoS attacks greatly throttle the TCP window growth and severely impact the transmission rate of the victim TCP connection. Although the LDoS attack sends a high-rate pulse every period, the duration of the pulse is very short. Therefore, it has a very low average volume traffic, which makes it masked in background traffic and difficult to detect [1], [6], [7]. In practical, LDoS attacks seriously threaten the security of the internet due to the widespread utilization of TCP in applications and services, and the extremely hidden nature of LDoS attacks [1], [8].

CUBIC is the most widely used TCP congestion control algorithm in current operating system [9]. For example, Linux kernel after 2.6.18 and Windows 10 use CUBIC as the default TCP congestion control algorithm [10]. The random early detection (RED) algorithm is a typical active queue management algorithm used in routers [11]. CUBIC and RED cooperate to form a network congestion control feedback

Manuscript received January 7, 2021; revised April 30, 2021 and July 12, 2021; accepted September 17, 2021. Date of publication October 1, 2021; date of current version October 28, 2021. This work was supported in part by the National Natural Science Foundation of China under Grant 62172418, Grant U1933108, and Grant 61802276; in part by the Scientific Research Project of Tianjin Municipal Education Commission under Grant 2019KJ117; and in part by the Fundamental Research Funds for the Central Universities of Civil Aviation University of China (CAUC) under Grant 3122020076. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Mika Ylianttila. *(Corresponding author: Zhijun Wu.)*

Meng Yue, Jing Li, and Zhijun Wu are with the Department of Cybersecurity, Civil Aviation University of China, Tianjin 300300, China (e-mail: myue_23@163.com; 17121186390@163.com; zjwu@cauc.edu.cn).

Minxiao Wang is with the Department of Electrical and Computer Engineering, Southern Illinois University, Carbondale, IL 62901 USA (e-mail: minxiao.wang@siu.edu).

Digital Object Identifier 10.1109/TIFS.2021.3117066

system, which makes the congestion window to quickly and steadily converge to the available bandwidth, thus improving the transmission performance of TCP connections.

In contrast, attackers face severe challenges in the new scenario of CUBIC combined with RED.

1) Compared to traditional congestion control algorithms, CUBIC increases the TCP transmission rate and improves the link utilization. Even if the network is severely congested, CUBIC TCP can fast recovery its window [10]. Moreover, current operating systems usually use a shorter RTO [12]. The above two folds make the traditional RTO-based attacks less efficient [13], [14].

2) Since the RED algorithm does not make the router buffer full, the traditional full-buffer LDoS attack is no longer applicable [15].

3) The primary target of LDoS attacks is to maximize attack potency (i.e., take the lowest cost making the greatest damage). Here, potency represents how many TCP bits can be damaged by a single attack bit cost. To do so, attackers need to design precise attack models in the new scenario and explore the conditions for achieving the attack target.

Currently, few researchers paid attentions to LDoS attacks against CUBIC TCP. This paper devotes to reveal the patterns of the LDoS attack in the CUBIC + RED scenario, and accordingly explores attack models referring to maximizing attack potency. The main contributions of this paper can be summarized as follows.

1) We develop a double-pulse attack model by analyzing the TCP window behaviors, in the case of CUBIC congestion control combined with RED queue management.

2) We improve the double-pulse model by adjusting the attack period to form the single-pulse model which further simplifies the attack implementation and further upgrades the upper bound of the attack potency in some specific network configurations.

3) We conduct extensive measurements to assess the attack performance and reveal the effect of network parameter variations on the proposed two models.

Our study has practical significance, as CUBIC has replaced the traditional congestion control algorithms such as Reno, New Reno and BIC in current networks. It can be expected that this study will be of help in evaluating the extent of the attack's damage and the extent that existing countermeasures can mitigate the impact of the attack, and inspire defenders to develop new solutions against such attack.

The rest of this paper is organized as follows. Section II presents related works. Section III briefly reviews the CUBIC congestion control algorithm and the RED queue management mechanism, which are regarded as foundations of this study. Section IV proposes two attack models by analyzing the window behaviors of CUBIC TCP. Section V conducts NS-2 experiments to validate the effectiveness of the proposed models, and presents a comparison in their potencies. And then, it expands our experiments via extensive variations of network parameters to further evaluate the performance of the proposed two models. Section VI presents a brief discussion about attack countermeasures. Section VII finally concludes our contributions, and presents future research plans.

II. Related Works

The LDoS attack was first identified on the Internet2 Abilene backbone in 2001, and was official named in 2003 [1]. This attack has attracted widespread attentions since its appearance. So far, researchers have devoted many efforts to defending against such attacks [16], [17]. Recently, new technologies such as game theory [18], deep learning [19], cloud computing [20]–[22], and SDN [23]–[25] have been commonly used to construct LDoS detection and mitigation frameworks.

Meanwhile, it is also worth concerning the study on attack models, as it is closely related to the evaluation of the attack performance and the extent to which countermeasures have abilities in mitigating the impact of such attacks. Next, we specially review existing typical studies on attack models.

Kuzmanovic and Knightly [1] modeled the LDoS attack by periodic square pulses. And then, they extended the original model to form a double-rate model which can minimize the number of bytes transmitted by the attacker. The attacker firstly transmits the attack pulse at the maximum possible rate. Once the buffer fills, the attacker decreases its rate to the bottleneck rate to ensure continued packet loss using the lowest possible rate. Although the double-rate LDoS attack minimize the resources required by the attacker, authors unfortunately failed to discuss the relationship between the attack parameters and the network parameters and to present attack performance in detail. This results in inaccurate performance assessment in different network environment.

Sarat and Terzis [26] presented the effect of the router buffer size on the LDoS attack performance in 2005. They used a fluid model to correlate the router buffer size with the attack pulse rate required to cause damage. Mathematical analysis combined with simulation tests presented that smaller buffer size is indeed vulnerable to LDoS attacks, and a slight increment of the buffer size is able to mitigate the attack's impact. As the increment of the buffer size, LDoS attack has to increase its transmitting rate to fill a larger buffer to induce packet loss. In this case, attacker's resource cost is greatly increased as the attack is not low-rate any longer. The advantage of this strategy is simple to implement. However, the over-buffered router will cause extra queuing delay and consequently decline network throughput under normal situation.

Guirguis *et al.* [27] exposed two variants of the classic LDoS attack in 2006. Authors defined the damage-to-cost ratio as the attack potency, and proposed the full-capacity model and the full-buffer model to increase this metric. These two models first allow the normal TCP packets to occupy the pipe between the TCP sender and the receiver. And then, once the bottleneck link or the router buffer is full, the attacker only needs to fill the rest of the pipe vacancy to cause TCP packet loss. Therefore, the cost involved to mount the attack is reduced. The advantage of this work is to propose the metric of the attack potency to measure the attack performance, not just focus on the maximum absolute damage. The disadvantage of this work is its neglect of the queuing delay devoting to round trip time, which leads to several inaccurate assessments [28].

Zhang *et al*. [29] extended the LDoS attack to the Border Gateway Protocol (BGP) in 2007. BGP's routing session is established over the TCP connection between neighboring border routers, so BGP is also threatened by the TCP-targeted LDoS attack. Authors indicated that the LDoS attack could lead to session resets and delayed routing convergence, seriously impacting routing stability and network reachability. Moreover, authors proposed a distributed LDoS attack model that synchronized multiple attack streams with different round trip time aggregating at the victim to form the expected pulse shape for improving attack effectiveness and enhancing attack concealment.

Luo *et al*. [30] challenged the traditional LDoS model in 2014. Authors indicated that the model proposed by [1]ignored TCP's congestion window adaption, which inevitably failed to consider the impact of network environment (such as the bottleneck bandwidth, and one way delay) on TCP throughput. In consequence, many important properties about LDoS attack's effect cannot be revealed accurately by the traditional models. To overcome this problem, authors proposed a new mathematical model to reveal the relationship between attack properties and network parameters. This model can analyze LDoS attack's performance in a more precise way and reveal novel attack's properties. This work motived researchers to explore different attack patterns that may be utilized by an attacker, so that more effective defense strategies can be designed accordingly.

Ficco and Rak [31] improved the traditional LDoS attack to orchestrate the slowly-increasing-polymorphic DDoS Attack (SIPDA) against applications running in cloud. The main objective of SIPDA is affecting the cloud customer more on financial aspects than on the service availability. Authors modeled such attack as incremental rate pulses rather than constant rate. With this pattern, attacker can slowly enhanced the attack potency to inflict significant financial losses, and evade or greatly delay the detection. The main inspiration given by this work is considering the tradeoff between the damage (service degradation and cloud resources consumed caused by the attack) and the budget (cost of mounting such attack) from the perspective of cost-effectiveness.

Shan *et al*. [2] evolved the classic LDoS attack into the Tail Attack in 2017. The Tail attack exploited resource contention with dependencies among distributed nodes. Attacker disguised attack packets as legitimate hypertext transfer protocol requests and intermittently sent them to the victim to trigger millibottlenecks and cross-tier queue overflow. By doing so, the attack caused the long-tail latency problem in n-tier web applications while the servers are far from saturation. Authors modeled the attack impact based on queuing network theory to effectively conceal attack behaviors. Also, authors adopted a feedback control-theoretic framework that allowed such attack to fit the dynamics of background requests and system state by dynamically tuning the optimal attack parameters.

Cao *et al*. [32] utilized the classic LDoS attack to implement the CrossPath attack in SDN. In the in-band controlled SDN, the shared links between data traffic and control traffic expose vulnerabilities. Attackers can inject malicious data traffic to the shared links, which forces the TCP connection of the control channel to repeat retransmission timeout. Since the delivery of the control message is disrupted, the services and the applications in SDN controller are severely impacted. In addition, as the data traffic does not enter the controller, such attack is stealthy and difficult to be perceived by the controller.

Pascoal *et al*. [5] offered and investigated two types of slow attacks in SDN. The first one is the slow ternary content-addressable memory (Slow TCAM) attack, which exploits the vulnerabilities of the timeout mechanism of the flow rule. The attacker first injects malicious flow rules, and then reactivates them whenever the timeout timer overflows. By doing so, switch's TCAM memory is always occupied by these malicious flow rules, consequently, new legitimate rules are not allowed to be installed. The second attack is the slow saturation attack, which is formed by combining the slow TCAM and a low-rate saturation attack [33]. The slow saturation attack extends the impact of slow TCAM from the data plane to the control plane.

The above mentioned works present that many new attacks are based on the classic LDoS attack, and researchers have been paying attentions to the research of LDoS attack models. Existing studies mainly focuses on two folds: exploring available models in new scenarios and optimizing existing attack models to improving attack performance.

