2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS) | 978-1-6654-7177-0/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICDCS54860.2022.00056

D3: Lightweight Secure Fault Localization in Edge Cloud

Songtao Fu*, Qi Li*†‖, Xiaoliang Wang‡, Su Yao*†, Xuewei Feng*, Ziqiang Wang§,
Xinle Du*, Kao Wan¶, Ke Xu*†‖
* Tsinghua University, † BNRist, ‡ Capital Normal University, § Southeast University,
¶ Peng Cheng Laboratory, ‖ Zhongguancun Laboratory
Emails:{fust18@mails., qli01@, yaosu@, fengxw18@mails., dxl18@mails., xuke@}tsinghua.edu.cn
wangxiaoliang@cnu.edu.cn, ziqiangwang@seu.edu.cn, sonicwk@hotmail.com

***Abstract*— In pursuit of high-performance applications, the cloud is moving out of the data center and towards the edge. Secure data forwarding is critical for the users between the edge and the remote cloud. In this paper, we propose D3 (Demon Detector in Data Plane), a lightweight, secure fault localization mechanism, which can enable the users in the edge cloud to localize faulty links and thus avoid the faulty links to guarantee secure data forwarding along the path to the remote cloud. D3 utilizes the user to instruct the transit routers, thus empowering the user to detect whether the transit routers forward the packet as expected. Compared with existing schemes that are difficult to be deployed in practice due to the incurred heavy storage, computation, and communication overhead, D3 offloads most of the transit router's storage and computation overhead, thus dramatically improving the deployment efficiency. Particularly, the length of the additional packet header in D3 is 2-5 times less than the state-of-the-art mechanisms, and the extra control packet overhead is ten times less while keeping a little constant storage overhead in the data plane. The evaluations in BMv2 and Barefoot Tofino hardware show that D3 could achieve high fault localization accuracy and efficiency.**

***Index Terms*—Edge Cloud, Path Verification, Fault Localization**

I. INTRODUCTION

The promise and potential of applications ranging from Internet-of-Things (IoT) to Autonomous Vehicles have triggered edge computing to become a research hotspot [1]. This shift to a distributed edge cloud model will require the entire cloud ecosystem to think differently about the role of network connectivity. The edge server performs preprocessing (e.g., sampling), significantly reducing the amount of data being transferred and allowing structured data to be directly sent to the remote cloud server for storage or further processing.

The data forwarding between the edge server and remote cloud server is inter-domain communication since the data packet might transit several ASes (We denote an AS as a domain). With emerging path-aware network such as SCION [2] and SR [3], the user (edge server or remote cloud server) could select the transit AS border router and embed its forwarding policy (e.g., expected path) in the packet header. But an adversary in a specific AS might drop, delay, modify, fabricate or redirect packets, thus violating the forwarding policy. To localize the faulty link which violates the forwarding policy, the user in the edge cloud needs a fault localization mechanism to detect the malicious AS, and then circumvent it to achieve secure data forwarding.

The fault localization mechanisms (e.g., AudIt [7]) lack the authentication of the forwarding packets. The $detector$ in these mechanisms cannot verify the cryptographic $mark$[1] in the packets. Then the adversary could manipulate the packet and subvert the fault localization. The secure fault localization could defend the adversary by the symmetric key shared between the $detector$ and each router. The most important characteristic of secure fault localization is it requires a key infrastructure [8]. It provides a confidential way to share the dynamic key between the routers and the $detector$ (source or other entity). Then the $detector$ could localize the faulty link with the cryptographic $mark$.

Unfortunately, as presented in Table [I], the state-of-the-art secure fault localization mechanisms are impractical in edge cloud for the following reasons:

Heavy storage and computation overhead in each router. ShortMAC [4] has intolerable storage (46 GB) overhead in the data plane, thus could not be applied to the inter-domain context. Faultprints [5] and RFL [6] incur prohibitive computation overhead in each router with per-session asymmetric cryptography operation and per-packet cryptography operation to derive the dynamic key on the fly.

High communication overhead in network. The communication overhead in the network comes from extra control packets and additional packet header. The number of the extra control packet is high in Faultprints and RFL. The source needs to probe $O(n)$ sampling packets in an epoch. Meanwhile, the additional packet header is 96 B and 58 B for the path length of 5, and it would be longer with the longer path length.

Deployment barrier. The MAC operation such as AES is hard to implement in router hardware since it significantly decreases forwarding performance [9]. And the per-packet timestamp is also impractical in large-scale networks [5]. It has little incentive towards the early adopter.

Compared with traditional inter-domain communication, there are two different characteristics of the communication in edge cloud: one is the number of users is much less, the other is that it is easier for the user in the data center and edge to upgrade new protocol. From this perspective, we propose a lightweight and secure fault localization mechanism named D3

[1]This paper calls the source user who localizes the faulty link on the forwarding path as detector and names the Message Authentication Code (MAC) as $mark$.

2575-8411/22/$31.00 ©2022 IEEE
DOI 10.1109/ICDCS54860.2022.00056

TABLE I
THE PROPERTIES OF DIFFERENT FAULT LOCALIZATION MECHANISMS

	Data plane storage per 100 Gbps link	Inter-domain support	No per-session asymmetric cryptography	No per-packet cryptography for dynamic key	Control packets in an epoch	Additional packet header (5 hops)
ShortMAC [4]	$46GB$	✗	✗	✓	$O(1)$	2 B
Faultprints [5]	$468MB$	✓	✗	✗	$O(n)$	96 B
RFL [6]	$468MB$	✓	✗	✗	$O(n)$	58 B
D3	$2.56MB$	✓	✓	✓	$O(1)$	20 B

(Demon Detector in Data Plane) via designing a new protocol to offload the router's overhead to the user. Particularly, the length of the additional packet header in our mechanism is much shorter than the state-of-the-art mechanisms (i.e., 2-5 times less than Faultprints and RFL). Meanwhile, our mechanism can preserve an equal fault localization accuracy. Our contributions are three-fold:

- **Light overhead in each router.** We utilize stateless processing at each router instructed by the probabilistic and cryptographic instructions determined by the source. As shown in Table [I], each router only needs to store the constant dynamic keys. And it records the per-session counters in the control plane with a sampling instruction determined by the source, thus having low data plane storage overhead (i.e., 2.56 MB). It does not need the per-session asymmetric cryptography and per-packet cryptography to derive the dynamic key, thus decreasing the computation overhead.
- **Lower communication overhead in network.** We utilize stateful processing at the source and destination. The users store most of the processing information of a session. The source evaluates the end-to-end corruption ratio based on the ACK from the destination. It only sends one $probe$ towards a specific router to get one $reply$. Table [I] shows that the number of control packets in an epoch is $O(1)$. Meanwhile, D3 has a constant and short additional packet header (20 B).
- **Convenient deployment.**We utilize the two-round Even-Mansour (2EM) [10] instead of AES [9] as the MAC operation in the data plane. With the emerging programmable switch [11] supporting packet processing using domain-specific language (e.g., P4 [12]), D3 accomplishes the router's processing within one pipeline. The evaluation in BMv2 testifies that the measured end-to-end corruption ratio is nearly the same as the theoretical value, and the localization accuracy could achieve 95% to localize the adversary with a corruption ratio of 5%. The evaluation in Barefoot Tofino hardware testifies that D3 could perform the MAC operation within one pipeline and incurs little computation overhead compared with the IPv6 packet forwarding.

