JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 38, 1265-1283 (2022)
DOI: 10.6688/JISE.202211_38(6).0009

A Study on Traffic Asymmetry for Detecting DDoS Attack in P4-based SDN*

TING-YU LIN[1], CHING-YUAN WANG[1], YA-PEI TUAN[1],
MENG-HSUN TSAI[1,3] AND YEAN-RU CHEN[2,+]
[1]*Department of Computer Science and Information Engineering*
[2]*Department of Electrical Engineering*
National Cheng Kung University
Tainan, 700 Taiwan
[3]*Department of Computer Science*
National Yang Ming Chiao Tung University
Hsinchu, 300 Taiwan
E-mail: tingyu@imslab.org; chingyuan0227@gmail.com; kinoe_T@imslab.org;
tsaimh@csie.ncku.edu.tw; chenyr@mail.ncku.edu.tw

With the popularity of the Internet, modern people increasingly rely on the Internet to complete a large amount of work, making the security of the Internet more and more important. Among many threats to network security, Distributed Denial-of-Service (DDoS) attacks have always been a problem that researchers want to solve. With the introduction of software-defined networking (SDN), more and more detection methods have been proposed. In this paper, we design a sketch-based method of data collection in the P4-based data plane, which sends less data to controller than the Openflow-based data plane with limited data size. Furthermore, our method collects data of both attackers and victims by asymmetric characteristics of data flows, which contributes to the mitigation of DDoS attacks by inserting rate-limited rules on the data plane. In experiments, our data collection structure can reach the 0.9 or more F1 score, and the number of entries is appropriate, while attack intensities are between 0Mbps to 500Mbps. In the evaluation section, we also present the result of labeling data by the K-means algorithm on the control plane.

Keywords: software defined network, data collection, DDoS, P4, network management

1. INTRODUCTION

With the popularity of the Internet today, more and more services, such as financial activities, public transformation, and social media, can or must be provided through the Internet. People increasingly rely on the speed and convenience of the Internet. Moreover, with the evolution of network technology, more and more tasks rely on the Internet. In this case, network security has become an important issue nowadays. If there is no proper defense mechanism, malicious people can easily crash the system, invade, obtain personal information, *etc*.

Received October 31, 2021; revised December 20, 2021; accepted January 13, 2022.
Communicated by Shin-Jie Lee.
+ Corresponding author.
* This work was supported in part by the MOST under Grant 109-2221-E-006-160-, 110-2221-E-006-016- and 110-2221-E-006-212-.

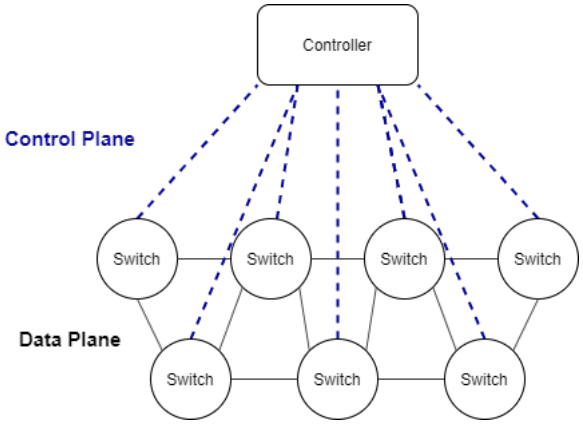


Fig. 1. Software-defined networking (SDN).

Among the threats to network security, Distributed Denial-of-Service (DDoS) attacks have always been a big problem [1,2]. Although people have found ways to prevent users from being attacked, the incidence of DDoS attacks is still increasing. By using multiple computers to send numerous packets to one or more targets, services may be interrupted or even suspended. Furthermore, DDoS attacks may cause economic losses because they may cause bank servers to stop working. Therefore, to detect DDoS attacks quickly and accurately, it is important to have a good method of measuring network traffic.

Since the introduction of Software Defined Network (SDN), control centers was separated from switches as shown in Fig. 1 to measure the network. By applying SDN, network administrators can use the controller to easily monitor and issue instructions to each switch, which makes methods to measure the network sprung up [3, 4]. They can also manage the logical architecture of the network by changing the flow table of each switch without moving the hardware. Unlike traditional networks, the characteristics of SDN allow researchers more flexibility to develop new methods to measure networks or detect DDoS attacks.

In SDN technology, Programming Protocol-Independent Packet Processor (P4) allows network administrators to define how the data plane works in their own way [5] as shown in Fig. 2. With the flexibility of the data plane, researchers can choose what or when to calculate in the switch. In addition, you can define your own data structure or perform some calculations on the data plane, which also reduces the limitations of developing DDoS attack detection methods. Therefore, to design an efficient and highly customized detection mechanism, it is a great idea to use P4 as a programming language.

There are some studies talking about the relationship between DDoS attacks and defense mechanisms [6–8]. As far as we know, the previously proposed methods have tried many different techniques, such as machine learning, neural network, and so on. However, these mechanisms often try to collect data for each IP address, which can lead to a large amount of memory usage. In addition, the bandwidth consumed by the data conversion from the switch to the controller is also an issue. Another way is to use sketch-based methods to count packets. By setting an appropriate hash function and number of

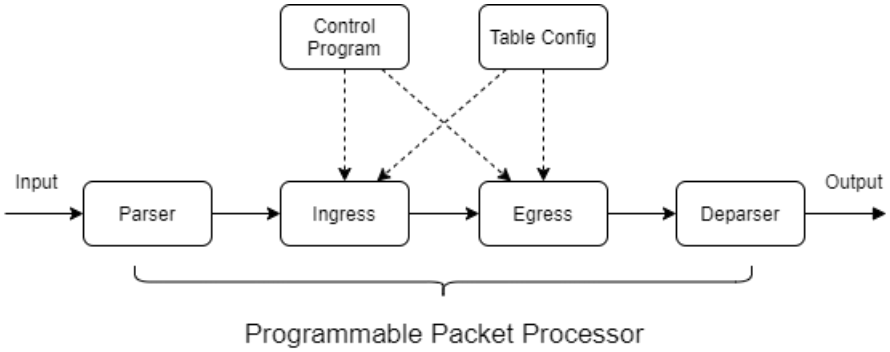


Fig. 2. Protocol-independent packet processor (P4).

entries, memory usage can be reduced to an acceptable level.

Among existing sketch-based solution, ElasticSketch [9] uses a two-phase sketch to collect DDoS flow and separate elephant flows from mouse flow and implements their method on six platforms that includes P4 and OVS. However, ElasticSketch lacks accuracy guarantees while recording too many flows and only records the information of attackers.