Rebecca and Hope [13] and our team [14] conducted experiments to test the effectiveness of the traditional LDoS attack on CUBIC TCP. Test results indicated that the traditional RTO-based LDoS attack model was less efficient for CUBIC. In experiments, the RTO was set to 1 s, the Droptail queue management was adopted, and the TCP throughput was used as a metric to assess the attack performance. Actually, RTO is often shorter (around 200–300 ms) in modern operation systems, and RED is commonly used for queue management. These cases make the attack more inefficient. In addition, these two works did not consider the attack potency, nor did accordingly give specific models in the CUBIC + RED scenario. As the wide use of CUBIC and RED challenges the traditional LDoS attack models, it is worth developing a new attack model suitable for the CUBIC + RED scenario. In addition, attack and defense is a dynamic game process (the winner is who consumes less resources), which inspired us to explore the upper-bound of the attack potency.

III. Background

The combination of CUBIC congestion control protocol and RED queue management algorithm plays a role in TCP utilization improvement and network congestion avoidance. In this section, we briefly review CUBIC and RED.

A. CUBIC Window Growth Function

CUBIC is the default TCP congestion control algorithm used in the Linux after kernel 2.6.18 and Windows 10. CUBIC relies on the time between consecutive packet loss events to adjust window. Compared with previous TCPs, CUBIC mainly modified the window growth pattern in the congestion avoidance state [34]. CUBIC uses a cubic function of the

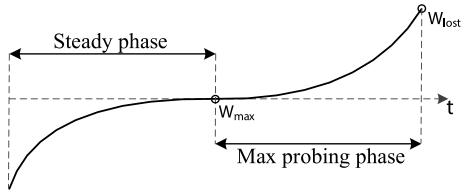

Fig. 1. Window growth function of CUBIC.

elapsed time from the last congestion event. A cycle of the CUBIC window growth includes a steady phase and a max probing phase, respectively corresponding to the convex profile and concave profile in Fig. 1 [10].

CUBIC defines W_{max} as the saturation point of the window, and ensures the window size to dynamically remain almost constant around W_{max} by a convex-concave alternating profile [10]. During the steady phase, the window increases fast initially when the windows size is far away from the saturation point. And then, when the window size is close to the saturation point, it switches to a slower increment as the network bandwidth is considered to approach saturation. During the max probing phase, the window size exceeds the saturation point, so it considers that the previous round saturation point is not accurate and a new saturation point need to be probed. In this case, the window increases slowly at the beginning to find the new saturation point, and if no packet loss, then it guesses the new saturation point is further away so it gradually increases its growth rate [10].

The details of CUBIC window behaviors can be described as follows. On packet loss, it registers W_{lost} to be the window size where the loss event occurred and sets the current window size *cwnd* to be $\beta \times W_{lost}$, where β is a constant factor used for decreasing window. The saturation point W_{max} depends on the phase where the packet loss event occurs. It can be expressed as follows [10].

$$W_{max} = \begin{cases} W_{lost} & \text{in max probing phase} \\ \dfrac{1+\beta}{2} \times W_{lost} & \text{in steady phase} \end{cases} \tag{1}$$

In Fig. 1, the window growth function is defined as follows [10].

$$W(t) = C \times (t-K)^3 + W_{max} \tag{2}$$

where C is a constant, t is the elapsed time from the last congestion event, and $K = \sqrt[3]{(1-\beta) \times Wmax/C}$ represents the time period for the window to increase from $W(t)$ to W_{max}, if no further packet loss.

CUBIC updates the window depending on the value of current window size *cwnd*. Upon receiving an acknowledgment (ACK), CUBIC sets $W(t+ RTT)$ as the target window size, meanwhile, checks whether the current window size *cwnd* is less than $W_{tcp}(t)$, where RTT is the round trip time, and $W_{tcp}(t)$ represents the window size of standard TCP in terms of the elapsed time t. If this is true, CUBIC sets *cwnd* as $W_{tcp}(t)$ for the TCP-friendliness. Otherwise, CUBIC adjusts the growth rate of window according to *cnt*. The above mentioned behaviors can be expressed as follows [10].

$$cwnd = \begin{cases} W_{tcp}(t) & \text{if } cwnd < W_{tcp}(t) \\ cwnd + \dfrac{1}{cnt} & \text{else} \end{cases} \tag{3}$$

Here, $W_{tcp}(t)$ is defined as follows [10].

$$W_{tcp}(t) = \beta \times W_{max} + 3 \times \frac{1-\beta}{1+\beta} \times \frac{t}{RTT} \tag{4}$$

And, *cnt* represents a count of ACKs that needs to be received before the window growth, and it depends on *cwnd* and the target size. *cnt* can be expressed as follows [10].

$$cnt = \begin{cases} \dfrac{cwnd}{W(t+RTT) - cwnd} & \text{if } cwnd < W(t+RTT) \\ 100 \times cwnd & \text{else} \end{cases} \tag{5}$$

Equation (4) implies that CUBIC window follows a linear growth in TCP-friendliness mode, which behaves very similarly to standard TCP [35]. Equation (5) implies that the window is allowed to increase to the target size per RTT, meanwhile, the window increases by one segment every time *cnt* ACKs are received. This presents the CUBIC window growth independent of RTT [36].

The effectiveness of CUBIC outperforms other traditional TCPs in terms of transmission rate and stability, mainly due to two reasons: 1) the cubic window growth function is faster than the traditional linear growth (additive increase) function (except the TCP-friendliness mode); 2) when congestion occurs, the window size is no longer halved but set to be $\beta \times W_{lost}$, where β is commonly greater than 0.5.

B. RED Packet Drop Mechanism

RED is the most classic AQM algorithm which is recommended by Internet Engineering Task Force (IEIF), and has also been widely implemented in network devices [37], [38]. Many variants of RED (e.g., ERED [39], LRED [40], DRED [41]) also inherit the core framework of RED.

The primary target of RED is to provide congestion avoidance depending on the average queue length which is calculated by a low-pass filter with an exponential weighted moving average (EWMA). For each packet arrival, RED updates the average queue as follows [11].

$$avg = (1-w) \times avg + w \times q \tag{6}$$

where avg is the average queue length, q is the instantaneous queue length, and w is the weight.

After that, avg is compared with a minimum threshold min_{th} and a maximum threshold max_{th}. Based on this, RED adjusts the packet dropping probability P_b as follows [11].

$$P_b = \begin{cases} 0 & \text{avg} < min_{th} \\ P_{max} \times \dfrac{avg - min_{th}}{max_{th} - min_{th}} & min_{th} \le avg < max_{th} \\ 1 & avg \ge max_{th} \end{cases} \tag{7}$$

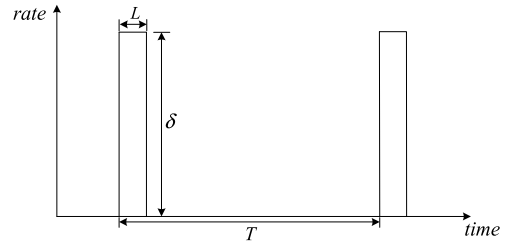

Fig. 2. Modeling of the classic LDoS attack: pulse rate δ, pulse width L, and pulse period T.

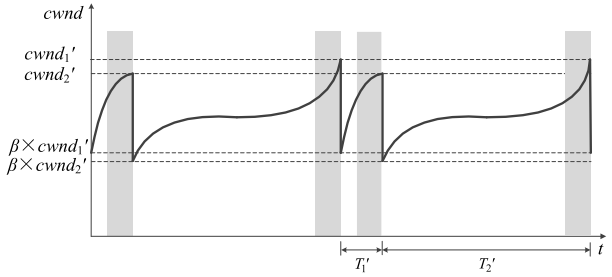

Fig. 3. D-Model: triggering packet loss in the max probing phase and the steady phase alternatively. Fig. 3 depicts the variation of the *cwnd* with respect to t. We divide the attack period T' into two sub-periods T_1' and T_2' to analyze the window behaviors under attack conditions.

Finally, RED calculates the final packet dropping probability P_a as follows [11].

$$P_a = \frac{P_b}{1 - count \times P_b} \tag{8}$$

where *count* is the number of packets received since the last dropping. If $avg \geq max_{th}$, *count* is set to zero.

Equation (7) and (8) indicate that, RED will drop all of packets when *avg* is greater than max_{th}, so the router buffer will not be full. Therefore, it is impossible to implement a full-buffer attack by completely filling the router buffer.

IV. Attack Models

Fig. 2 presents the classic LDoS attack model which is described by a sequence of square-pulse with triple parameters: a high rate δ, a short width L, and an appropriate period T [1]. Although the LDoS attack in this study follows the above mentioned model, we will specifically give a quantitative design for the triple parameters to maximize the attack potency in the CUBIC + RED scenario.

The essential task in this section is to discuss how to well configure these three parameters to achieve expected attack effects. We first analyze the packet drop probability model under RED queue. And then, we present two attack models, where a double-pulse model (D-model) forces the TCP congestion window to frequently alternate between max probing phase and steady phase, and a single-pulse model (S-model) is simpler to implement because it only sends one pulse per period.

A. Packet Loss Probability Model

CUBIC and RED form a congestion feedback system. The attack strategy is to cause packet loss in RED queue, and then the congestion signal is fed back to the end system to trigger CUBIC congestion control. Accordingly, TCP sender's window size is forced to remain at a low level. We assume that the RED packet dropping probability is P_a which can be calculated by (7) and (8). Under P_a, the probability that guarantees at least one TCP packet loss is P_t. P_t can be expressed as follows.

$$P_t = 1 - (1 - P_a)^{cwnd} \tag{9}$$

where *cwnd* represents the window size when the packet loss occurs.

Equation (9) can be explained as follows. The dropping rate of each packet is P_a, so the probability of not being dropped is $1 - P_a$. TCP sends *cwnd* packets per RTT, so the probability that all TCP packets are not dropped is $(1 - P_a)^{cwnd}$, and the probability of ensuring that at least one packet is dropped is $1 - (1 - P_a)^{cwnd}$. Equation (9) implies that the LDoS attacker must guarantee P_a large enough to obtain sufficient P_t. Meanwhile, a large *cwnd* facilitates P_t to be satisfied. Based on the above analysis, the attack pulse rate and the pulse width can be set as follows.

$$\begin{cases} \delta = \dfrac{C_b}{1 - P_a} \\ L = RTT \end{cases} \tag{10}$$

where C_b is the bottleneck link capacity given in bits per second [bps]. First, each attack packet is also dropped with probability P_a. Some of these attack packets are dropped by RED, and the remaining attack packets not dropped should guarantee to fill the bottleneck link capacity. After that, extra packets can be buffered to maintain P_a, so that the required P_t given by (9) can be obtained (i.e., guarantee at least one TCP packet loss). Second, the pulse width should set to be at least one RTT duration to guarantee sufficient P_t, since TCP employs the RTT duration as the time scale to send *cwnd* packets.