The rest of the paper organizes as follows: In the next section, we present the background, adversary model, and our assumptions. Section III presents the design of D3 at a high level, section IV details the D3, section V analyzes its characteristics and benefits. Section VI presents the implementation and the evaluation. Section VII discusses the deployment and the limitation of D3. Section VIII describes related works. We conclude the paper in the last section.

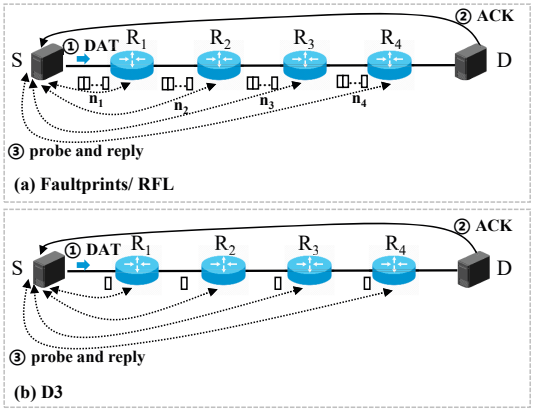

Fig. 1. An example of fault localization in different mechanisms. S and D represent the source and destination user (edge server or remote cloud server), R_i represents an ingress border router of an AS.

II. PROBLEM SETTING

We first describe the background of the fault localization in the edge cloud, then discuss the adversary model of fault localization. And we describe the requirements of the localization and some critical assumptions.

A. Background

From a bottom-up view, a general practical edge cloud model includes three layers. Basically, end devices at the END layer communicate with access points (APs) via the wireless channel. Then, LAN at the EDGE layer consists of many APs and edge servers (deployed at APs). The APs within a LAN are connected with each other, and they communicate with the remote cloud server via the Internet [13]. Finally, the remote cloud server at the CLOUD layer provides online services.

To guarantee secure data forwarding, the source in the edge cloud could achieve secure fault localization based on the actual data traffic. Fig. 1 presents the basic procedures of fault localization enforced by the source (S). ① Data forwarding: S instructs each transit router on the path to record the testimony.

An adversary might manipulate the testimony by learning the processing of other ASes. An efficient way to defend it is that S shares a symmetric key with each router. Then S announces each router to record the testimony according to a specific $mark$, calculated with the symmetric key and particular data packet (DAT) fields. ② End-to-end corruption ratio: An epoch has a certain number of packets (e.g., 4000). At each epoch, S evaluates the end-to-end corruption ratio based on the ACK from the destination (D). To prevent the adversary subvert the evaluation, S and D have a shared symmetric key to guarantee that D can verify each packet. ③ $probe\ and\ reply$: If the end-to-end corruption ratio indicates there might be an adversary on the path, the source receives the testimony with the $probe\ and\ reply$ mechanism towards each router. An adversary might distinguish the $probe\ and\ reply$ packet and alter the testimony. The $probe\ and\ reply$ packet should also be encrypted by the relative symmetric key.

B. Adversary model

We assume that an adversary in the Dolev-Yao model can compromise any routers on a path. The compromised routers can corrupt the packet by dropping, delaying, modifying, or fabricating it, or launch a path inconsistency attack by redirecting the packets to unexpected AS. These routers can launch a coward attack if they know they cannot be accurately localized. They could also frame other ASes of adversarial activity. Furthermore, several colluders could exchange information (e.g., secret keys or link information) and launch the packet corruption or path inconsistency attack. However, the adversary cannot eavesdrop, inject or influence traffic on links not adjacent to it [14].

C. Requirements and assumptions

The requirements for an Internet-scale deployment of fault localization are as follows:

Strong security and light processing. An adversary could not violate or evade the localization by observing the forwarded packet. And the light processing in the router is necessary for the deployment.

High fault localization accuracy. We denote the localization accuracy as: for a certain number of epochs, the ratio between the number of epochs the source successfully localizes the adversary (without false positive and false negative) and the total number of epochs which exists the adversary. The localization accuracy higher than 95% is practical [5].

We assume that the source knows an AS ingress router-level path to the destination. And the key distribution server (KDS) in each AS could derive and distribute dynamic keys to relative entities based on the following assumptions:

Secure dynamic key deriving system. Two KDSes in AS S and AS I share the AS fixed key (AK_{SI}) confidentially. The KDS could derive the router's dynamic key (RDK) and the user's dynamic key (HDK) with a MAC operation (e.g., AES) based on the AK_{SI}.

Secure dynamic key distribution. The user and router have a secure channel to get the dynamic keys from KDS in the local AS [15] [16]. Each router presets the dynamic keys from the control plane, thus decreasing the computation overhead to derive the dynamic key on the fly.

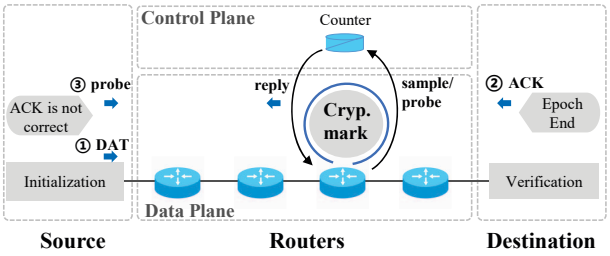

Fig. 2. Processing in different entities

III. Overview

As shown in Fig. 2, the most important characteristic of D3 is it hardly affects the data plane forwarding. At a high level, D3 utilizes the source to program each router on the path. With the cryptographic $mark$ ($Cryp.\ mark$) in each packet (section IV-A), D3 achieves fault localization with three procedures, which include ① Data forwarding, ② End-to-end corruption ratio, and ③ $probe\ and\ reply$.