In this paper, we design a sketch with a new data structure that can catch anomaly flows with characteristics of traffic volume and asymmetries (the difference of flow that an IP has sent and received). In our simulation, the sketch maintains the F1 score upper than 0.9 while attack intensities are between 0Mbps to 500Mbps. We also present the result of labeling data by the K-means algorithm on the control plane.

The remainder of the paper is organized as follows. In the next section, we describe related detection methods in statistic-based, machine0based, and sketch-based. In Section 3, we present our sketch-based method, which includes a data structure, and how to insert and eviction in this structure. In Section 4, we tune the parameter and test the validity of our model. In the last section, we summarize this paper.

2. RELATED WORKS

This section describes related works in three aspects: statistic based, machine learning based and sketch based.

2.1 Statistic-based Method

By finding the probability of an event, one can infer whether the event is happening. Several detection models are proposed based on entropy [10–12]. Entropy essentially describes the “uncertainty” of random variables. Therefore, since DDoS attacks always aim at one or several targets, calculating the entropy of the destination is a good way to measure the dispersion of packets received by each IP address.

In [10], the authors deploy their model on edge switches. The model first collects packets that go through each switch, then calculates the entropy of the destination IP address. If the entropy is below a threshold, then the switch alarms the controller for a

DDoS attack. In [11], the model calculates the entropy of the destination IP address on the controller, but it requires switches to forward all the packets to the controller and causes heavy bandwidth usage. For the model proposed in [12], the entropy should stay under the threshold for a fixed period to alarm for DDoS attacks. Researches above use entropy as an important feature with different ways of defining thresholds. Despite the sensitivity, we can not find which IP address is actually sending malicious packets with the entropy of the destination IP address. Moreover, further analysis is also needed to find out which IP address is attacked. Therefore, it is quite not enough to use entropy alone for detection.

In [13], the authors proposed a cross-plane detection model with two-phase analysis. For coarse-grained detection, they use traffic volume and asymmetry to detect potential DDoS attacks. However, they set the threshold with Pauta criterion, which is based on an assumption that these variables follow the normal distribution. Nevertheless, there is no evidence that these variables follow the distribution. In addition, the article does not mention accurately which fine-grained analysis should the control plane apply.

Statistic-based methods do not need too much calculating resource. However, most methods are based on assumptions such as distribution, which may vary due to the network environment. On the other hand, with the advancement of machine learning techniques, some other researchers try to apply them to their DDoS detection model.

2.2 Machine-Learning-Based Method

For another category, researchers use machine learning for detection model [14–17]. Among many machine learning techniques, support vector machine (SVM) is a popular method due to its small classification error. In [14], the authors use sFlow to collect data, sample packets with a fixed ratio and send the sampled packets to the controller. For feature selection, decision trees are used to eliminate less important features. Finally, an SVM classifier will be used to detect whether there is an attack. In [15], the model only let suspicious traffic forward packets to the controller and the authors tried a few methods such as logistic regression and SVM. In [16], the authors proposed a new way to collect information. That is, sending traffic data along with Packet_In to prevent overloading the bandwidth. They claim to use their method and sampling method according to the network scale. In addition, the model calculates the entropy of source IP address, destination IP address, source port, and destination port, and puts them into an SVM classification model. The model proposed in [17] maintains a blacklist for preliminary filtering. Packets whose source IP address is not on the blacklist will be further processed and classified. The authors tried several classification methods, namely SVM, Naive Bayes, K-Nearest, and multi-layer perception, which SVM performs the best in most experiments.

Overall, machine-learning-based methods usually have nice accuracy, especially SVM. However, the computing complexity often causes the algorithms to spend more time than others. Furthermore, using complex mechanisms causes the algorithms unavailable to run on switches, meaning all data should be forwarded to the controller, which causes serious bandwidth overload.

2.3 Sketch-Based Method

Sketch-based data structures have been used to find frequent items, such as count sketch [18, 19], or count-min sketch [20, 21]. Sketch-based methods do not always as-

sign a single counter for each item, instead, they use hash functions to make each entry available to record information of multiple items. After recording, they approximate an item's features such as frequency by extracting information from multiple entries that are related to the specific item. Given the actual size of sketches, some sketch-based methods have restricted error bounds and can be shown with numerical analysis. Recently, new kinds of sketches are proposed to gather information from the network [9, 22]. Benefiting from the few sizes needed and the ability to record information, researchers apply sketch-based data structure to collect traffic data to run the detection model subsequently. For instance, in [23], the authors develop a mechanism using count sketch to approximate IP address frequency. With the programmability of P4, they can easily define several arrays on switches for sketching. Additionally, they estimate the entropy of source and destination IP address with the approximated frequency while reducing the computing complexity by using the Ternary Content Addressable Memory (TCAM) table, which is pre-computed so that the model only needs to look up instead of calculating. Finally, the estimated entropy is then used for characterizing network traffic and subsequent detection.

In [22], the authors aim to collect information of heavy hitter in the network and start from improving Space-Saving algorithm [24]. The mechanism is also implemented on P4 for self-defined switch behavior. By applying hash functions, the number of memory access drops significantly while high accuracy remained. In addition, a multi-table technique is included for decreasing collision rate. In [9], the authors introduce another mechanism called Ostracism in their data structure. An IP address can occupy a bucket, but also can be expelled by other IP addresses if they have sent enough packets more than the occupying one. After a phase of traffic, the one that stays in the bucket should more likely be the heavy hitter, since IP addresses that send a few packets should be expelled by it. We think the mechanism is very useful to filter the heavy hitters so that we'll use it as a part of our method. Also, the sketch is reversible so that heavy hitters' id is already shown without further queries.

In general, sketch-based network measuring methods often spend less memory space than others. To develop an efficient measuring method beneficial to a DDoS attack, we refer to the researches mentioned above. In spite of the accuracy of finding heavy hitters in [9, 22], both piece of research did not consider asymmetry of packets. According to previous research [25], using asymmetry features can improve the effect of predicting a DDoS attack, compared to only using volume features. Therefore, we extend the sketch-based data structure reference from Elastic Sketch [9] to record more information. In addition, we design the formula for evaluating traffic volume and asymmetry used in our model, in order to improve the data collection process. We use an F1 score as the metric for measuring effectiveness, expecting our method can preserve information about a DDoS attackers/victims more accurately and have a higher F1 score.