Here, we only explore the pulse rate δ and the pulse width L. Next, we will design two attack models by adjusting the pulse period T.

B. Double-Pulse Model

Equation (1) implies that the congestion event will alternatively occur in the max probing phase and the steady phase. This inspired us to design a double-pulse attack model with alternate pulse intervals. Fig. 3 depicts the window behaviors during an attack period.

In order to limit the growth of the CUBIC window, the attacker sends two pulses with intervals of T_1' and T_2' during each period T' ($T' = T_1' + T_2'$). We assume the first packet loss (triggered by the first pulse) occurs at the window size of $cwnd_1'$ corresponding to the max probing phase, and the second packet loss (triggered by the second pulse) occurs at the window size of $cwnd_2'$ corresponding to the steady phase. According to CUBIC algorithm (see Fig. 1 and equation (1)), in order to allow the packet loss to alternately occur between the stated phases, $cwnd_2'$ should be set to be

less than $cwnd'_1$. Additionally, $cwnd'_2$ should be set as large as possible according to (9). Therefore, $cwnd'_2$ should be quantified as $cwnd'_2 = cwnd'_1 - 1$. Next, we analyze window behaviors during T'_1 and T'_2.

T'_1: In the max probing phase where the window size is $cwnd'_1$, the attacker launches the first pulse with well-configured δ and L given by (3). This pulse will induce TCP packet loss and cause $cwnd$ to be $\beta \times cwnd'_1$ and W_{max1} to be $cwnd'_1$. After that, the window enters the steady phase and increases following the cubic function given by (2). As expected, the window size will increase to $cwnd'_2$ after the time period of T'_1. Therefore, $cwnd'_2$ can be expressed as follows.

$$\begin{aligned} cwnd'_2 &= \max[cwnd(T'_1), cwnd_{tcp}(T'_1)] \\ &= \max[C \times (T'_1 - K)^3 + W_{max}, \beta \times W_{max} \\ &\quad +3 \times \frac{1-\beta}{1+\beta} \times \frac{T'_1}{RTT}] \end{aligned} \tag{11}$$

where $K = \sqrt[3]{(1-\beta) \times cwnd'_1/C}$. According to (11), T'_1 can be deduced as follows.

$$T'_1 = \min\left[\sqrt[3]{\frac{cwnd'_2 - cwnd'_1}{C}} + K,\ \frac{cwnd'_2 - \beta \times cwnd'_1}{\frac{1-\beta}{1+\beta} \times \frac{3}{RTT}}\right] \tag{12}$$

T'_2: In the steady phase where the window size is $cwnd'_2$, the second pulse is launched to induce TCP packet loss again. The TCP sender decreases $cwnd$ to be $\beta \times cwnd'_2$ and sets W_{max2} to be $(1+\beta) \times cwnd'_2/2$. And then, the window behaves from the steady phase to the max probing phase given by (2). After the time period of T'_2, once the window size returns to $cwnd'_1$, the next round of attacks will be initiated. As analysis above, $cwnd'_1$ can be expressed as follows.

$$\begin{aligned} cwnd'_1 &= \max[W(T'_2), W_{tcp}(T'_2)] \\ &= \max[C \times (T'_2 - K)^3 + W_{max2}, \beta \times W_{max2} \\ &\quad +3 \times \frac{1-\beta}{1+\beta} \times \frac{T'_2}{RTT}] \end{aligned} \tag{13}$$

where $K = \sqrt[3]{(1-\beta) \times cwnd'_2/(2 \times C)}$. Accordingly, T'_2 can be deduced as follows.

$$T'_2 = \min\left[\sqrt[3]{\frac{cwnd'_1 - (\frac{1+\beta}{2}) \times cwnd'_2}{C}} + K,\ \frac{cwnd'_1 - \beta \times cwnd'_2}{\frac{1-\beta}{1+\beta} \times \frac{3}{RTT}}\right] \tag{14}$$

Note that, (12) and (14) consider the TCP-friendliness mode. This is because that attack pulses force the window to remain at a low level, which may cause $cwnd$ is less than $W_{tcp}(t)$. Experiment results in Section V will reveal this.

C. Single-Pulse Model

Since the window size is given in packets, the calculated $cwnd$ is rounded down (i.e., downwards to the nearest integer) [42]. In this case, although the window enters the max probing phase where $cwnd = W_{max}$, the window size will not increase immediately, that is, it will increase by one segment after a certain time. Accordingly, we design the single-pulse model shown in Fig. 4.

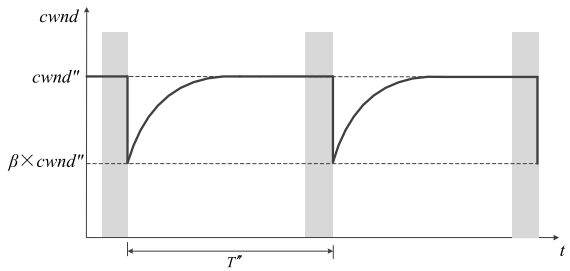

Fig. 4. S-Model: simplicity of D-model by exploiting the rounding. Fig. 4 depicts the variation of the $cwnd$ with respect to t.

We assume that the window size is $cwnd''$ when the attack pulse triggers the packet loss. Here, the window is in max probing phase, so we get $W_{max} = cwnd''$, and $cwnd = \beta \times cwnd''$. In order to extend the attack period as long as possible, the attack pulse causes the packet loss to occur within an RTT before the window size is close to $cwnd'' + 1$. Here, $cwnd'' + 1$ is the calculated window value, while the practical window size remains $cwnd''$ until the next packet loss.

Since the window takes T'' to increase from $\beta \times cwnd''$ to $cwnd'' + 1$, $cwnd'' + 1$ can be expressed as follows.

$$\begin{aligned} cwnd'' + 1 &= \max[cwnd(T''), cwnd_{tcp}(T'')] \\ &= \max[C \times (T'' - K)^3 + W_{max}, \beta \times W_{max} \\ &\quad +3 \times \frac{1-\beta}{1+\beta} \times \frac{T''}{RTT}] \end{aligned} \tag{15}$$

where $K = \sqrt[3]{(1-\beta) \times cwnd''/C}$. Accordingly, T'' can be deduced as follows.

$$T'' = \min\left[\sqrt[3]{\frac{(cwnd'' + 1) - W_{max}}{C}} + K,\ \frac{(1-\beta) \times cwnd'' + 1}{\frac{1-\beta}{1+\beta} \times \frac{3}{RTT}}\right] \tag{16}$$

As long as attack pulses are launched with the period T'', the TCP window size will be limited. Although the double-pulse model is not difficult to implement, the single-pulse model further simplifies the implementation of the attack as it only sends one pulse per period.

V. Experiments and Assessments

This section establishes a NS-2 simulation environment with a standard network configuration to validate the effectiveness of our two models, and to compare their attack performance. NS2 is commonly used by relevant studies to conduct experiments [1], [26], [43]. We can use it to conveniently configure networks and measure performance. More importantly, NS2 provides substantial support for simulations of CUBIC and RED. The experimental network topology and related parameters settings are shown in Fig. 5.

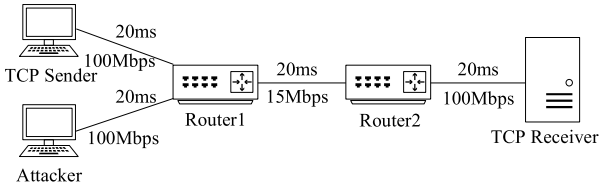

Fig. 5. Network topology of NS-2 simulations to verify the proper functioning of D-model and S-model, and to assess their attack performance.

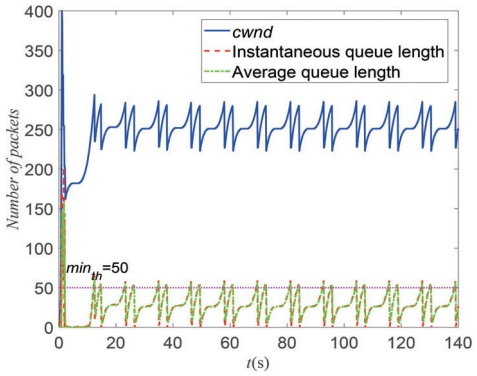

Fig. 6. The window and queue behaviors under normal conditions.

Here, a typical dumb-bell topology is established, where a single CUBIC TCP flow passing through a bottleneck link between two routers. The bottleneck link capacity is set as 15 Mbps and the round trip time is set as 120 ms. The router buffer size is set as the bandwidth-delay product (BDP). The RED parameters are set as $min_{th} = 50$, $max_{th} = 150$, $P_{max} = 0.06$, and $w = 0.002$. A legitimate CUBIC TCP connection is established between TCP sender and TCP receiver. The packet MTU size of TCP is set as 1000 bytes [1], [16], [44]. Attacker in Fig. 5 is in charge of sending LDoS attack stream which is formed by user datagram protocol packets. The attack packet MTU size is set as 50 bytes [1], [16], [44]. In addition, the attack parameters δ and L are set according to (8).

In our experiments, the TCP window and router queue behaviors are first traced under normal conditions. The simulation results are shown in Fig. 6, which are consistent with theoretical behaviors.

Fig. 6 shows that the window growth follows the cubic function referring to time t. Packet loss alternately occurs between the max probing phase and the steady phase, so that the window is dynamically stabilized at the saturation state. Fig. 6 also depicts queue behaviors referring to time t. The queue length is closely related to the window size. The increasing window gradually fills the queue until RED's packet dropping mechanism triggers a packet loss event, and then the queue length quickly decreases to almost zero. In addition, the queue length is always small, which implies small queuing delay and no severe network congestion. Also, the queue always has packet to drain (not empty), which implies the queue is fully utilized. Therefore, the given RED parameters works well in our experimental scenario.

A. Verification of Attack Models

The congestion control is triggered by packet loss, so the attacker first needs to select an appropriate window size to guarantee the packet loss. According to (9), P_t depends on P_a and *cwnd*. During the attack pulse, the packet dropping probability will reach its upper-bound, so we assume P_a equal to P_{max}. When P_t is determined, a larger P_a can decrease the value of the required *cwnd*. Fig. 7 presents P_t obtained based on (9) as a function of *cwnd*.