First, the data forwarding includes packet initialization at the source (section IV-B), and lightweight processing at the router (section IV-C). With packet initialization, the source nests the instruction in the $Cryp.\ mark$ and sends a certain number of DAT towards the destination in each epoch. With lightweight processing, each stateless router only processes the $Cryp.\ mark$ in the data plane. If the $Cryp.\ mark$ instructs the router to sample the packet, it records the $counter$ as testimony in the control plane. In contrast, if the $Cryp.\ mark$ instructs to $probe$ a packet, it replies to the source with a $reply$ (packet) includes the $counter$ information. Second, to guarantee a comprehensive end-to-end corruption ratio, the processing at the destination includes the verification for each packet and sending the ACK to the source at each end of an epoch (section IV-D). Finally, D3 achieves fault localization at the source (section IV-E). If the end-to-end corruption ratio is not as expected or no ACK after a timeout, the source sends a $probe$ (packet) towards each router and localizes the faulty link based on the $reply$ from each router, as shown in Fig. 1 (b). While in Fig. 1 (a), the routers record n sampling data packets (n_1, n_2, n_3, and n_4, respectively) in Faultprints/RFL. The source needs to probe $O(n)$ sampling packets in an epoch.

We wedge the additional packet header of D3 as part of the routing header in the IPv6 packet [17]. Fig. 3 shows our designed packet header of D3 in IPv6.

- $flag$ (8-bits): We utilize the leftmost three bits to indicate the length of $mark_r$ for each router.
- $epoch$ (8-bits): The epoch number.
- seq (16-bits): The sequence number of a specific packet.
- tag (32-bits): The tag to instruct each router to select the relative RDK, includes the time slot (16-bits) and the user identifier (16-bits).

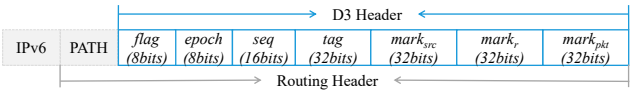

Fig. 3. Packet Header of D3 in IPv6

- $mark_{src}$ (32-bits): The $mark$ calculated by the source which instructs a selected router to sample or reply the packet. The selected router updates it with a new $mark$ after successful verification.
- $mark_r$ (32-bits): The $marks$ verified at each router.
- $mark_{pkt}$ (32-bits): The $mark$ calculated by the source with the packet payload, $mark_{src}$, $mark_r$, and HDK, verified by the destination.

IV. PROTOCOL DESIGN

We first detail how to calculate the cryptographic $mark$ for each packet, then we analyze the packet initialization at the source, the lightweight processing at each router, and the fault localization at the source based on the ACK and $reply$.

A. Cryptographic $mark$ in each packet

The cryptographic $mark$ is calculated by a specific MAC operation with RDK_{si} and HDK_{sd}, to guarantee that an adversary could guess the correct $mark$ with no more than randomness. The KDS first derives the dynamic key between the source and router i with the fixed key (AK_{SI}) between the source's AS and router i's AS:

$$RDK_{si} = MAC_{AK_{SI}}(id_s||id_i||tag) \quad (1)$$

$MAC(.)$ represents the MAC computation (e.g., AES). id_s and id_i are the identifiers of router s (the border router of source AS) and router i. In practice, all the border routers of an AS could utilize the same id (e.g., a virtual id) if there's more than one border router in an AS. tag is an identifier to distinguish a user (named uid) and the time slot (ts), '||' represents the concatenation operation.

The bit width of uid affects the dynamic key storage overhead in the router, and the bit width of ts affects the dynamic key switching frequency. The bit width K of uid would exponentially increase the storage overhead with 2^K. That means each router only stores the dynamic keys for a certain number of uid. There comes a challenge that some users in an AS utilize the same dynamic key in a specific time slot, giving an adversary a chance to require the dynamic key towards a specific router i, then subverts the secure localization. We utilize two measures to overcome it.

Firstly, the users from one group (with the exact security requirement) use the same uid. Then the dynamic key only affects the users in the same group. Furthermore, the KDS could only distribute the dynamic key towards the edge router of a group instead of the user. The edge router works as an agent of a group. With an intra-domain secure channel, the user sends the D3 packet to the edge router. The edge router operates the packet header with the dynamic keys towards the AS border routers on the path. The user eventually achieves the localization after receiving the packet from the edge router. It is sustainable since the throughput in the edge router is much lower than in the AS border router.

Secondly, we could adjust the key switching frequency with ts. A 16 bits width ts could divide 24 hours into 2-second granularity. In practice, an AS could customize the key switching time by utilizing different steps of ts. For example, an AS switches dynamic key every 2 seconds could increase the ts with the step of 1. Another AS needs to switch the dynamic key every 200 seconds with the step of 100.

With this design, the user could customize its dynamic key policy with other ASes, including the bit width of uid and the switch step of ts. The KDS then presets the dynamic keys in a border router. From this perspective, D3 is an edge-based mechanism instead of an end-based mechanism. We will detail the storage overhead in section V.

With RDK_{si}, the source instructs each router to probabilistic sample a packet by a cryptographic $mark$. Each router only calculates one MAC operation to get a $mark$ with no less than 64 bits output with the link information and cst_{PH}. cst_{PH} is the constant packet header of D3, which includes the IP address (especially the source IP address), the path information, and the $flag$, $epoch$, seq, and tag. Router i calculates the $mark_i$ as Equation (2).

$$mark_i = MAC_{RDK_{si}}(id_{i-1}||id_i||cst_{PH}) \quad (2)$$

The KDS calculates the HDK_{sd} between the source and destination:

$$HDK_{sd} = MAC_{AK_{SD}}(IP||path||tag) \quad (3)$$

In which AK_{SD} is the AK between the source AS and destination AS, IP is the concatenation of the source and the destination's IP address, $path$ is the concatenation of all the routers' id on the path, tag is the information of the first time slot in a session. With the HDK_{sd}, the source calculates the $mark_{pkt}$ as Equation (4). In which $marks$ represent the $mark_{src}$ and $mark_r$ in the received packet header. $hash$ represents the hash of the upper layer information or payload. $[0:32]$ represents truncating the rightmost 32-bits for efficiency reasons. It guarantees that the destination could verify the source and the packet's integrity.

$$mark_{pkt} = MAC_{HDK_{sd}}(cst_{PH}||marks||hash)[0:32] \quad (4)$$

B. Packet initialization at source

As shown in Algorithm 1, the source first gets the $path$ and RDK_{path}, HDK_{sd}, and determines the number of packets in an epoch, e.g., $N = 4000$. At the beginning of each epoch, the source initializes the $epoch$, seq (line 1-2). With seq from 0 to $N-1$, it nests the cst_{PH} in the packet header (line 3-4). Then calculates the three $marks$ ($mark_{pkt}, mark_{src}, mark_r$), and nests them in the packet header (line 5-6), forwards the packet to the first router, and increases the seq (line 7). At the end of each epoch, the source processes the packet in the next epoch if it needs to send another packet (line 8). If the source