3. PROPOSED SCHEME

In this section, we first introduce our data structure in Subsection 3.1. After that, we describe how the counting is done when packets arrive at a switch in Subsection 3.2. Finally, we introduce eviction and the thresholds we use in Subsection 3.3.

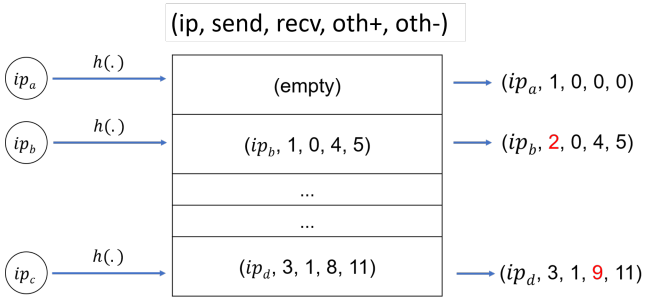


Fig. 3. Operations for source IP address.

3.1 Data Structure

In this subsection, we introduce the data structure of our model. We plan to make the data structure work on SDN switches and use P4 as our programming language. The data structure records the information when switches forward data packets. The core idea of our model is Ostracism and two important features of DDoS detection: *volume* and *asymmetry*. We show how these idea is realized in our model in the next subsection. As shown in Figs. 3 and 4, the data structure is essentially an extended hash table. The table is associated with a hash function $h(.)$ for hashing the IP addresses of incoming packets. Each table entry consists of several fields, which records the IP address (ip), packets sent by the IP address (send), packets received by the IP address (recv), packets sent by other IP addresses (oth+), and packets received by other IP addresses (oth-).

3.2 Insertion

For each packet going through a switch, our model first hashes the source IP address and records the information in the data structure. Given the source IP address ip_{src}, and assume that it is hashed to an entry $E = (ip_a, send, recv, oth^+, oth^-)$. According to ip_{src} and ip_a, there are three cases and examples are shown above in Fig. 3:

- Case 1: If E is empty, then we insert a record $(ip_{src}, 1, 0, 0, 0)$ to the entry.
- Case 2: If E is not empty and $ip_{src} = ip_a$, then we increment the field *send*, meaning that ip_a has sent one more packet.
- Case 3: If E is not empty and $ip_{src} \neq ip_a$, then we increment the field oth^+, meaning that there is a packet sent by an IP address different from ip_a.

After the operations are done with the source IP address, the model will hash the destination IP address of the packet, given ip_{dst}, and do similar operations. However, since we are recording the information about IP addresses receiving packets in this step, the fields modified will be different from above. Assume that ip_{dst} is hashed to another bucket $E' = (ip_b, send, recv, oth^+, oth^-)$. According to ip_{dst} and ip_b, there are also three cases, examples are shown above in Fig. 4:

- Case 1: If E' is empty, then we insert a record $(ip_{dst}, 0, 1, 0, 0)$ to the entry.
- Case 2: If E' is not empty and $ip_{dst} = ip_b$, then we increment the field *recv*, meaning that ip_b has received one more packet.

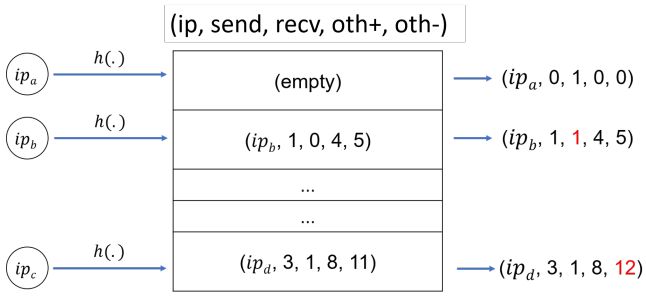


Fig. 4. Operations for Destination IP Address

- Case 3: If E' is not empty and $ip_{dst} \neq ip_b$, then we increment the field oth^-, meaning that there is a packet received by an IP address different from ip_b.

With the operations described above, we can record the packet volume and asymmetry not only of a single IP address but also of other IP addresses hashed to the same entry. The model will decide whether an eviction should be done with the information. In the next subsection, we'll introduce the standard for judging it, and how evictions are done.

3.3 Eviction

We have already shown how we record information from the network. Now we describe how the model preserves the IP addresses we want among numerous users in the network. Since we are using a sketch-based model and several IP addresses will be hashed to the same entry, we want the data structure to save the IP addresses which have the highest asymmetry of sending/receiving packets. We will call them **asymmetric IP addresses** in the rest of the paper for convenience. Since each entry is empty after initialization, any IP address hashed to the entry can occupy the fields, including those that might send/receive only a few packets. Therefore, eviction is needed as a mechanism for the asymmetric IP addresses to replace other IP addresses.

Given an IP address of incoming packet ip_i, and it is hashed to an entry $E = (ip, send, recv, oth^+, oth^-)$. The model will decide if an eviction is needed when E is not empty and $ip_i \neq ip$, that is, case 3 when introducing operations in the previous subsection. Whether ip_i is a source or destination IP address, thresholds for the decision are the same. Now we introduce the thresholds we use for decision.

First, for the entry E, we count $T = send + recv + oth^+ + oth^-$. If $T < T_{thr}$, where T_{thr} is a predefined threshold, then eviction will not happen. This threshold ensures that each IP address will have enough time to show its trend without being replaced immediately after occupying an entry. In addition, the threshold makes sure that eviction won't happen too often since it involves some memory access and may cause burdens on switches. On the other hand, if $T \geq T_{thr}$, the model will go on and calculate another two thresholds, α and β.

α and β are the thresholds that filter the features of IP addresses for the model.

Among them, α can be expressed as

$$\alpha = \left| \frac{send - recv}{oth^+ - oth^-} \right|. \tag{1}$$

It calculates the difference of packets ip sent/received, divided by the difference of packets sent/received by other IP addresses, and uses its absolute value as a reference for the asymmetry of ip. If α is smaller than a predefined threshold, *i.e.* $\alpha < \alpha_{thr}$, it means that other IP addresses hashed to E have more asymmetric packet sending/receiving than ip. For example, when the first IP hashed in the table is benign, an attacker arouses a DoS attack by another IP. Cause the difference of $other^+$ and $other^-$ is becoming bigger and making α smaller, which means that the asymmetric of the not recorded IP in a hashed table is more significant than the recorded IP. In this case, eviction will be executed.