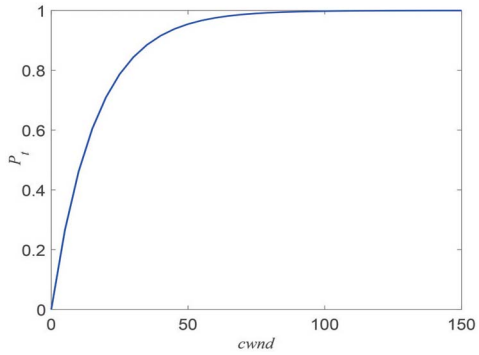

Fig. 7. P_t obtained as a function of *cwnd*.

As shown in Fig. 7, if it is believed that P_t is sufficient large when P_t is greater than 0.95, the corresponding minimum *cwnd* is about 50 under the given P_{max}. Therefore, we trigger the packet loss at $cwnd = 50$ (i.e., $cwnd'_2 = 50$ for D-model and $cwnd'' = 50$ for S-model). Note that, a smaller *cwnd* has advantages in overwhelming the TCP throughput. In contrast, a larger *cwnd* can increase P_t and extend the attack period, thereby reducing the cost of attack bits.

To verify D-model, we let two attack pulses to trigger packet loss respectively at $cwnd'_1 = 51$ and $cwnd'_2 = 50$, and set the pulse intervals according to (12) and (14). In Fig. 8, the window behaviors during a single attack period is traced. Compared to Fig. 6, the window size presented in Fig. 8 is restricted at a low level, which reflects the attack's effectiveness.

Fig. 8 (a) indicates that the test results are consistent with the theoretical analysis presented in Fig. 3 referring to a single attack period. First, as shown in Fig. 8 (b), the period 42.73–44.31 s corresponding to T'_1 lasts 1.58 s which is consistent with the theoretical value given by (12). During this period, the window is in the steady phase. As *cwnd* is not less than $cwnd_{tcp}$, the window follows the cubic function increase from 40 to 50, which agrees with the theoretical value given by (11). Second, as shown in Fig. 8 (c), the period 44.51–48.52 s corresponding to T'_2 lasts 4.01 s which is consistent with the theoretical value given by (14). During this period, the window increases from the steady phase to the max probing phase. During the steady phase, *cwnd* is not less than $cwnd_{tcp}$, the window follows the cubic function increment from 39 to 45. Once the window size increases to the critical point at 46.32 s, the window enters into the max probing phase. After that, as the theoretical *cwnd* given by (13) is less than $cwnd_{tcp}$, the practical *cwnd* presents the TCP-friendliness that the window follows the linear increment from 45 to 51.

To verify the S-model, we let the attack pulse to trigger packet loss at $cwnd'' = 50$, and set the pulse period according to (16). Fig. 9 shows the window behaviors traced in a single

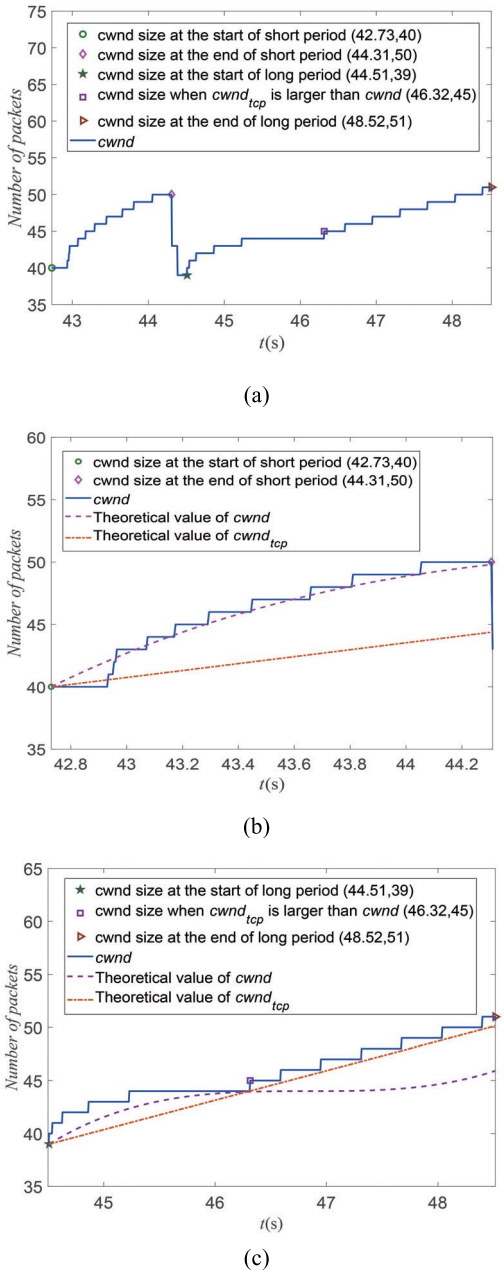

Fig. 8. Test results for D-model. (a) Overview of the window during an attack period. (b) Detailed behaviors over 42.73–44.31 s. (c) Detailed behaviors over 44.51–46.32–48.52 s.

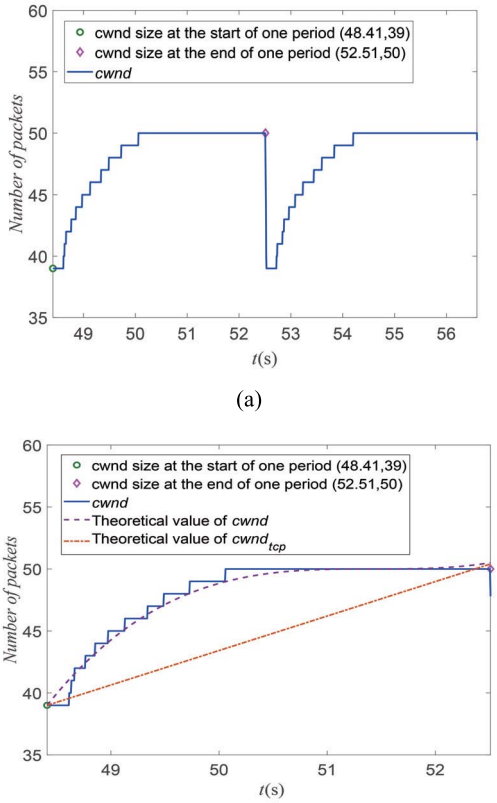

Fig. 9. Test results for S-model. (a) Overview of the window behaviors during an attack period. (b) Detailed behaviors over 48.41–52.51 s.

attack period. Compared to Fig. 6, the window growth in Fig. 9 has been severely restricted.

Fig. 9 (a) indicates that the test results are consistent with the theoretical behaviors shown in Fig. 4 referring to a single period. As shown in Fig. 9 (b), the period 48.41–52.51 s corresponding to T'' lasts 4.10 s which agrees fairly well with the theoretical value given by (16). During this period, the window is in the steady phase. As *cwnd* is not less than $cwnd_{tcp}$, the window follows the cubic function increment from 39 to 50, which agrees with the theoretical value calculated by (15). In addition, Fig. 9 (b) verifies the rounding mechanism, based on which, the window size remains 50 for a period of time until 52.51 s. At 52.51 s, the window should have increased by one segment, but the attack pulse causes the packet loss, so the window decreases to 39.

B. Performance Comparison

This subsection compares the attack performance of the proposed models. To this end, three basic indicators used for the assessment of the attack effect are defined.

The first indicator is the attack damage (termed as *Damage*) which represents the volume of bits that could have been sent, but were not successfully transmitted. *Damage* is defined as follows.

$$Damage = TP_o - -TP_a \tag{17}$$

Here, TCP throughput (*TP*) denotes the average number of successfully transmitted TCP bits per second, and accordingly, TP_o and TP_a denote the TCP throughput under normal conditions and attack conditions. The damage depends on window behaviors, because the window size determines how many TCP packets can be successfully transmitted. Under normal conditions, as long as the RED parameters are well configured, it can be guaranteed that the router queue is not empty, that is, the link bandwidth can be fully utilized. Therefore, TP_o is equal to C_b.

The second indicator is the attack cost (termed as *Cost*) that represents the average number of attack bits required

per second. *Cost* is defined as follows.

$$Cost = \frac{L \times \delta}{T} \tag{18}$$

Accordingly, we define the third indicator as the attack potency (termed as *Potency*) which is expressed as follows.

$$Potency = \frac{Damage}{Cost} \tag{19}$$

Potency represents the ratio of the damage induced by the attack to the attacker's cost for mounting the attack. In other words, it represents how many TCP units can be damaged by a single attack unit. This indicator reflects the tradeoffs between damage and cost. For D-model, TP'_a can be given as follows.

$$TP'_a = packetsize_{tcp} \times \frac{G'_1 + G'_2}{T'_1 + T'_2} \tag{20}$$

where G'_1 and G'_2 can be expressed in (21) and (22), as shown at the bottom of the page.

T'_1 and T'_2 are given by (12) and (14). $packetsize_{tcp}$ represents the TCP packet size. In addition, the cost of D-model can be given as follows.

$$Cost' = \frac{L \times \delta}{T'_1 + T'_2} \tag{23}$$

For S-model, TP''_a can be given as follows.

$$TP''_a = packetsize_{tcp} \times \frac{G''}{T''} \tag{24}$$

where G'' can be expressed in (25), as shown at the bottom of the page.

T'' is given by (16). And, the cost of S-model can be given as follows.

$$Cost'' = \frac{L \times \delta}{T''} \tag{26}$$

Next, five attack periods are randomly selected in experiments to report *Damage* and *Potency*. Simulation results are shown in Figs. 10 and Fig. 11.

We note from Fig. 10 that D-model induces average 12.15 Mbps TCP damage, and S-model induces average 11.99 Mbps TCP damage. These results demonstrate that the D-model and S-model reduce TCP throughput by approximately 81% and 80%, respectively, in the given experimental scenario. Also, we note from Fig. 11 that about 18 CUBIC TCP bits are damaged by one attack bit of D-mode, and about 23 CUBIC TCP bits are damaged by one attack bit of S-model.

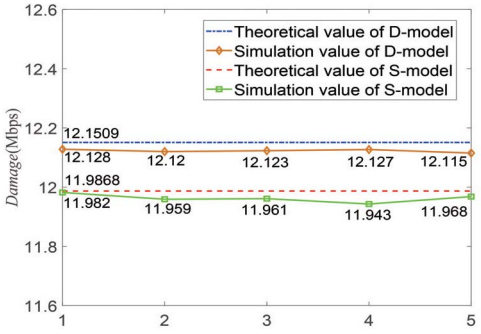

Fig. 10. Comparison of *Damage* for D-model and S-model.