Algorithm 1: packet initialization at source

input : $path, RDK_{path}, HDK_{sd}, N$

```
for each epoch do
    seq = 0
    while seq < N do
        nest cst_PH in the packet header
        calculate the marks
        nest marks in the packet header
        forward packet and seq = seq + 1
    epoch = (epoch+1)%256
```

selects router i to process the packet, it nests the $mark_{src}$ in the packet header as:

$$mark_{src} = mark_i[32:64] + ins \tag{5}$$

The ins is an instruction that indicates the packet type. For example, 0 represents a data packet that does not need to record the sampling counter (DAT_{UNS}), 1 represents a data packet that needs to record the sampling counter (DAT_{SAM}), 2 represents the $probe$. We name this mechanism the $blind\ policy$ since only the source and the selected router know the sampling decision. An adversary could not distinguish the packet type with the confidential ins, thus only corrupting these packets with no more than randomness. For $reply$ and ACK, we nest $ins = 3\ and\ 4$ in the $mark_{pkt}$.

We utilize $mark_r$ for each router to verify and filter the malicious packet. The source initializes the $mark_r$ as:

$$mark_r = mark_0[0:4]||\cdots||mark_n[0:4] \tag{6}$$

For each router, $mark_{src}$ and $mark_r$ come from one $mark$. The $binding\ marks$ not only decrease the MAC operation of a router but also improve the security characteristic with the $blind\ policy$. We will detail it in section V.

C. Lightweight processing at router

Algorithm 2: processing at each router

input : $pkt, RDK_{si}, id_i, id_{i-1}$

```
calculates mark'_{src_i} and mark_{r_i}'
ins = mark_{src_i} − mark'_{src_i}
switch ins do
    DAT_UNS : update mark_{src_i}
    DAT_SAM : Counter++ and update mark_{src_i}
    probe : reply to the source
if mark_{r_i} != mark_{r_i}' then
    fCounter++ and filter the packet
forward packet to next router
```

Each router keeps two counters in the control plane. One is for DAT_{SAM} ($Counter$), the other is for false packet ($fCounter$). As shown in Algorithm 2, when receiving a packet, each router first calculates $mark_i{}'$ with Equation (2), then derives $mark'_{src_i} = mark_i{}'[32:64]$ and $mark_{r_i}{}' = mark_i{}'[0:4]$ (line 1). It calculates the ins with Equation (5), and processes the packet with different values of ins (line 2-6). For DAT_{UNS}, it updates the $mark_{src}$ as:

$$mark_{src_i} = mark_i[0:32] \oplus mark_i[32:64] + ins \tag{7}$$

For DAT_{SAM}, it clones the packet header (the IP and D3 packet header) to the control plane, and updates the $mark_{src}$ as Equation (7). The control plane increases the $Counter$ in the relative epoch.

For $probe$, it clones the packet header to the control plane. The control plane sends a $reply$ with the information of the two counters to the source. It updates the $marks$ according to the reverse path. We utilize the dynamic key RDK_{si} to encrypt the counter information as the payload of $reply$:

$$reply_{info} = encrypt_{RDK_{si}}(Counter||fCounter) \tag{8}$$

It then compares the $mark'_{r_i}$ ($mark_i[0:4]'$) with $mark_{r_i}$ in the packet header (line 7), increases the $fCounter$ in the control plane and filters the packet with the wrong $mark_{r_i}$ (line 8). If this router does not filter this packet, it forwards the packet to the next router (line 9).

The $probe$ uses the reply border router's address as the destination address (instead of the destination's). In contrast, the $reply$ uses its address as the source address. There comes a challenge that an adversary might distinguish the $probe$ and $reply$ with the IP address.

To prevent this, for the $probe$, we utilize the destination's IP address instead of the reply border router's as the destination IP address. The reply border router could distinguish it with ins while the adversary could not.

For the $reply$, we modify the source IP address's suffix with a random bit string. The adversary could not distinguish it from the DAT with the source IP address. The source could distinguish it from the uid information in the packet header. It utilizes the dynamic key to calculate the $mark$, then gets the ins information.

D. Processing at destination

Algorithm 3: processing at destination

input : pkt, HDK_{sd}

```
initialize Couter, fCounter = {0}
if (mark_pkt == mark'_pkt) then
    Counter++
else
    fCounter++
if the end of an epoch then
    forward ACK toward the source
```

The destination processes the packet as Algorithm 3. It first initializes the successful and false counter ($Couter$ and

$fCounter$) with 0 (line 1). When the destination receives a packet, it first calculates $mark'_{pkt}$ with Equation (4).

If the $mark_{pkt}$ in the packet header is correct, it increases the $Couter$ (line 2-3). In contrast, it increases the $fCouter$ when a packet has the wrong $mark_{pkt}$ (line 4-5).

The destination sends two counters to the source with an ACK at the end of an epoch (line 6-7). An ACK has the same packet header as the DAT, except the reverse path might not be symmetrical with the forward path. The destination used the HDK_{sd} to encrypt the information. It is worth mentioning that the destination could send all the lost seq of an epoch to facilitate the source to get a comprehensive end-to-end corruption ratio.

$$ACK_{info} = encrypt_{HDK_{sd}}(Counter||fCounter) \quad (9)$$

E. Fault localization at source

Algorithm 4: fault localization at source

input : $pkt, path, HDK_{sd}, RDK_{path}$

1 **if** $ins == ACK$ **then**
2 **if** $corruption\ ratio > \theta_{threshold}$ **then**
3 *probe toward each router*
4 **if** no ACK after timeout **then**
5 *probe toward each router*
6 **if** $ins == reply$ **then**
7 *calculate* $repu_i$
8 *localize the faulty link with reputation gaps*

The source localizes the faulty link with Algorithm 4. At each end of an epoch, it decrypts the ACK and gets the information if it receives an ACK (line 1). Then it computes the end-to-end reputation as:

$$repu = (Couter - fCounter)/N \quad (10)$$

In which N is the number of packets in an epoch (e.g., 4000). We denote the end-to-end corruption ratio as $corr_e = 1 - repu$. For a path length of k, the natural packet loss ratio in a router is ρ_n, the theoretical corruption ratio incurred by the natural packet loss is $\theta_n = 1 - (1 - \rho_n)^k$. We denote the corruption ratio threshold as:

$$\theta_{threshold} = \rho/2 + \theta_n \quad (11)$$

The source does not send $probe$ if the measured packet corruption ratio $corr_e$ is no more than the threshold $\theta_{threshold}$, e.g., 3.7% for localizing the adversary with corruption ratio of 5%, natural packet loss of 0.3%, and path length of 5. Or else, the source forwards the $probe$ towards each router (line 2-3). If there's no ACK after a timeout, the source also forwards the $probe$ (line 4-5).