β is another threshold that can be expressed as

$$\beta = \frac{send + recv}{oth^+ + oth^-}. \tag{2}$$

It calculates total packets associated with ip and is divided by total packets of other IP addresses. This represents the volume of the current IP address relative to others, and there is also a correspondingly predefined threshold β_{thr}. If $\beta < \beta_{thr}$, meaning other IP addresses cause more traffic than ip does. For example, the first IP hashed in the table only transit a few packets then stop the connection, causing the sum of $other^+$ and $other^-$ is becoming bigger while the sum of "send" and "recv" is a fixed value. In this case, β will become smaller, which means that the volume of the not recorded IP in a hashed table is more significant than the recorded IP. In this case, eviction will be executed, too.

To summarise the overall algorithm of our method, we show a macroscopic flow chart below in Fig. 5. When a packet passes through the P4 switch, we update the same hashed table by the source IP (ip_{src}) and the destination IP (ip_{dst}), respectively, of the packet. Depending on the IP, there are the four updating cases shown in the sub-flowchart including

1. being the first IP hashed into the row
2. being the IP recorded in the row
3. being the IP not recorded and going to evict the origin IP, and
4. being the IP not recorded but not going to evict the recorded IP.

The corresponding method of each case marks on the upper left of the sub-flowchart in Fig. 5. Notably, the updating methods of the destination IP and source IP are the same.

Here we show an example to describe how an eviction is done in Fig. 6. Assume we set the thresholds $T_{thr} = 10$, $\alpha_{thr} = 0.6$, and $\beta_{thr} = 0.6$. A packet is going through a switch and the model hashes the source IP address ip_i to an entry $E = (ip, 3, 1, 4, 2)$. Then according to the operations introduced previously, the model increments the oth^+ field, making it $E = (ip, 3, 1, 5, 2)$, and check if an eviction is needed. It first calculates T, makes sure that $T \geq T_{thr}$. Then it checks α and β, which calculations are shown in the figure. By following the flowchart, the model infers that eviction is needed.

It is easy to execute an eviction. First, the new incoming IP address (ip_i) replaces the original one (ip). Then, since ip_i is a source IP address, the model fills 1 in the *send*

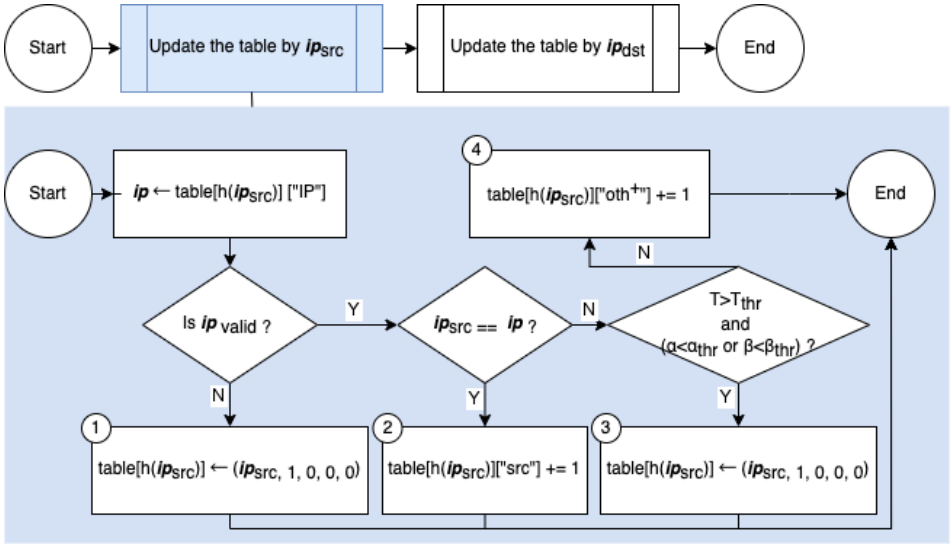
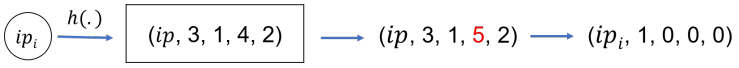


Fig. 5. Overall flowchart of our method.



$$T = 3 + 1 + 5 + 2 = 11 \geq T_{thr} = 10$$

$$\alpha = \left|\frac{3-1}{5-2}\right| \approx 0.67 \geq \alpha_{thr} = 0.6$$

$$\beta = \frac{3+1}{5+2} \approx 0.57 < \beta_{thr} = 0.6$$

Fig. 6. Eviction example.

field and lets the other fields set to 0. Note that if ip is a destination IP address, the model will set the *recv* field to 1 and others 0. The replacement means that the model stops considering ip as an asymmetric IP address, and make ip_i its new candidate Finally, after the eviction is done, the entry $E = (ip, 3, 1, 5, 2)$ becomes $E = (ip_i, 1, 0, 0, 0)$.

By recording the packets with the data structure and the evicting mechanism, the model tries to preserve information of asymmetric IP addresses. In the next section, we'll discuss evaluations of the model, including the optimization of thresholds, accuracy of detecting asymmetric IP addresses, and so on.

4. EVALUATION

In this section, we demonstrate the simulations of our model. Since the model needs three parameters, we first describe the tuning process of the parameters. After that, we test the validity of our model, *i.e.*, the ability to preserve those we called asymmetric

IP addresses. Additionally, we extend our simulation to a DDoS attack scenario to test whether the model effectively preserves IP addresses of DDoS attackers and victims. Last, we test the effect of whether using our method to preserve data for the controller when it comes to machine-learning detecting methods.

4.1 Model Tuning

4.1.1 Simulation parameters

As mentioned above, each of the thresholds is in charge of some part of the filtering. Therefore, it is important to know which combination of them performs the best. We simulate with two kinds of traces, which split the simulation into two parts, to find the best combination.

The first part of the simulation uses traces composed of randomly produced records of packets following predefined distributions and parameters, simulating a DDoS attack scenario. We use Pareto distribution based on previous research [26] and the observation of the dataset we found [27]. The density formula of Pareto distribution:

$$f(t) = \frac{\alpha\beta^{\alpha}}{(t+\beta)^{\alpha+1}} \tag{3}$$

and the way to match distribution parameters with datasets are introduced in [28]. The total number of IP addresses is 4000, while the number of packets each address sends and the inter-sending time of packets follow Pareto distribution with the mean around 100 packets and 200 ms. Moreover, there are three attackers and one victim in each trial. Each attacker sends additional packets to the victim, which the packet inter-sending time follows Pareto distribution with mean = 3 ms. We assume the size of normal packets is 1000 bytes and the malicious ones 1500 bytes. Also, the bandwidth of all IP addresses is 5 Mbps while the bandwidth of the victim is 10 Mbps. The parameters of the traces are listed below in Table 1 for ease of reading.