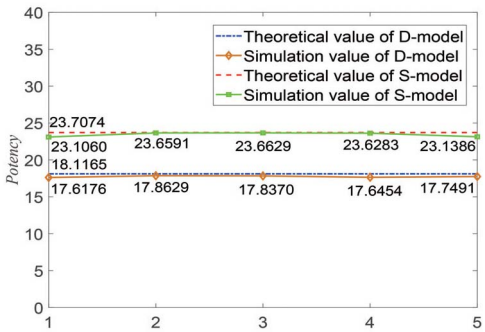

Fig. 11. Comparison of *Potency* for D-model and S-model.

In order to validate the effectiveness and assess the performance of the two models in a practical scenario, we establish a test-bed experimental environment with the same network topology and parameter configurations as Fig. 5. We use two multi-network-card PCs with Linux Red Hat operation system (version 2.6.39) to replace the two routers. We use two tools iproute and tc installed in the PC to configure the queue, the propagation delay, and the bottleneck link capacity. All machines are configured with Intel Xeon Quad-Core CPU E5504 @ 2.0GHz and 4GB RAM. We records the averages of ten attack periods of tests performed in the test-bed experiments. The average *Potency* obtained for D-model and S-model is 17.77 and 23.43 respectively. These test-bed experimental results are similar to NS-2 experimental results,

$$G'_1 = \int_0^{T'_1} \frac{\max[C \times (t-K)^3 + W_{max1}, \beta \times W_{max1} + 3 \times \frac{1-\beta}{1+\beta} \times \frac{t}{RTT}]}{RTT} dt \tag{21}$$

$$G'_2 = \int_0^{T'_2} \frac{\max[C \times (t-K)^3 + W_{max2}, \beta \times W_{max2} + 3 \times \frac{1-\beta}{1+\beta} \times \frac{t}{RTT}]}{RTT} dt \tag{22}$$

$$G'' = \int_0^{T''} \frac{\max[C \times (t-K)^3 + W_{max}, \ \beta \times W_{max} + 3 \times \frac{1-\beta}{1+\beta} \times \frac{t}{RTT}]}{RTT} dt \tag{25}$$

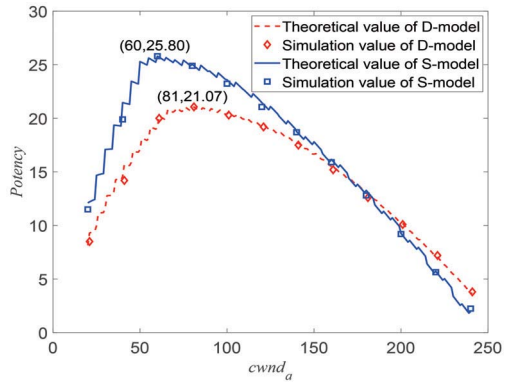

Fig. 12. Comparison of *Potecny* for D-model and S-model with respect to $cwnd_a$.

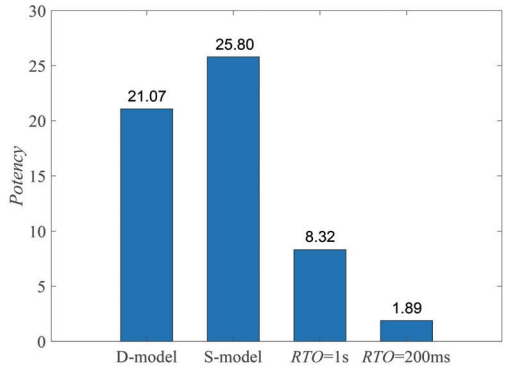

Fig. 13. Comparison between our proposed models and the traditional RTO-base attack model.

and validate the effectiveness of the proposed two models. In the following subsection, for simplicity, we will use NS-2 simulation to conduct more extensive experiments.

Next, we let $cwnd_a$ denote the window size when triggering packet loss. And then, we explore the maximum attack potency of D-model and S-model by varying $cwnd_a$ from 20 to 240.

Fig. 12 plots the curves of the attack potency obtained for D-model and S-model according (17)–(26) versus $cwnd_a$, meanwhile, presents the simulation values for comparison. The simulation values are consistent with the theoretical values. In addition, one attack bit of D-model can damage up to 21.07 TCP bits at $cwnd_a = 81$ ($Potency = 21.07$), while this metric of S-model is 25.80 obtained at $cwnd_a = 60$. Therefore, they represent an increment in the upper-bound attack potency of about 20%.

In practice, in order to determine the optimal $cwnd_a$ at which the attack model can obtain the maximum potency, the attacker can calculate this metric in theory, and then use network measurement techniques [45]–[50] to trace the practical window size. Alternatively, the attacker can induce the target TCP connection to be time out, and then trace the window behaviors from the slow start to obtain the expected $cwnd_a$.

Next, we compare our proposed models with the traditional RTO-based attack model. We implement the traditional RTO-based attack in the same experimental environment as shown in Fig. 5. We set attack parameters δ and L according to (8), and $T = RTO$. Fig. 13 reports the average potency of five

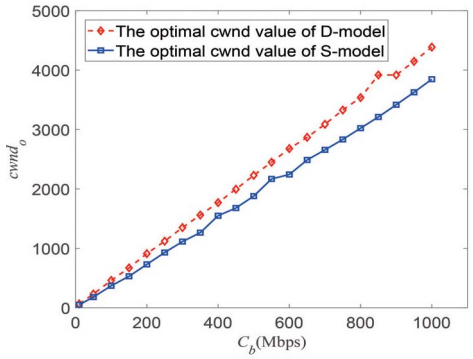

Fig. 14. The variation of the optimal window size $cwnd_o$ with C_b.

attack periods. We note that, for traditional RTO-based attack model, about 8 CUBIC TCP bits are damaged by one attack bit when RTO is 1 s (recommended value in RFC 6298 [51]), and less than 2 CUBIC TCP bits are damaged when RTO is 200 ms (actual value in modern systems [13]). Test results show that, in the worst case, our proposed attack models outperform the traditional attack model in terms of attack potency by at least 250%. Especially, this attack is no longer a low-rate attack when RTO is 200 ms, and therefore loses its concealment and is easier to detect.

C. Effect of Network Parameters on Attack Performance

In order to further validate the effectiveness of the two models and assess the effect of network parameters on their attack performance, this subsection conducts more extensive experiments in different network environments. We use the same network topology as Fig. 5, but with varying network parameters. In addition, as our study is focused towards worst-case scenarios, we define the optimal window size where the model can maximize attack potency under a given network parameter as $cwnd_o$. In the following experiments, all analysis are conducted at $cwnd_o$.

1) Effect of Bottleneck Link Capacity: this part explores the effect of C_b on the performance of D-model and S-model. In our analysis, *RTT* is fixed at 120 ms, and the bottleneck link capacity C_b ranges from 10 Mbps to 1 Gbps.

Fig. 14 presents the optimal window size $cwnd_o$ under a given C_b. We note that $cwnd_o$ increases with increasing C_b. This is because that the increment of link capacity makes the CUBIC TCP increase its congestion window size in normal case, which allows the attacker to select a larger window size at which trigger packet loss to obtain the maximum attack potency.

Fig. 15 plots the curves of the theoretical attack period T obtained for D-model and S-model according to (12), (14) and (16), respectively, versus C_b. We can observe that T increases more rapidly for D-model than it does for S-Model. When C_b exceeds 150 Mbps, T for D-model is more than twice the length of that for S-model.

Fig. 16 plots the curves of the theoretical attack cost obtained for the proposed two models according to (23) and (26), respectively, versus C_b. We observe that the cost increases with increasing C_b as the attacker needs to increase the attack rate to congest the bottleneck link. Also, we observe

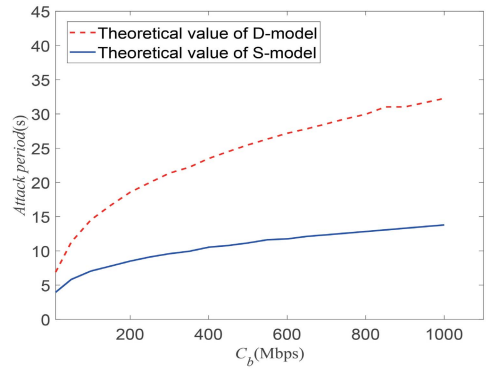

Fig. 15. Attack period as a function of C_b.

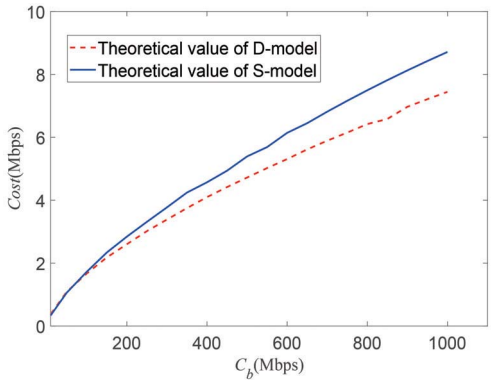

Fig. 16. Attack cost as a function of C_b.

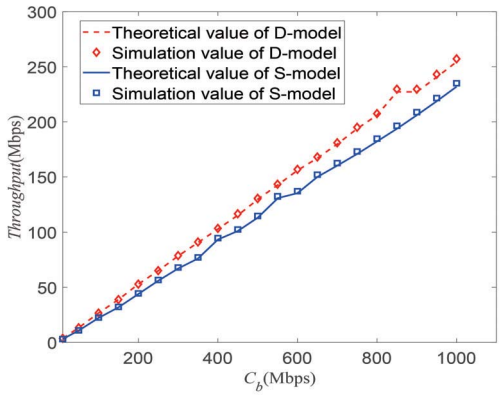

Fig. 17. Throughput as a function of C_b.

that D-model requires the cost approximately the same as S-model at relatively small C_b values, while S-model increases its cost more rapidly than D-model does. The cost of D-model is obviously lower than that of S-model when C_b is greater than 150 Mbps. This is because that D-model significantly extends its attack period (see Fig. 15), although it sends two pulses every period.

Fig. 17 plots the curves of theoretical TCP throughput obtained for D-model and S-model according to (20)–(22), (24) and (25), versus C_b, meanwhile, presents the simulation values for comparison. The simulation values are consistent with the theoretical values. We observe that the TCP throughput increases with the growth of C_b, and S-model

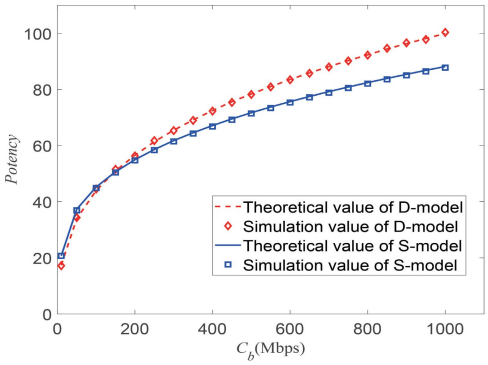

Fig. 18. Attack potency as a function of C_b.