The source calculates the reputation of each router after it decrypts the $Counter[i]$ and $fCounter[i]$ from the $reply$ (line 6-7). If $fCouter[i]$ is more than a threshold (e.g., $fCouter[i] > \beta,\ \beta = 100$), it localizes the link between $i-1$ and i as fault. Or else, the source calculates the reputation with Equation (12), the reputation gap ($repu_i - repu_{i+1}$) between two neighbor routers larger than the gap caused by natural packet loss possibly signals the fault behavior (line 8).

$$repu_i = (Couter[i] - fCouter[i])/Counter_{src}[i] \quad (12)$$

The setting of the threshold is based on the user's requirement, e.g., an application couldn't bear the end-to-end corruption ratio of more than 5%. In section VI, we will testify that the threshold is accurate since the user samples all the packets. We set the value of $fCounter[i]$ as 100, since that for 4000 packets in an epoch, 100 damaged packets represent that about 2.5% of the packets were damaged, which is intolerable for the user. In D3, the accurate localization is that the source localizes the specific faulty link without false positive and false negative. For example, if R_2 in Fig. 1 corrupts packets with the ratio of $\rho = 3\%$, the reputation gap between R_2 and R_3 is within $\rho \pm \rho/2$ (e.g., $3\% \pm 1.5\%$), and others are less than $\rho/2$. The false positive means that other links (e.g., between R_3 and R_4) have a gap of more than $\rho/2$. In contrast, the false negative means that the gap between R_2 and R_3 is less than $\rho/2$. We will evaluate it in section VI.

V. ANALYSIS

We analyze the security and performance in this section to show that D3 achieves strong security characteristics with relatively low overhead.

A. Security analysis

As shown in Table [II], we evaluate D3 against 6 prevalent attacks. In general, D3 could localize the faulty link that launches the packet corruption (drops, delays, modifies, or fabricates the packet) and path inconsistency (redirects the packet) attack. It is resilient to coward attack, framing attack, colluding attack, and replay attack. Finally, we explain why D3 does not create the opportunity for a new DoS attack.

Packet corruption and path inconsistency. If the adversary modifies or fabricates the packet header, the $fCounter$ of the next honest router will increase. Suppose the adversary delays or drops the packets, the $Counter$ in the downstream router will decrease. If the adversary launches the path inconsistency attack, the $Counter$ in the downstream router will decrease since the router in an unexpected domain will filter the packets. The source localizes the link with the highest reputation gap between two adjacent routers as faulty. Faultprints and RFL have the same security property as D3.

Coward attack. An adversary might launch the coward attack [18], e.g., only drop the packets it sampled since that other routers on the path do not sample them. The source could not localize the faulty link with the testimony from routers. But as all the lost packets are sampled by the adversary, it could localize the coward attack with the lost seq from the destination. Faultprints and RFL act the same characteristic as D3.

TABLE II
COMPARISON OF SECURITY WITH EXISTING MECHANISMS

	Packet corruption	Path inconsistency	Coward attack	Framing attack	Colluding attack	Replay attack
Faultprints	✓	✓	✓	(✓)	✓	(✓)
RFL	✓	✓	✓	(✓)	✓	✗
D3	✓	✓	✓	✓	✓	✓

Framing attack. The adversary might incriminate other routers by packet corruption or path inconsistency [5]. For packet corruption, assume that R_1 in Fig. 1 modifies the $mark_r$ of R_4, thus R_4 records the $fCounter$ information. But with the $binding\ marks$, R_4 will observe the framing when it receives a correct $mark_{src}$ and a wrong $mark_r$. R_1 had to modify the $mark_r$ and $mark_{src}$ simultaneously, but with the $blind\ policy$, R_1 could not know the $mark_{src}$ of R_4 since it does not have the dynamic key between the source and R_4. In Faultprints and RFL, the sampling policy is based on the chained $marks$ from the source to the current router. If R_1 changes the $mark$ for R_4, it only affects the routers after R_4. Then the source gives R_4 a low reputation. In fact, R_1 is the adversary.

For path inconsistency, R_1 in Fig. 1 changes the path information from R_4 to R_j, which is not the source anticipated. But as the path information is the partial input of the $mark_r$, R_2 and R_3 would filter the changed packet with the probability of $15/16$ and $255/256$, thus increasing $fCounter$ in R_2. The source would localize the link between R_1 and R_2 as a fault. Faultprints and RFL have the same characteristic as D3.

Colluding attack. Colluding adversaries might avoid localization by framing another router [5]. There are two situations of colluding adversaries: (1) non-adjacent colluders, e.g., two colluders R_i and R_j on the path, $j > i + 1$, (2) adjacent colluders, e.g., two colluders R_i and R_{i+1}.

For the case of non-adjacent colluders, as the adversaries do not know the dynamic key of a skipped honest router R_{i+1}, the first adversary R_i would be localized since the source could not get a correct reply from R_{i+1}. With adjacent colluders, for example, R_{i+1} and R_i share the dynamic key. If R_i corrupted a packet should be sampled by R_{i+1}, R_{i+1} would record the relative information. The source could localize the link between R_{i+1} and R_{i+2}. R_i evades the localization since R_{i+1} and R_i act like a virtual router R_{vc}. But the adversaries have little benefit since the source could first circumvent R_{i+1}, then localize R_i in the new epoch. D3 has the same characteristic compared with Faultprints and RFL.

Replay attack. Assume that an adversary launches a replay attack [19]. The destination could observe the misbehavior with seq and announce it in the ACK with a massive number of $fCounter$. Compared with Faultprints which utilize the timestamp in the packet header to defend against the replay attack, D3 is more practical since the secure time synchronization in the data plane is impractical with today's implementation. RFL does not use sequence numbers or timestamps to defend against the replay attack.