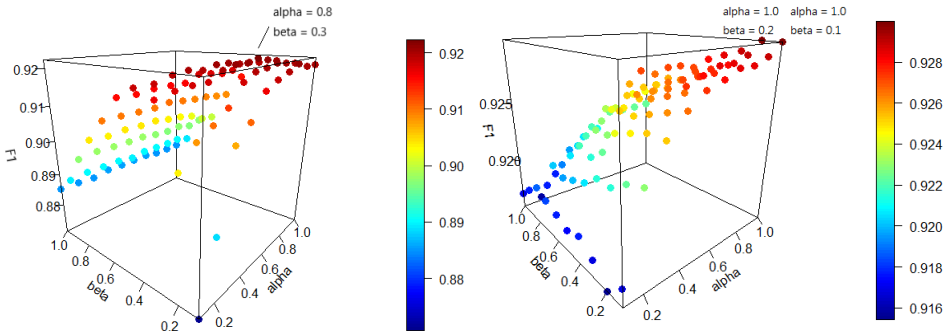
For the second part of the simulation, we use the Boğaziçi University distributed denial of service dataset [27]. The dataset consists of two parts - TCP flooding and UDP flooding attack with IP spoofing, and we choose one of the attack periods of the TCP flooding dataset as our simulation data. Since we focus on attack scenarios with no IP spoofing, we replace the spoofed attacker IP addresses into three different new IP addresses, to simulate the attack scenario without IP spoofing.

We assume that the effects of α_{thr} and β_{thr} don't interact with the effect of T_{thr}. Therefore, we tune α_{thr} and β_{thr} first while setting $T_{thr} = 50$. After finding the combination of α_{thr} and β_{thr} that performs the best, we use the combination to tune the threshold T_{thr}.

For each trial, our targets are IP addresses that have the largest absolute difference between the number of packets they sent and the number of packets they received. If a target IP address is preserved in the *ip* field in any entry, we treat the situation as a *hit* defined in the detection theory. To evaluate how a combination of thresholds performs, we use the F1 score as our metric.

Table 1. Trace parameters (Alpha and beta are for pareto distribution).

Parameter	Value
Number of total IP addresses	4000
Number of attackers	3
Number of victim	1
Number of entries	500
Alpha for number of packets sent by normal hosts (α_{np})	2.02
Beta for number of packets sent by normal hosts (β_{np})	99.88
Alpha for packet inter-sending time of normal hosts (α_{nt})	3.45
Beta for packet inter-sending time of normal hosts (β_{nt})	0.58
Alpha for packet inter-sending time of attackers (α_{at})	2.01
Beta for packet inter-sending time of attackers (β_{at})	42.26



(a) Distribution-based dataset. (b) Modified real dataset.

Fig. 7. Tuning alpha and beta using different dataset.

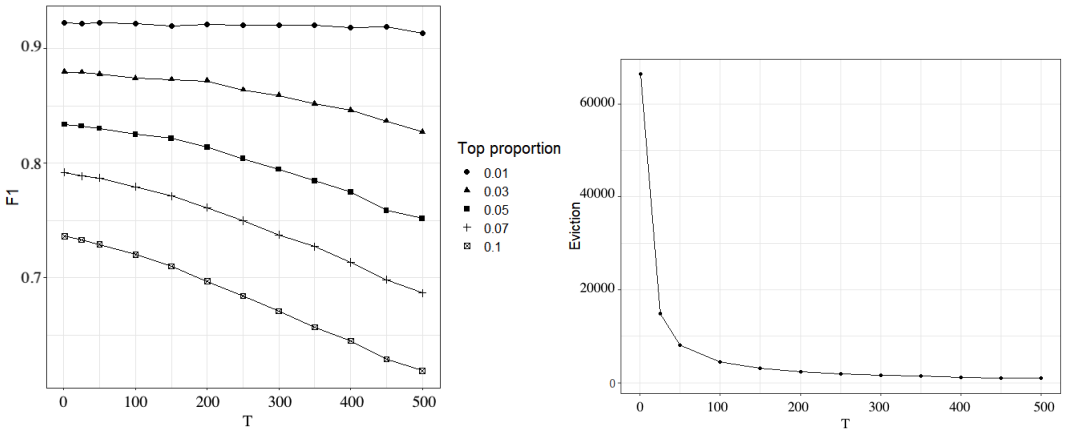
4.1.2 Tuning α_{thr} and β_{thr}

For each trace, we test 100 combinations of α_{thr} and β_{thr}, while the thresholds ranges from 0.1 to 1. The formula of α and β determine that the lower bounds are zero. On the other hand, according to our observation of sampling the data we get, most of the α and β are below 1.0. Therefore, we set the upper bound of α and β in this simulation to 1.0. The targets are set to the first 1% IP addresses that have the biggest absolute difference between the packets they sent and the packets they received. For the first part of the simulation, the result is shown below in Fig. 7 (a), and the second simulation in Fig. 7 (b).

In these two figures, we find that the best combination of α_{thr} and β_{thr} is 0.8 and 0.3 in the first part of the simulation, which has the best F1 score = 0.922.

On the other hand, when $(\alpha_{thr}, \beta_{thr}) = (1.0, 0.1)$ or $(1.0, 0.2)$, the model performs the best in the second part of simulation, which has F1 score = 0.93. In both figures, we can observe that their F1 score generally rises as α_{thr} rises and as β_{thr} drops, while there may also exist some local maximums.

Using $\alpha_{thr} = 0.8$ and $\beta_{thr} = 0.3$ in the first part of simulation, and $\alpha_{thr} = 1.0$ and $\beta_{thr} = 0.1$ in the second part of simulation, we tune the threshold T_{thr} subsequently.



(a) Tuning by F1 score with different target proportion.

(b) Tuning by number of evictions.

Fig. 8. Tuning T_{thr} in the first part of simulation.