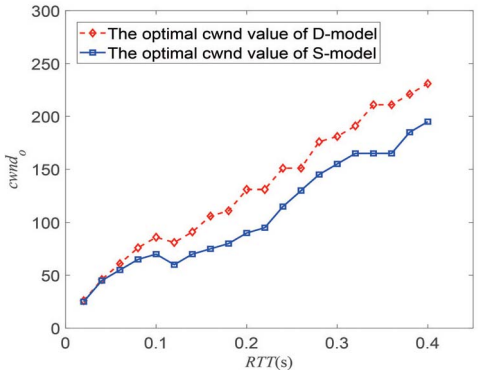

Fig. 19. The variation of the optimal window size $cwnd_o$ with *RTT*.

damages more than D-model because S-model allows the window size to be lower than D-model does.

Fig. 18 plots the curves of the maximum *Potency* for D-model and S-model according to (17)–(26), versus C_b, meanwhile, presents the simulation values for comparison. The simulation values are consistent with the theoretical values. The attack potency for both models increases with the growth of C_b over its entire given range. The maximum *Potency* of D-model is lower than that of S-model at relatively small C_b values. Whereas, D-model increases its *Potency* more rapidly than S-model does, and D-model makes greater *Potency* than S-model when C_b exceeds 150 Mbps.

The above discussions illustrate that TCP would benefit from increasing the network bandwidth to improve its transmission efficiency. However, high bandwidth contributes to the attack potency, especially for D-model, where an increment of C_b makes a more negative impact. Here, the attack effect of D-model becomes more and more prominent in high-bandwidth networks. In practice, in order to obtain the upper-bound of the attack potency, the attacker can flexibly select these two attack models according to the target network scenario.

2) Effect of Round Trip Time: this part explores the effect of *RTT* on the performance of our two models. C_b is also set as 15 Mbps, and *RTT* is varied from 20 ms to 400 ms [52], [53].

Fig. 19 presents the optimal window size $cwnd_o$ under a given *RTT*. We observe that $cwnd_o$ gradually increases with increasing *RTT*. This is because that the increment of round trip time allows the CUBIC TCP to increase its

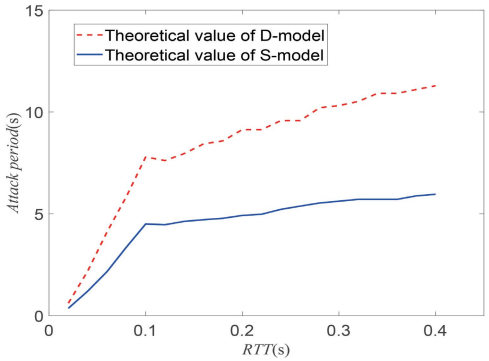

Fig. 20. Attack period as a function of *RTT*.

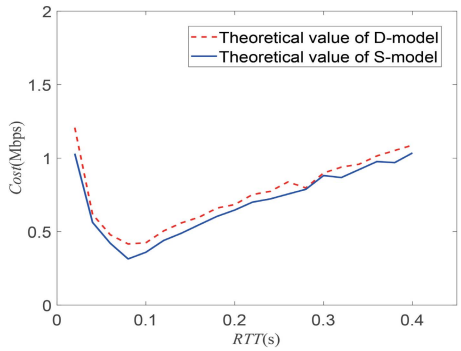

Fig. 21. Attack cost as a function of *RTT*.

congestion window size to fully utilize the network bandwidth. Consequently, the attacker can select a larger window size at which trigger the packet loss to obtain the maximum attack potency.

Fig. 20 plots the curves of the theoretical attack period T for D-model and S-model according to (12), (14) and (16), respectively, versus *RTT*. We observe that T increases with the growth of *RTT* for both models. T for D-model is never more than twice the length of that for S-model. Therefore, it can be predicted that D-model will cost more resources than S-model over the given range of *RTT*. Moreover, T sharply increases initially with the growth of *RTT*, then, when *RTT* is larger than 80 ms, T slowly increases with further growth in RTT. This behaviors can be explained as follows. CUBIC mainly works at the TCP-friendly mode under short RTTs. In this case, the window growth is dependent of RTT, and consequently, T increases with increasing $cwnd_a$ and *RTT*. Whereas, the window growth is independent of RTT under long RTTs. In this case, T increases with increasing cubic root of $cwnd_a$.

Fig. 21 plots the curves of the theoretical attack cost for D-model and S-model according to (23) and (26), versus *RTT*, meanwhile, presents the simulation results for comparison. We can observe that the simulation values are consistent with the theoretical values. As is expected, D-model required a greater cost than S-model over the entire variation range of *RTT*. Also, the cost reduces initially before the value of RTT increases to 80 ms. This behavior also results from the TCP-friendly mode. According to (13), (14) and (16), if the window

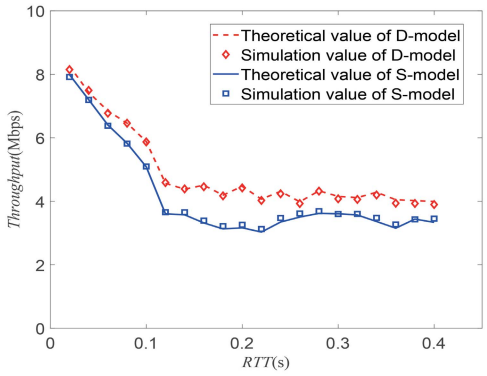

Fig. 22. Throughput as a function of *RTT*.

growth always follows the TCP-friendly mode (limit case) under short-RTTs, then *RTT* in the numerator and denominator is reduced, leaving only $cwnd_a$ in the denominator, so the cost decreases with increasing $cwnd_a$. In contrast, the window growth gradually becomes independent of RTT under long-RTTs. In this case, the value of *Cost* should increase with increasing *RTT* and decrease with increasing cubic root of $cwnd_a$. Because the RTT increment is particularly prominent, the value of *Cost* gradually increases.

Fig. 22 plots the curves of the theoretical TCP throughput for D-model and S-model according to (20)–(22), (24) and (25), versus *RTT*, meanwhile, presents the simulation results for comparison. We can observe that the simulation values are consistent with the theoretical values. In Fig. 22, the TCP throughput presents an overall downward trend with increasing *RTT*, and S-model induces greater damage than D-model because S-model allows the window size to be lower than D-model does. Furthermore, we observe from Fig. 22 that the TCP throughput presents a sharp decline at $RTT = 80$ ms. This indicates that our attack models make relatively slight damage on CUBIC TCP connections with relatively short RTTs up to 80 ms due to the TCP-friendly mode, beyond which CUBIC TCP connections with longer RTTs are obviously greater damage.

Fig. 23 plots the curves of the maximum *Potency* for D-model and S-model according to (17)–(26), versus *RTT*, meanwhile, presents the simulation values for comparison. We can observe that the simulation values are consistent with the theoretical values. Also, the maximum *Potency* of D-model is lower than that of S-model over the given range of *RTT*. The maximum attack potencies appear in the range of 80 ms to 120 ms. During this period of time, CUBIC transmissions run both the TCP-friendly mode and the cubic window growth mode.

The above analysis illustrates that the value of *RTT* has an effect on the vulnerability of CUBIC TCP to LDoS attacks. CUBIC could have improved the transmission efficiency of long-RTT flows, but larger values of *RTT* will lead to more damage. Moreover, the use of the S-model can benefit attackers in enhancing the maximum attack potency.

3) The Case of Multiple TCP Flows: this part explores the effect of the two models on multiple TCP flows. When TCP always has data available from the application layer to send,

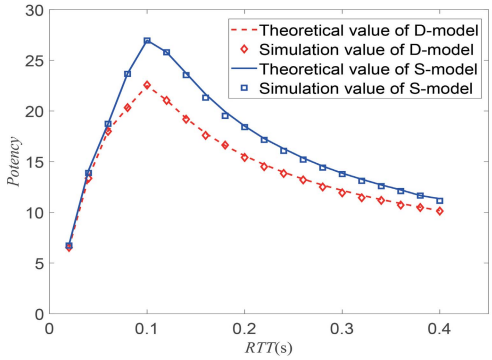

Fig. 23. Attack potency as a function of *RTT*.

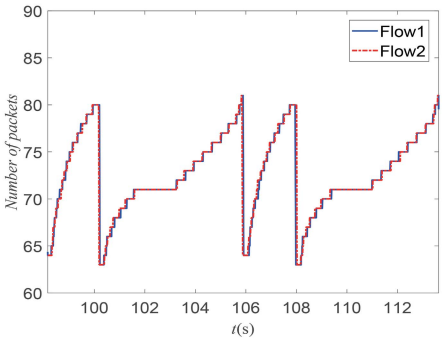

Fig. 24. Window behaviors of homogeneous CUBIC flows under D-model.

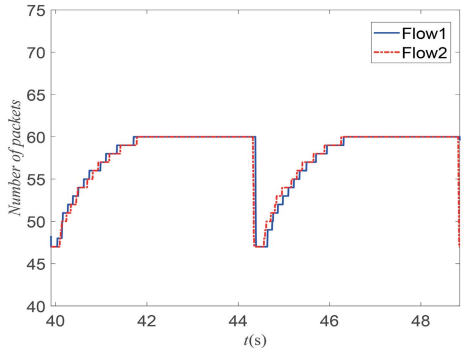

Fig. 25. Window behaviors of homogeneous CUBIC flows under S-model.

the only difference between TCP flows is their round trip times.

First, we discuss the effect of our models on two homogeneous CUBIC flows with identical $RTT = 120$ ms sharing a bottleneck link of capacity $C_b = 15$ Mbps.

As two CUBIC flows have an identical RTT, their behaviors related to congestion control will be exactly synchronous. Fig. 24 and Fig. 25 show the window behaviors of these two flows under our proposed attack models. We can obverse that CUBIC flows simultaneously enter the steady phase and the max probing phase, and increase their windows at the same time, and when packet loss occurs, they synchronously reduce their windows. The throughput on the bottleneck link decreases 61.20% under D-model attack, while this metric under S-model decreases 48.47%. Moreover, one attack bit of D-model can damage up to 11.35 TCP bits (*Potency* = 11.35), while this metric of S-model is 17.58.