New DoS attack. The adversary could aggressively send packets to put a computational strain and storage overhead on each router or the user [5]. It would not hurt D3 since we minimize the computation overhead in the data plane and have constant storage in the data plane. The adversary could also aggressively send DAT_{SAM} and $probe$ to overwhelm the channel between the data plane and control plane. But the router only clones the packet header to the control plane after the verification of $mark_r$ and $mark_{src}$. The adversary had little chance to fabricate the correct $marks$ since it did not know the dynamic key. The only option of the adversary is that it aggressively sends false packets. It might overwhelm the channel between the data plane and the control plane. Each router could set a separate interface for the false packets report and stop increasing the $fCounter$ when bandwidth usage is higher than a threshold. It returns the maximum value (e.g., 65,535) of $fCounter$ to the source after receiving a $probe$. As the user could directly filter the fabricated packet with the $mark_{pkt}$, the adversary has no chance to overwhelm the user.

Moreover, adversaries might compromise the localization system from non-deployed domains. As the adversary could not get the dynamic key, D3 could filter the fabricated packet with the $mark_r$ in the packet header.

B. Performance analysis

As shown in Table [III], compared with the state-of-the-art mechanisms, D3 has a significantly lighter overhead in terms of storage, computation, communication overhead, and localization delay.

Storage overhead. In Faultprints and RFL, each router derives the dynamic key on the fly to keep 1 key (16 B) storage overhead. And they record the sampling packet with the Bloom Filter, which is a relatively sustainable way (468 MB with the bandwidth of 100 Gbps) to store the testimony in the data plane. D3 presets dynamic keys to keep a constant and low key storage overhead. The total number of ASes on the Internet is less than 80,000[2]. In the extreme case, a router needs to preset the dynamic keys for all the other ASes. With the key length of 16 B and each router presets the key for 2 time slots, the storage overhead is $80 * 10^3 * 2 * 16 = 2.56\ MB$.

D3 offloads the testimony storage overhead of the router to the control plane. The storage overhead in the control plane is also sustainable. We record the two counters (each counter with 2 B) in the table set, and a hash table sets up the data

[2]https://www.cidr-report.org/as2.0/

TABLE III
COMPARISON OF PERFORMANCE WITH EXISTING MECHANISMS

	Data plane storage overhead (B)[1]	Extra control packet number (5 hops)	Additional packet header(n hops)	No asymmetric cryptography	MAC operation	Localization delay (pkts)
Faultprints	$16 + 468 \times 10^6 \times BW$	1-4001	$(56 + 8 \times n)$ B	✗	2	4000
RFL	$16 + 468 \times 10^6 \times BW$	1-4001	$(38 + 4 \times n)$ B	✗	2	4000
D3	$32 \times M \times (2^K)$	1-11	20 B	✓	1	4000

[1] BW represents the bandwidth in metric of 100 Gbps, M represents the number of ASes, K represents the bit width of uid

index (calculated by the hash of the session information). Two epochs are enough since each router clears the counters of epoch $i-1$ at the end of epoch i, which takes 8 B for a session. Consider the load factor is 0.75. According to the CAIDA results, each router's mean flows per second are 39.73 K[3]. The total storage overhead of D3 in the control plane is $N = 39.73 * 10^3 * 8/0.75 = 424\ KB$.

Communication overhead. In Faultprints, RFL, and D3, the number of the extra control packet is only 1 ACK if end-to-end corruption ratio is lower than the threshold. In contrast, when the ratio is higher than the threshold, Faultprints and RFL need to process $O(n)$ $probe$ and $reply$. For the path length of 5, the sampling packets number is 2000 if each router samples 10% of the 4000 packets, and the number of extra control packets is 4001. D3 decreases these control packets from $O(n)$ to $O(1)$. The number of extra control packets is 11 if the source needs to get the $reply$ from each router.

The additional packet header in Faultprints and RFL is 96 B and 58 B with the path length of 5, respectively, and will grow with the path length increase. D3 has the constant additional packet header of 20 B and could embed the probabilistic and cryptographic sampling policy.

Computation overhead. Faultprints and RFL need to negotiate the dynamic keys with asymmetric encryption, which is prohibited expensive in the data plane. Moreover, they need to derive the dynamic key on the fly for each packet, which takes 1 extra MAC operation. In D3, each router only performs 1 MAC operation, significantly decreasing the computation overhead in the data plane.

Localization delay. We denote the localization delay as the number of packets in an epoch. The RTT in Internet peaks at 100 ms and 200 ms[4]. For a packet size of 1500 B, 4000 packets in an epoch make up 48 M bits. The forwarding delay is nearly 500 ms with an end-to-end bandwidth of 100 Mbps. One epoch is enough to receive the ACK and $reply$. Therefore, D3 utilizes 4000 packets in an epoch as default, the same as Faultprints and RFL.

VI. EVALUATION

A. Implementation

We implement D3 in the BMv2 environment [5] and Barefoot Tofino hardware. Our BMv2 testbed hosts in a virtual machine with Ubuntu 16.04, Intel Core i5-6200U CPU, 2.4 GHz, and 4 GB RAM. We instantiate the MAC operation with 2EM [10]. As the 2EM has been proven to be secure up to $2^{2n/3}$, we utilize $n = 64$ to keep the confidentiality. The processing of each router includes nearly 1000 lines $P4_{16}$ program in the data plane, which performs all the data plane operations for different kinds of packets, and nearly 500 lines of Python program in the control plane, including preparing the $reply$. We instantiate two terminals as the source and the destination user, implement the processing of different packets, and the fault localization with nearly 800 lines Python program. We also implement D3 in commodity Barefoot Tofino programmable switch S9180-32X, which includes nearly 900 lines $P4_{16}$ program with 2EM operation as MAC operation in hardware. Finally, we test the packet processing performance at the user.

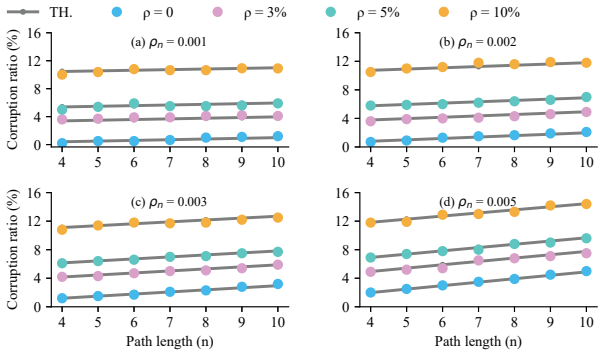

Fig. 4. The end-to-end corruption ratio with different parameters

B. Evaluation in BMv2

End-to-end corruption ratio. We evaluate the end-to-end corruption ratio with the path length from 4 to 10. Each result represents an average of 1000 runs. The source sends 4000 DAT in an epoch. An adversary corrupts packets with the corruption ratio of $\rho = 0, 3\%, 5\%, 10\%$. Fig. 4 shows the comparison of theoretical (TH) and measured end-to-end corruption ratio with different nature packet loss ($\rho_n = 0.001, 0.002, 0.003, 0.005$). The results show that the measured corruption ratio is nearly the same as the theoretical value since the destination records the $counter$ for all the packets (more than sampling packets).