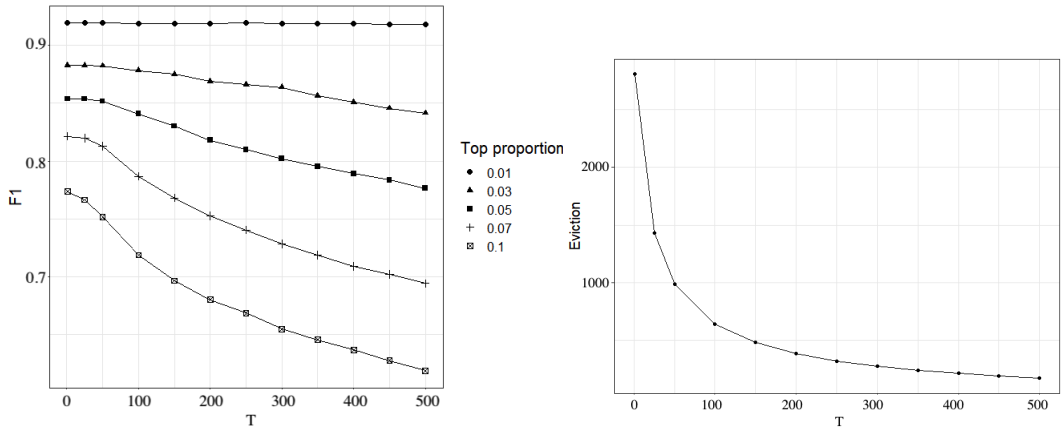
4.1.3 Tuning T_{thr}

We test the T_{thr} ranges from 1 to 500 and the results are shown below. In the first part of simulation using distribution-based data, Fig. 8 (a) shows how the F1 score changes as T_{thr} increases when setting the different proportion of the most asymmetric of IP addresses as targets. That is, the true positive of the F1 score adds when the target IP is stored in the table while the false positive adds when the others are stored. We can observe that the F1 score remains high when T_{thr} is below 50. However, as T_{thr} increases, the F1 score starts to drop at different slopes. Especially when the target proportion is high, the F1 score drops more significantly.

On the other hand, Fig. 8 (b) shows the number of evictions that happened in the simulation under the different value of T_{thr}. Although small T_{thr} results in high accuracy, the amount of evictions rises and it may cause the burden of switches to become heavier. Therefore, it exists a trade-off between the number of evictions and accuracy. As a result of the simulation, we think that it is appropriate to choose T_{thr} ranges from 50 to 100.

In the second part of the simulation using a modified real dataset, results are shown in Figs. 9 (a) and (b). We can observe similar trends as shown in 8a and 8b, but the F1 score drops more intensely and the number of evictions drops more smoothly at the beginning as T_{thr} increases. According to this simulation, we suggest choosing T_{thr} ranges from 50 to 100.

As a result, we find the trends between α_{thr}, β_{thr} ,and the F1 score. We also find the relationship between T_{thr}, the F1 score, and the number of evictions. We think that an appropriate α_{thr} is between 0.8 to 1.0. In addition, an appropriate β_{thr} is between 0.1 to 0.3, and 50 to 100 for T_{thr}. For subsequent experiments, we choose $\alpha_{thr} = 1.0$, $\beta_{thr} = 0.1$, and $T_{thr} = 100$ to be the thresholds.



(a) Tuning by F1 score with different target proportion.

(b) Tuning by number of evictions.

Fig. 9. Tuning T_{thr} in the second part of simulation.

4.2 Model Validity

In this section, we test the validity of our model. In other words, we run an experiment to see if our model can preserve the IP addresses we want under real traffics. We use part of the CTU-13 dataset recorded in the CTU University, Czech Republic, in 2011 [29].

We choose a time interval containing no attack packets to run a trace-based experiment. The interval lasts for about 780 seconds and we divide it into 13 one-minute-long traces. For each trace, we let the packets go through a single switch with our model installed. In addition, we manipulate the number of entries to observe how it affects the validity of the model. The result is shown below in Fig. 10. We can observe that when we set the target proportion to 0.01, the F1 score rises a little as the number of entries reaches 300 to 400. After that, the F1 score drops slightly as the number of entries increases.

In other cases, the F1 score generally becomes higher when the model uses more entries. The result is intuitive since the IP addresses should have more chance to be preserved in the model as the number of entries increases, resulting in the true positive rate to rise. In addition, when the model uses more entries, the collision rate between target IP addresses will decrease, which also benefits the true positive rate. However, in the case of setting the target proportion to 0.01, most targets are preserved in the model when the number of entries is around 400. Therefore, although the increase of entries can further let the rest of the targets easier to be preserved, the model will also preserve IP addresses that are not our target, causing the false positive rate to rise. Generally speaking, as the number of entries climbs to 500, the F1 score starts to become stable. In the case of target proportion = 1%, the model can achieve the F1 score above 0.9, which infers that our model can indeed record the IP addresses that have the biggest absolute difference between packets sent/received. In the next section, we test our model in attack scenarios, which contain different numbers of attackers and victims.

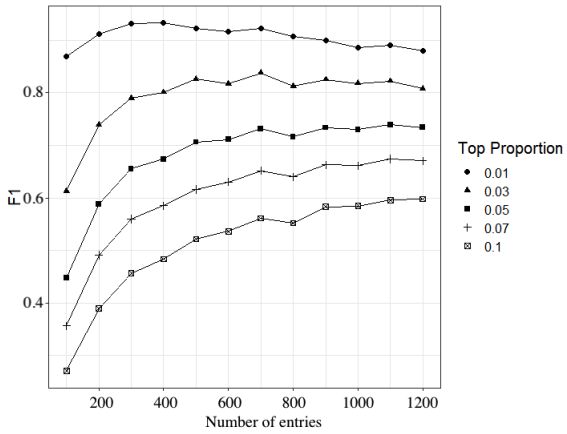


Fig. 10. F1 score under different number of entries.

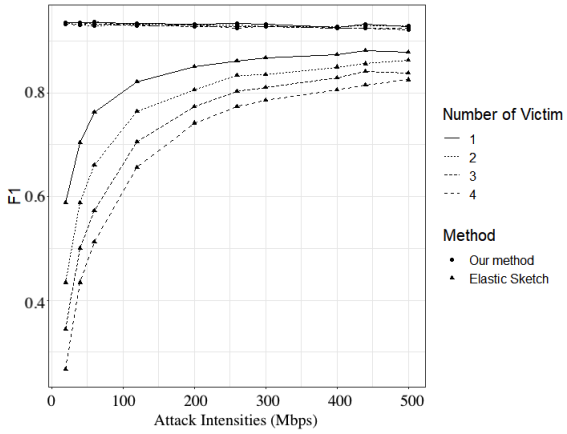


Fig. 11. F1 score under different attack intensities and number of victim.