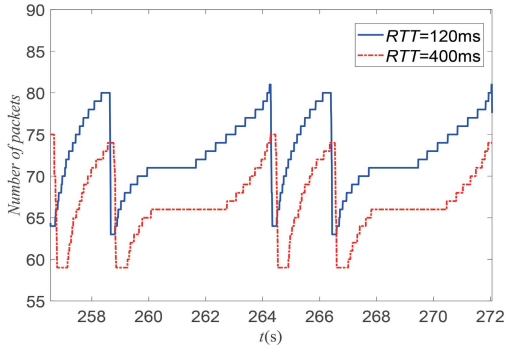

Fig. 26. Window behaviors of heterogeneous CUBIC flows under D-model.

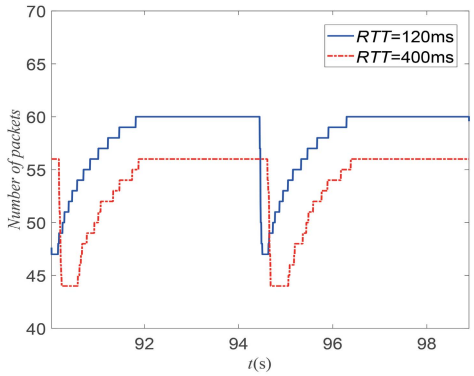

Fig. 27. Window behaviors of heterogeneous CUBIC flows under S-model.

Second, we extend our experiments for two heterogeneous CUBIC flows, where *RTTs* are set as 120 ms and 400 ms respectively, and the bottleneck link capacity C_b is set as 15 Mbps. Previous studies have indicated that the fairness of CUBIC is not enough [54]–[56]. When two CUBIC flows have different RTTs, the flow with shorter RTT will obtain higher throughput than that with longer RTT [57]. Therefore, in order to obtain a better attack effect, we select the flow with shorter RTT as the attack target.

Fig. 26 and Fig. 27 present the window behaviors of these two flows under our proposed attack models. We can obverse that the shorter-RTT based attack not only limit the window growth of the CUBIC flow with the shorter RTT, but also force that with the longer RTT to remain at a low level. The throughput on the bottleneck link decreases 38.97% under D-model attack, while this metric under S-model decreases 31.27%. And, one attack bit of D-model can damage up to 17.98 TCP bits (*Potency* = 17.98), while this metric of S-model is 23.43.

Next, we fix the RTT of the first flow as 120 ms, and vary that of the second flow from 4 ms to 400 ms [1], [10], [54], [58]. Also, we always select the flow with the shorter RTT as the target, and configure the attack parameters referring to the target flow. In Fig. 28, the horizontal axis represents RTTs of the second flow, and the vertical axis shows the maximum attack potency obtained for D-model and S-model. We can observe that, in the scenario of heterogeneous flows, if the RTT of one flow is fixed, the attack potency will increases with the growth of the RTT of another flow. This is

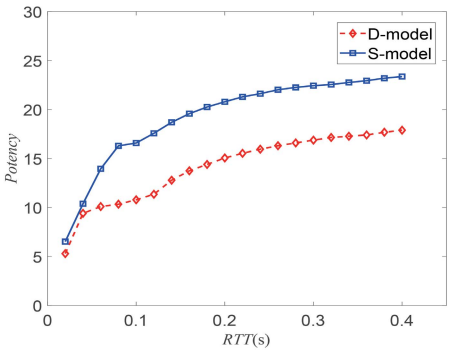

Fig. 28. The maximum attack potency as a function of the RTT of the second flow.

because the flow with the longer RTT decreases its congestion window more under the attack, which reduces the link utilization more significantly.

The above discussions illustrate that our proposed attack models can achieve excellent attack performance in the scenario of multiple CUBIC flows. Especially, for heterogeneous CUBIC flows, the attack can force these flows to lose packets at the same time and recover windows synchronously.

VI. A Brief Discussion About Attack Countermeasures

To launch successful LDoS attacks, attackers needs to probe the bottleneck link in advance. If the network can actively identify such potential target links, and then dynamically adjust them, it will make attackers harder to locate a target link. Accordingly, such attacks become blind and inefficient. Even if attackers can probe the bottleneck link again, they will take more time and resources. Therefore, this strategy can effectively mitigate LDoS attacks without being restricted by the TCP congestion control version. To implement the above mentioned idea, we are currently working on a SDN-based Moving Target Defense (MTD) approach. The SDN controller can easily identify target links that are potentially exploited by the attacker, since it has a globe view of the network. Also, based on SND's globe view, it is easy to implement an optimal MTD strategy to reroute the network traffic before attacks [59], [60], thus changing the bottleneck link and making it highly expensive to launch a successful attack. The exact details of this countermeasure is outside the scope of this paper.

VII. Conclusion

CUBIC has replaced the traditional TCP congestion control protocols and become the default algorithm used in most of current operating systems, and RED is the most typical active queue management algorithm. CUBIC combined with RED challenges traditional LDoS attacks. Inspired by this, we develop new LDoS attack models in the CUBIC + RED scenario, and accordingly explores the effects of network parameters to the attack performance. This paper first presented the normal congestion window behaviors in the absence of attacks, and then accordingly proposed two attack models. D-model was designed to alternatively limit the window growth in both the steady phase and the max probing phase of CUBIC. Then, D-model was further simplified by exploiting the rounding mechanism to form S-model. To verify the effectiveness of these two models and compare their performance, we first conduct theoretical analysis and simulation tests in a standard-configured network. Moreover, we vary different network parameters to further access the attack performance of these two models. Experimental results are consistent with theoretical analysis. In addition, test results indicate that these two models that these two models outperform the traditional attack model and alternatively maximized the attack potency under different network scenarios. As CUBIC has been widely used in today's internet, this study has practical positive application significance: 1) this study reveals the potential security vulnerability of CUBIC + RED, which will aid CUBIC users to be alert to such attack, or to select appropriate TCP congestion control algorithms for their upper-layer applications, 2) this study will aid DoS defenders to recognize the new attack manner which may be exploited by attackers. Further, this study will be of help in evaluating the extent of the attack's damage and the extent that existing countermeasures can mitigate the impact of the attack, and inspire defenders to develop effective solutions against such attack. In essence, DoS can be regarded as a game between attack and defense. The victorious balance will tend to the side that costs less but gains more. Therefore, this study finally proposed an outline for an attack countermeasure from the perspective of making attacks harder. In the future, two aspects are worth researching: exploring available models for other TCP congestion control protocol and developing effective measurements against LDoS attacks. Specifically, we are going to develop the attack model for bottleneck bandwidth and RTT (BBR) TCP, and deeply investigate defense strategies against such attack using MTD combined with SDN.

References

[1] A. Kuzmanovic and E. W. Knightly, "Low-rate TCP-targeted denial of service attacks: The shrew vs. the mice and elephants," in *Proc. ACM SIGCOMM Comput. Commun. Rev.*, Kalrushe, Germany, Aug. 2003, pp. 75–86.

[2] H. Shan, Q. Wang, and C. Pu, "Tail attack on web applications," in *Proc. ACM SIGSAC Conf. Comp. Commun. Secur.*, Dallas, TX, USA, Nov. 2017, pp. 1725–1739.

[3] Z. A. Baig, S. M. Sait, and F. Binbeshr, "Controlled access to cloud resources for mitigating economic denial of sustainability (EDoS) attacks," *Comput. Netw.*, vol. 97, pp. 31–47, Mar. 2016.

[4] A. Shawahna, M. Abu-Amara, A. Mahmoud, and Y. E. Osais, "EDoS-ADS: An enhanced mitigation technique against economic denial of sustainability (EDoS) attacks," *IEEE Trans. Cloud Comput.*, vol. 8, no. 3, pp. 790–804, Jul./Sep. 2020.

[5] T. A. Pascoal, I. E. Fonseca, and V. Nigam, "Slow denial-of-service attacks on software defined networks," *Comput. Netw.*, vol. 173, May 2020, Art. no. 107223.

[6] Y. Chen and K. Hwang, "Collaborative detection and filtering of shrew DDoS attacks using spectral analysis," *J. Parallel Distrib. Comput.*, vol. 66, no. 9, pp. 1137–1151, Sep. 2006.

[7] Y. Tang, X. Luo, Q. Hui, and R. K. C. Chang, "Modeling the vulnerability of feedback-control based internet services to low-rate DoS attacks," *IEEE Trans. Inf. Forensics Security*, vol. 9, no. 3, pp. 339–353, Mar. 2014.

[8] H. Li, J. Zhu, Q. Wang, T. Zhou, H. Qiu, and H. Li, "LAAEM: A method to enhance LDoS attack," *IEEE Commun. Lett.*, vol. 20, no. 4, pp. 708–711, Apr. 2016.

[9] P. Yang, J. Shao, W. Luo, L. Xu, J. Deogun, and Y. Lu, "TCP congestion avoidance algorithm identification," *IEEE/ACM Trans. Netw.*, vol. 22, no. 4, pp. 1311–1324, Aug. 2014.

[10] S. Ha, I. Rhee, and L. Xu, "CUBIC: A new TCP-friendly high-speed TCP variant," *ACM SIGOPS Oper. Syst. Rev.*, vol. 42, no. 5, pp. 64–74, 2008.

[11] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. Netw.*, vol. 1, no. 4, pp. 397–413, Aug. 1993.

[12] A. M. Abdelmoniem, B. Bensaou, and A. J. Abu, "Mitigating incast-TCP congestion in data centers with SDN," *Ann. Telecommun.*, vol. 73, nos. 3–4, pp. 263–277, Apr. 2018.

[13] W. Rebecca and C. A. Hope. (Jun. 2017). *CS244 '17: Low-Rate TCP-Targeted Denial of Service Attacks*. [Online]. Available: https://reproducingnetworkresearch.wordpress.com/2017/06/05/cs244-17-low-rate-tcp-targeted-denial-of-service-attacks/

[14] W. Zhijun, L. Wenjing, L. Liang, and Y. Meng, "Low-rate DoS attacks, detection, defense, and challenges: A survey," *IEEE Access*, vol. 8, pp. 43920–43943, 2020.

[15] M. Yue, M. Wang, and Z. Wu, "Low-high burst: A double potency varying-RTT based full-buffer shrew attack model," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 5, pp. 2285–2300, Sep./Oct. 2019.

[16] Z. Wu, L. Zhang, and M. Yue, "Low-rate DoS attacks detection based on network multifractal," *IEEE Trans. Dependable Secure Comput.*, vol. 13, no. 5, pp. 559–567, Sep. 2016.

[17] Z. Chen, C. K. Yeo, B. S. Lee, and C. T. Lau, "Power spectrum entropy based detection and mitigation of low-rate DoS attacks," *Comput. Netw.*, vol. 136, pp. 80–94, May 2018.