[3]https://www.caida.org/catalog/datasets/trace_stats/
[4]https://www.caida.org/catalog/software/walrus/rtt/
[5]https://github.com/p4lang/behavioral-model

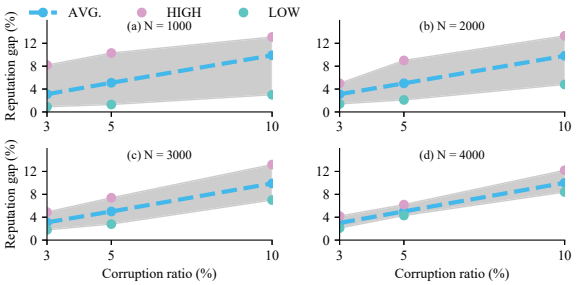

Fig. 5. The range of reputation gap with different parameters

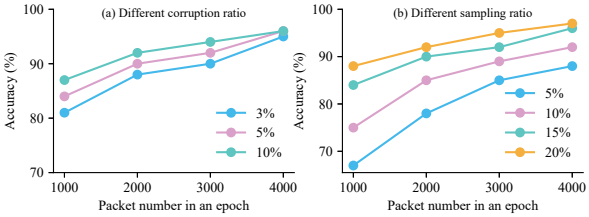

Fig. 6. The localization accuracy with different parameters

Reputation gap. We evaluate the reputation gap of the faulty link with the sampling ratio of 15%, the path length of 5, and $\rho_n = 0.001$. The source sends $N = 1000, 2000, 3000, 4000$ packets in each epoch. An adversary's corruption ratio ρ is 3%, 5%, and 10%, respectively. As shown in Fig. 5 (a), the low reputation gap is around 1.2% with 1000 packets in an epoch and the corruption ratio of 10%. This false negative decreases the localization accuracy. In Fig. 5 (b), (c), (d), with more than 3000 packets in an epoch, the range of reputation gap is as expected (most of the cases are within the range of $\rho \pm \rho/2$). With the corruption ratio of 5% and 4000 packets in an epoch, the lowest and highest reputation gap are 4.3% and 6.2%, while the expected value is within the range of $5\% \pm 2.5\%$.

Localization accuracy. Fig. 6 (a) shows the localization accuracy under different corruption ratios with the sampling ratio of 15%. It shows that with the corruption ratio of 10%, the accuracy is around 90%, 95% with 2000, 4000 packets in an epoch, respectively. It's around 80% when the corruption ratio is less than 5%, and the packet number in an epoch is 1000. It indicates that the low corruption ratio and packet number result in a reputation gap less than the threshold. The corruption ratio has little effect on the accuracy when the packet number is more than 2000. It testifies that D3 can achieve fault localization even with the adversary's corruption ratio of 3%. Fig. 6 (b) is the localization accuracy under different sampling ratios with corruption ratio of 5%. It shows that with 1000 packets in an epoch and a sampling ratio of 5%, the accuracy is only around 65%. It is around 85% with the packet number of 2000 and a sampling ratio of 10%. And it is more than 95% with 4000 packets and a sampling ratio of 15%. As the baseline of the state-of-the-art mechanisms (Faultprints and RFL) is to achieve a localization accuracy of 95% for 4000 packets in an epoch under the weaker attackers (5% corruption ratio), we can conclude that D3 achieves nearly the same localization accuracy with much less overhead.

Extra control packet overhead. The adversary can only corrupt the control packets (ACK, $reply$, $probe$) with the same ratio as DAT. The source sends another $probe$ toward a specific router that did not return the $reply$ after a timeout. We evaluate the number of $probes$ in different corruption ratios. The results show that under the corruption ratio of 3%, more than 98% of epochs return the $reply$ with one $probe$; under the corruption ratio of 5%, two $probes$ for each router could satisfy that more than 95% of epochs to get a $reply$; under the corruption ratio of 10%, two $probes$ could meet that more than 90% of epochs to get a $reply$. Considering each epoch has 4000 packets, and the control packet size is less than 100 B, assume that the average size of DAT is 1000 B, the extra communication overhead incurred by D3 is less than 0.06% for the path length of 5. In Faultprints and RFL, the corresponding overhead is 1.81%.

C. Evaluation in hardware

We implement 2EM [10] in S9180-32X as a MAC operation. And testify that the hardware could accomplish D3 within one pipeline. In contrast, other mechanisms have to achieve the operation with recirculation.

Computation overhead. The MAC operation in different mechanisms behaves the same on specific hardware. From this perspective, we evaluate the computation overhead in terms of the computation delay of relative MAC operation. In D3, each router performs 1 MAC operation. In contrast, Faultprints and RFL derive the dynamic keys with 1 MAC operation and then accomplish the processing with 1 MAC operation. As the hardware could not accomplish 2 MAC operations in one pipeline, we achieve 2 MAC operations with one recirculation.

The delay in different packet sizes is shown in Fig. (7), in which IP represents IPv6 packet forwarding without MAC operation, which is the baseline with the value from 0.27 µs (128 B) to 0.91 µs (1500 B), D3 has nearly the same computation delay when the packet size is more than 768 B, and slightly above the baseline when the packet size is less than 512 B. With a packet size of 128 B and 1500 B, the delay is 0.41 µs and 0.92 µs, respectively. In contrast, the delay in Faultprints/RFL is significantly higher than D3, which is 1.08 µs with the packet size of 128 B, and 1.69 µs with the packet size of 1500 B.

Throughput. We instantiate the MAC operation of D3 as a 2EM operation, which is compatible with a single packet-processing pipeline of the hardware. Thus, D3 can achieve localization at the hardware line rate [10] (e.g., up to 3.2 Tbps in S9180-32X). In contrast, in Faultprints and RFL, each packet had to recirculate to accomplish the processing. The throughput is less than the hardware's recirculation bandwidth bottleneck (The default recirculation bandwidth is 200 Gbps in S9180-32X).