4.3 Accuracy in Attack Scenario

Our model is designed to preserve data that is beneficial to DDoS attack detection. In this section, we observe how well our model preserves the IP addresses of DDoS attackers and victims under different scenarios. Furthermore, we compare this ability with Elastic Sketch [9]. Elastic Sketch also uses the sketch-based data structure and it is designed for preserving data of heavy hitters. We use another attack period of the Boğaziçi University distributed denial of service dataset [27]. We duplicate the malicious packets and reassign the source and destination IP addresses, to simulate different intensities of attacks and different numbers of victims.

Still using detection theory as our evaluation method, we set the targets to the IP addresses of attackers and victims in this simulation. The result is shown below in Fig. 11. We can observe that no matter under which condition, our method has F1 score remaining above 0.9. As attack intensity increases, there is only a tiny drop for about 0.01. On

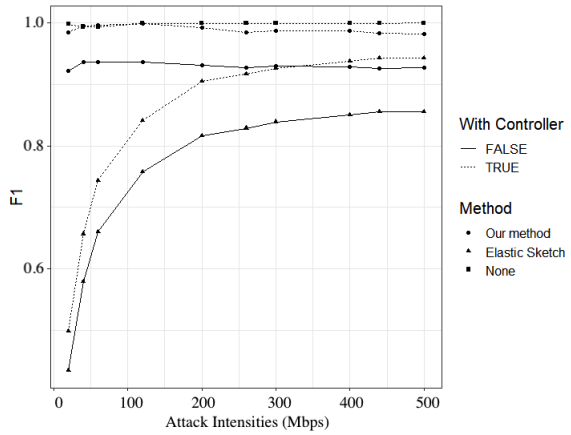


Fig. 12. F1 score under different conditions when a controller is involved.

the other hand, Elastic Sketch has the F1 score that rises significantly when the attack intensity starts to increase, flattens out when attack intensity comes to 200 Mbps, and falls between 0.8 to 0.9 when attack intensity reaches 500 Mbps. Furthermore, our model can collect IP addresses related to DDoS attackers or victims well, and the F1 score is 27% better than Elastic Sketch on average. We find that our method performs better than Elastic Sketch, especially when the number of victims increases. We think the result is caused mainly by the difference in the ability to preserve the victims' IP addresses since Elastic Sketch does not consider packets received by each IP address, while our model will also preserve IP addresses that receive a lot more packets than they send. In conclusion, when it comes to preserving IP addresses that are related to DDoS attackers and victims, our method outperforms Elastic Sketch and maintains high accuracy under different conditions.

4.4 Effect on Controller

Last, we test the effect of detecting DDoS attacks when a controller is involved. The controller collects data using different preserving methods from the switch and runs the k-means algorithm for detecting DDoS attackers and victims. We set the number of victims to 2 and test the F1 score of detecting DDoS attackers and victims under different conditions. The result is shown below in Fig. 12.

We can find that in contrast to using our method or Elastic Sketch, the F1 score increased by about 6% to 14% after using the k-means algorithm on the controller. On the other hand, when the switch does not use any method and sends statistics of each IP address to the controller, the F1 score of the detection performs the best.

In this condition, all information is recorded and information related to attackers and victims will never be lost.

The machine-learning method can cluster or filter by its algorithm and have the best F1 score. However, sending all information to the controller is not a practical method, since the size of data passed to the controller is too large, which should cause more serious problems when DDoS attacks truly exist. We calculate the size of data needs to be

Table 2. Data size from switch to controller.

Our Method	Elastic Sketch	None
10000 bytes	9000 bytes	43404 bytes

transformed from a single switch to a controller when using different method, and show the results in Table 2.

We can see that without using any method to filter information, data that need to be transformed is 3 times larger than using our method or Elastic Sketch. As the size of the network scales up, the burden caused by not using any method also raises. On the other hand, by using our method for filtering, we can eliminate about 75% information and keep the F1 score still high at the same time (Fig. 12). Therefore, the simulation shows that our method can effectively decrease the cost while remaining high-quality of detection.

5. CONCLUTIONS

DDoS attacks have caused serious problems and losses due to their ability to interrupt or suspend servers that provide important services such as financial activities. Many detection models have been proposed to detect DDoS attacks in various ways. However, most of them have some disadvantages, such as consuming a lot of bandwidth when collecting data.

In this paper, we introduced a sketch-based data structure to calculate the asymmetry of data transmission/reception, and use Ostracism to evict IP addresses below the threshold. The sketch-based data structure uses limited memory space to collect important data filtered through predefined thresholds. To improve the data collection process for further detection, we designed a sketch-based data structure that collects information based on the difference between data transmission and reception for each IP address.

In addition, we showed that our model can accurately reserve highly asymmetric IP addresses in packet transmission/reception. When the target ratio is set to 0.01, the F1 score can reach 0.9 or more, and the number of entries is appropriate. Finally, we show that our model can collect IP addresses related to DDoS attackers and victims very well, and the F1 score is 27% better than Elastic Sketch on average.

REFERENCES

1. A. Chadd, "Ddos attacks: past, present and future," *Network Security*, Vol. 2018, 2018, pp. 13-15.
2. C.-Y. Chen, L.-A. Chen, Y.-Z. Cai, and M.-H. Tsai, "Rnn-based ddos detection in iot scenario," in *Proceedings of IEEE International Computer Symposium*, 2020, pp. 448-453.
3. Y.-Z. Cai, Y.-T. Wang, and M.-H. Tsai, "Dynamic adjustment for proactive flow installation mechanism in sdn-based iot," *Computer Networks*, Vol. 194, 2021, p. 108167.
4. M.-H. Tsai, C.-C. Chuang, S.-F. Chou, A.-C. Pang, and G.-Y. Chen, "Enabling efficient and consistent network update in wireless data centers," *IEEE Transactions on Network and Service Management*, Vol. 16, 2019, pp. 505-520.