[18] A. Dahiya and B. B. Gupta, "A reputation score policy and Bayesian game theory based incentivized mechanism for DDoS attacks mitigation and cyber defense," *Future Gener. Comput. Syst.*, vol. 117, pp. 193–204, Apr. 2021.

[19] A. Praseed and P. S. Thilagam, "Modelling behavioural dynamics for asymmetric application layer DDoS detection," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 617–626, 2021.

[20] A. Dahiya and B. B. Gupta, "Multi attribute auction based incentivized solution against DDoS attacks," *Comput. Secur.*, vol. 92, May 2020, Art. no. 101763.

[21] A. Tewari and B. B. Gupta, "Security, privacy and trust of different layers in Internet-of-Things (IoTs) framework," *Future Gener. Comput. Syst.*, vol. 108, pp. 909–920, Jul. 2020.

[22] G. Somani, M. S. Gaur, D. Sanghi, M. Conti, and M. Rajarajan, "Scale inside-out: Rapid mitigation of cloud DDoS attacks," *IEEE Trans. Dependable Secure Comput.*, vol. 15, no. 6, pp. 959–973, Nov. 2018.

[23] R. Xie, M. Xu, J. Cao, and Q. Li, "SoftGuard: Defend against the low-rate TCP attack in SDN," in *Proc. ICC*, Shanghai, China, May 2019, pp. 1–6.

[24] A. Mishra, N. Gupta, and B. B. Gupta, "Defense mechanisms against DDoS attack based on entropy in SDN-cloud using POX controller," *Telecommun. Syst.*, vol. 77, no. 1, pp. 47–62, Jan. 2021.

[25] K. Bhushan and B. B. Gupta, "Distributed denial of service (DDoS) attack mitigation in software defined network (SDN)-based cloud computing environment," *J. Ambient Intell. Hum. Comput.*, vol. 10, no. 5, pp. 1985–1997, May 2019.

[26] S. Sarat and A. Terzis, "On the effect of router buffer sizes on low-rate denial of service attacks," in *Proc. 14th Int. Conf. Comput. Commun. Netw.*, San Diego, CA, USA, Oct. 2005, pp. 281–286.

[27] M. Guirguis, A. Bestavros, and I. Matta, "On the impact of low-rate attacks," in *Proc. IEEE ICC*, Istanbul, Turkey, Jun. 2006, pp. 2316–2321.

[28] M. Yue, Z. Wu, and M. Wang, "A new exploration of FB-shrew attack," *IEEE Commun. Lett.*, vol. 20, no. 10, pp. 1987–1990, Oct. 2016.

[29] Y. Zhang, Z. M. Mao, and J. Wang, "Low-rate TCP-targeted DoS attack disrupts internet routing," in *Proc. Netw. Distrib. Syst. Secur. Symp. (NDSS)*, Feb. 2007, pp. 1–15.

[30] J. Luo, X. Yang, J. Wang, J. Xu, J. Sun, and K. Long, "On a mathematical model for low-rate shrew DDoS," *IEEE Trans. Inf. Forensics Security*, vol. 9, no. 7, pp. 1069–1083, Jul. 2014.

[31] M. Ficco and M. Rak, "Stealthy denial of service strategy in cloud computing," *IEEE Trans. Cloud Comput.*, vol. 3, no. 1, pp. 80–94, Jan./Mar. 2015.

[32] J. Cao *et al.*, "The crosspath attack: Disrupting the SDN control channel via shared links," in *Proc. 28th USENIX Secur. Symp.*, Aug. 2019, pp. 19–36.

[33] K. Kannan and S. Banerjee, "Compact TCAM: Flow entry compaction in TCAM for power aware SDN," in *Proc. Int. Conf. Distrib. Comput. Netw.* Berlin, Germany: Springer, Jan. 2013, pp. 439–444.

[34] M. Polese, F. Chiariotti, E. Bonetto, F. Rigotto, A. Zanella, and M. Zorzi, "A survey on recent advances in transport layer protocols," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 4, pp. 3584–3608, 4th Quart., 2019.

[35] T. Henderson, S. Floyd, A. Gurtov, and Y. Nishida, *The NewReno Modification to TCP's Fast Recovery Algorithm*, document RFC 6582, Internet Engineering Task Force, Apr. 2012.

[36] I. Rhee, L. Xu, S. Ha, A. Zimmermann, L. Eggert, and R. Scheffenegger, *CUBIC for Fast Long-Distance Networks*, document RFC 8312, Internet Engineering Task Force, Feb. 2018.

[37] G. Fairhurst and F. Baker, *IETF Recommendations Regarding Active Queue Management*, document RFC 7567, Jul. 2015.

[38] N. Kuhn, P. Natarajan, N. Khademi, and D. Ros, *Characterization Guidelines for Active Queue Management (AQM)*, document RFC 7928, Jul. 2016.

[39] S. Liu, T. Basar, and R. Srikant, "Exponential-RED: A stabilizing AQM scheme for low and high-speed TCP protocols," *IEEE/ACM Trans. Netw.*, vol. 13, no. 5, pp. 1068–1081, Oct. 2005.

[40] C. Wang, J. Liu, B. Li, K. Sohraby, and Y. T. Hou, "LRED: A robust and responsive AQM algorithm using packet loss ratio measurement," *IEEE Trans. Parallel Distrib. Syst.*, vol. 18, no. 1, pp. 29–43, Jan. 2007.

[41] M. Cheng, H. Wang, and L. Yan, "Dynamic RED: A modified random early detection," *J. Comput. Inf. Syst.*, vol. 7, no. 14, pp. 5243–5250, Dec. 2011.

[42] L. Brett, C. Mark, and K. Robert, "A TCP CUBIC implementation in NS-3," in *Proc. WNS3*, Atlanta, GA, USA, May 2014, pp. 1–8.

[43] G. Appenzeller, I. Keslassy, and N. McKeown, "Sizing router buffers," in *Proc. ACM SIGCOMM Comput. Commun. Rev.*, Portland, OR, USA, Aug. 2004, pp. 281–292.

[44] C. Zhang, J. Yin, Z. Cai, and W. Chen, "RRED: Robust RED algorithm to counter low-rate denial-of-service attacks," *IEEE Commun. Lett.*, vol. 14, no. 5, pp. 489–491, May 2010.

[45] C. Dovrolis, P. Ramanathan, and D. Moore, "What do packet dispersion techniques measure?" in *Proc. 20th Annu. Joint Conf. Comput. Commun. Soc.*, Anchorage, Alaska, Apr. 2001, pp. 905–914.

[46] M. Jain and C. Dovrolis, "End-to-end available bandwidth: Measurement methodology, dynamics, and relation with TCP throughput," *IEEE/ACM Trans. Netw.*, vol. 11, no. 4, pp. 537–549, Aug. 2003.

[47] J. Liu and M. Crovella, "Using loss pairs to discover network properties," in *Proc. IEEE/ACM SIGCOMM Int. Meas. Workshop*, San Francisco, CA, USA, Nov. 2001, pp. 127–138.

[48] O. Gurewitz, I. Cidon, and M. Sidi, "One-way delay estimation using network-wide measurements," *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2710–2724, Jun. 2006.

[49] A. Vakili and J.-C. Gregoire, "Accurate one-way delay estimation: Limitations and improvements," *IEEE Trans. Instrum. Meas.*, vol. 61, no. 9, pp. 2428–2435, Sep. 2012.

[50] L. Xue, X. Ma, X. Luo, E. W. W. Chan, T. T. Miu, and G. Gu, "LinkScope: Toward detecting target link flooding attacks," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 10, pp. 2423–2438, Oct. 2018.

[51] V. Paxson, M. Allman, J. Chu, and M. Sargent, *Computing TCP's Retransmission Timer*, document RFC 6298, Jun. 2011.

[52] H. Jiang and C. Dovrolis, "Passive estimation of TCP round-trip times," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 32, no. 3, pp. 75–88, Jul. 2002.

[53] S. Floyd and E. Kohler, "Internet research needs better models," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 1, pp. 29–34, Jan. 2003.

[54] T. Kozu, Y. Akiyama, and S. Yamaguchi, "Improving RTT fairness on CUBIC TCP," in *Proc. 1th CANDAR*, Matsuyama, Japan, Dec. 2013, pp. 162–167.

[55] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "BBR: Congestion-based congestion control," *Commun. ACM*, vol. 60, no. 2, pp. 58–66, 2017.

[56] S. Lee, D. Lee, M. Lee, H. Jung, and B.-S. Lee, "Randomizing TCP payload size for TCP fairness in data center networks," *Comput. Netw.*, vol. 129, pp. 79–92, Dec. 2017.

[57] G. Vardoyan, C. V. Hollot, and D. Towsley, "Towards stability analysis of data transport mechanisms: A fluid model and an application," in *Proc. IEEE INFOCOM- IEEE Conf. Comput. Commun.*, Honolulu, HI, USA, Apr. 2018, pp. 666–674.

[58] H. Young and H. Bradley. (Oct. 2020). *Round-Trip Time Internet Measurements from CAIDA's Macroscopic Internet Topology Monitor*. [Online]. Available: https://www.caida.org/research/performance/rtt/walrus0202/

[59] S. Sengupta, A. Chowdhary, A. Sabur, A. Alshamrani, D. Huang, and S. Kambhampati, "A survey of moving target defenses for network security," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 3, pp. 1909–1941, 3rd Quart., 2020.

[60] J.-H. Cho *et al.*, "Toward proactive, adaptive defense: A survey on moving target defense," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 1, pp. 709–745, 1st Quart., 2020.

Meng Yue received the Ph.D. degree in information and communication engineering from Tianjin University, China, in 2017. He is currently an Associate Professor with the Department of Cybersecurity, Civil Aviation University of China. His current research interests include network security and cloud computing.

Jing Li received the B.S. degree in communication engineering from the Hebei University of Science and Technology, China, in 2019. She is currently pursuing the master's degree in communication and information system with the Civil Aviation University of China. Her current research interests include network security and information security.

Zhijun Wu received the Ph.D. degree in cryptography from the Beijing University of Posts and Telecommunications, China, in 2004. He is currently a Professor with the Department of Cyberspace Security, Civil Aviation University of China. He is also the Supervisor of Ph.D. candidates at Tianjin University, China, and the Civil Aviation University of China. His current research interests include network security and cloud computing.

Minxiao Wang received the B.S. degree in communication engineering and the M.S. degree in information and communication engineering from the Civil Aviation University of China in 2014 and 2018, respectively. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, Southern Illinois University, USA. His research interests include network security, NDN, and autonomous and connected vehicles.