5. P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese *et al.*, "P4: Programming protocol-independent packet processors," *ACM SIGCOMM Computer Communication Review*, Vol. 44, 2014, pp. 87-95.
6. J. Mirkovic and P. Reiher, "A taxonomy of ddos attack and ddos defense mechanisms," *ACM SIGCOMM Computer Communication Review*, Vol. 34, 2004, ppp. 39-53.
7. N. Dayal, P. Maity, S. Srivastava, and R. Khondoker, "Research trends in security and ddos in sdn," *Security and Communication Networks*, Vol. 9, 2016, pp. 6386-6411.
8. Y. Cai, T. Lin, Y. Wang, Y. Tuan, and M. Tsai, "E-replacement: Efficient scanner data collection method in p4-based software-defined networks," *International Journal of Network Management*, Vol. 31, 2021, pp. 1099-1190.
9. T. Yang, J. Jiang, P. Liu, Q. Huang, J. Gong, Y. Zhou, R. Miao, X. Li, and S. Uhlig, "Elastic sketch: Adaptive and fast network-wide measurements," in *Proceedings of ACM SIG Conference on Data Communication*, 2018, pp. 561-575.
10. R. Wang, Z. Jia, and L. Ju, "An entropy-based distributed ddos detection mechanism in software-defined networking," in *Proceedings of IEEE Trustcom/BigDataSE/ISPA*, Vol. 1, 2015, pp. 310-317.
11. R. N. Carvalho, J. L. Bordim, and E. A. P. Alchieri, "Entropy-based dos attack identification in sdn," in *Proceedings of IEEE International Parallel and Distributed Processing Symposium Workshops*, 2019, pp. 627-634.
12. R. Swami, M. Dave, and V. Ranga, "Defending ddos against software defined networks using entropy," in *Proceedings of the 4th International Conference on Internet of Things: Smart Innovation and Usages*, 2019, pp. 1-5.
13. X. Yang, B. Han, Z. Sun, and J. Huang, "Sdn-based ddos attack detection with cross-plane collaboration and lightweight flow monitoring," in *Proceedings of IEEE Global Communications Conference*, 2017, pp. 1-6.
14. P. Wang, K.-M. Chao, H.-C. Lin, W.-H. Lin, and C.-C. Lo, "An efficient flow control approach for sdn-based network threat detection and migration using support vector machine," in *Proceedings of IEEE 13th International Conference on e-Business Engineering*, 2016, pp. 56-63.
15. M. Nobakht, V. Sivaraman, and R. Boreli, "A host-based intrusion detection and mitigation framework for smart home iot using openflow," in *Proceedings of the 11th International Conference on Availability, Reliability and Security*, 2016, pp. 147-156.
16. D. Hu, P. Hong, and Y. Chen, "Fadm: Ddos flooding attack detection and mitigation system in software-defined networking," in *Proceedings of IEEE Global Communications Conference*, 2017, pp. 1-7.
17. A. Sahi, D. Lai, Y. Li, and M. Diykh, "An efficient ddos tcp flood attack detection and prevention system in a cloud environment," *IEEE Access*, Vol. 5, 2017, pp. 6036-6048.
18. M. Charikar, K. Chen, and M. Farach-Colton, "Finding frequent items in data streams," in *Proceedings of International Colloquium on Automata, Languages, and Programming*, 2002, pp. 693-703.
19. H. Liu, Y. Sun, and M. S. Kim, "Fine-grained ddos detection scheme based on bidirectional count sketch," in *Proceedings of IEEE 20th International Conference on Computer Communications and Networks*, 2011, pp. 1-6.

20. G. Cormode and S. Muthukrishnan, "An improved data stream summary: the count-min sketch and its applications," *Journal of Algorithms*, Vol. 55, 2005, pp. 58-75.
21. G. Cormode and M. Muthukrishnan, "Approximating data with the count-min sketch," *IEEE Software*, Vol. 29, 2011, pp. 64-69.
22. V. Sivaraman, S. Narayana, O. Rottenstreich, S. Muthukrishnan, and J. Rexford, "Heavy-hitter detection entirely in the data plane," in *Proceedings of ACM Symposium on SDN Research*, 2017, pp. 164-176.
23. A. C. Lapolli, J. Adilson Marques, and L. P. Gaspary, "Offloading real-time ddos attack detection to programmable data planes," in *Proceedings of IFIP/IEEE Symposium on Integrated Network and Service Management*, 2019, pp. 19-27.
24. A. Metwally, D. Agrawal, and A. El Abbadi, "Efficient computation of frequent and top-k elements in data streams," in *Proceedings of International Conference on Database Theory*, 2005, pp. 398-412.
25. Y. Xu and Y. Liu, "Ddos attack detection under sdn context," in *Proceedings of the 35th Annual IEEE International Conference on Computer Communications*, 2016, pp. 1-9.
26. J. Gordon, "Pareto process as a model of self-similar packet traffic," in *Proceedings of IEEE Global Telecommunications Conference*, Vol. 3, 1995, pp. 2232-2236.
27. D. Erhan and E. Anarım, "Boğaziçi university distributed denial of service dataset," *Data in Brief*, Vol. 32, 2020, p. 106187.
28. F. Huebner, D. Liu, and J. Fernandez, "Queueing performance comparison of traffic models for internet traffic," in *Proceedings of IEEE Global Telecommunications Conference*, Vol. 1, 1998, pp. 471-476.
29. S. García, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods," *Computers & Security*, Vol. 45, 2014, pp. 100-123.

Ting-Yu Lin received the BS degree in Department of Computer Science and Information Engineering from National Cheng Kung University, Taiwan, in 2020. She is currently pursuing an MS degree with the Department of Computer Science and Information Engineering, National Cheng Kung University, Taiwan. She was a Summer Intern with TSMC, Inc., Taiwan between July and August in 2020. Her research interests include network security and management in software-defined networks.

Ching-Yuan Wang received the BS degree in Department of Psychology from National Chen Kung University, Taiwan, in 2019. He received an MS degree in Department of Computer Science and Information Engineering, National Chen Kung University, Taiwan, in 2021. His research interests include network security in software-defined networks.

Ya-Pei Tuan received the BS degree in Department of Computer Science and Information Engineering from National Cheng Kung University, Taiwan, in 2020. She is currently pursuing the Ph.D. degree with the Department of Computer Science and Information Engineering, National Cheng Kung University, Taiwan. She is also studying in Teacher Education Program. Her research interests include network security in 5th generation mobile networks.

Meng-Hsun Tsai is a Professor in Department of Computer Science and Information Engineering, National Cheng Kung University. He received the BS, the MS and the Ph.D. degrees from National Chiao Tung University in 2002, 2004 and 2009, respectively. He was a visiting scholar at University of Southern California between July and August 2012. He was a recipient of the Exploration Research Award of Pan Wen Yuan Foundation in 2012 and the Outstanding Contribution Award from IEEE Taipei Section in 2010. He is a senior member of the IEEE. His current research interests include design and analysis of mobile networks, software defined networking, Internet of Things, and cybersecurity.

Yean-Ru Chen is an Assistant Professor in Department of Electrical Engineering, National Cheng Kung University. She received the Ph.D. degree from National Taiwan University in 2014 and served as a senior formal verification engineer in MediaTek and Cadence Design Systems in 2015 and 2016, respectively. Her current research interests include security verification with formal methods, especially for micro processors and neural network designs.