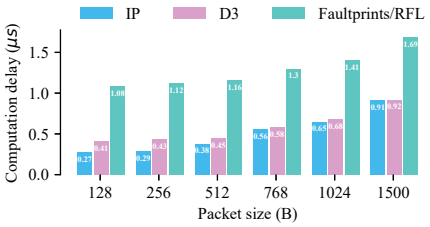

Fig. 7. The computation delay in Barefoot Tofino hardware

D. Performance at the user

In D3, the user requests path information and dynamic keys from the KDS. The additional latency is insignificant since the paths and keys information is available at the local KDS. Moreover, a user can cache both path and key information, eliminating extra latency for subsequent packets.

The source and destination need to perform the MAC operations for each packet, which comprises most computation overhead. We test the MAC operations with 2EM in the Intel Core i5-6200U CPU virtual machine. It could achieve more than 1,000,000 2EM operations in one second. The source needs a 2EM operation to calculate the $mark_{pkt}$, and n (the path length) 2EM operations to calculate the $marks$ for each transit router. In comparison, the destination needs to calculate the $mark_{pkt}$. With a path length of 4, we could learn that an ordinary commodity host could support more than 200,000 packets for the source and 1,000,000 for the destination, which is 2.4 Gbps and 12 Gbps for the packet size of 1500 Bytes. We can conclude that the computation overhead at the user is sustainable.

VII. Discussion

The length of $mark_r$. On Internet, the average AS path length is less than 5, and the vast majority of the length is at most 10^6. As the bit width of $mark_r$ is 32, we could utilize the $flag$ to indicate the dynamic length of $mark_r$. We use 4 bits in each router ($mark_i[0:4]$) as default to facilitate the implementation on hardware. If the path length i is more than 8, we could use 3 bits $mark_r$ to keep the ability that all the routers on the path could filter the malicious traffic.

Limitations. When the number of adversaries passes more than half of the path length, for example, assume that R_1, R_2, and R_3 in Fig. 1 launch the colluding attack, the source might not localize them. Consider that most AS are honest. An entity could localize the adversaries by establishing the AS reputation system based on many detectors' evaluations. In future work, we will aggregate the reputation from the detectors to establish a reputation system for each AS.

VIII. Related Work

Data plane MAC operation. Achieving MAC operation in the data plane is important for data forwarding to defend the strong adversary. The dedicated hardware achieves the MAC operation, but it needs upgrading hardware [20]. SPINE [21] achieves the MAC operation with SipHash in the BMv2 environment, but the hardware could not accomplish it within one single pipeline. P4-AES [9] requires at least four recirculations to accomplish a MAC operation, thus degrading forwarding performance. PINOT [10] implements the 2EM operation in a single pipeline on commercial hardware to encrypt the IP address. Based on PINOT, we utilize the dynamic keys, combined with the 2EM operation to calculate two 32 bits $marks$ in a single pipeline, then achieve the lightweight processing with the two $marks$.

Path availability and verification. The path-aware network such as SCION [2], gives transparency and choices to users, while SR [3] provides a practical way to program the network. These mechanisms provide the basis to build a path-aware network that allows the hosts to embed their policy in the packet header. When it comes to path verification, EPIC [20] utilizes highly efficient symmetric cryptographic operations in the forwarding process, while PPV [22] and MASK [23] improve the efficiency of the path verification with probabilistic packet marking. PSVM [24] presents a flexible path verification scheme in SDN. These mechanisms can only verify the forwarding path, and cannot localize the fault entities.

Fault localization. The mechanism such as AudIt [7] without any authentication of the packets suffers from packet modification attacks. DynaFL [25] proposes the secure neighborhood-based fault localization protocol in SDN. TrueNet [26] leverages trusted computing technology to build a trusted network-layer architecture. These mechanisms are difficult to apply on the Internet. ShortMAC [4] achieves high-security assurance even in the presence of strong adversaries. But it could only use in intra-domain scenarios since it records per-flow testimony in the data plane, thus having prohibitively heavy storage overhead in the data plane. Faultprints [5] is the first fault localization mechanism used in the inter-domain context based on the mechanism deriving a per-session dynamic key on the fly and recording the per-packet testimony with Bloom Filter. Based on the unreliable communication channel on today's Internet, RFL [6] utilizes a sophisticated way to build a fault localization mechanism. Unfortunately, the two mechanisms probe each sampling packet. It incurs the communication overhead, especially on the unreliable channel. In a word, these mechanisms did not take account of the router's performance, thus reducing the deployability. With the edge-based mechanism, D3 achieves high localization accuracy with lightweight overhead in the router.

IX. Conclusions

It's difficult to deploy the existing secure fault localization mechanisms in the edge cloud for the lack of efficiency in terms of storage, communication, and computation overhead. This paper designs D3, which offloads the router's overhead to the user and KDS. D3 has low storage overhead in the data plane and communication overhead in the network. With

[6]https://bgp.potaroo.net/as6447

the 2EM operation, it could accomplish all the MAC operations within a single pipeline on commodity hardware, thus significantly lowering computation overhead. Furthermore, it has strong security characteristics with the $blind\ policy$ and $binding\ marks$. With D3, the user could program the router on the path sophisticatedly and build a more reliable network to meet the user's requirements. It thus brings us closer to localizing the malicious ASes in the edge cloud.

ACKNOWLEDGMENT

This work was in part supported by the China National Funds for Distinguished Young Scientists with No. 61825204, the NSFC Project with No. 61932016, No. 62132011, and No. 62132009, the Beijing Outstanding Young Scientist Program with No. BJJWZYJH012019100030011. Ke Xu and Su Yao are the corresponding authors.

REFERENCES

[1] L. L. Peterson, T. E. Anderson, S. Katti, N. McKeown, G. M. Parulkar, J. Rexford, M. Satyanarayanan, M. O. Sunay, and A. Vahdat, "Democratizing the network edge," *Comput. Commun. Rev.*, vol. 49, no. 2, pp. 31–36, 2019.

[2] J. Kwon, J. A. García-Pardo, M. Legner, F. Wirz, M. Frei, D. Hausheer, and A. Perrig, "SCIONLAB: A next-generation internet testbed," in *28th IEEE International Conference on Network Protocols, ICNP 2020, Madrid, Spain, October 13-16, 2020*, pp. 1–12, IEEE, 2020.

[3] R. Hartert, S. Vissicchio, P. Schaus, O. Bonaventure, C. Filsfils, T. Telkamp, and P. François, "A declarative and expressive approach to control forwarding paths in carrier-grade networks," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication, SIGCOMM 2015, London, United Kingdom, August 17-21, 2015* (S. Uhlig, O. Maennel, B. Karp, and J. Padhye, eds.), pp. 15–28, ACM, 2015.

[4] X. Zhang, Z. Zhou, H. Hsiao, T. H. Kim, A. Perrig, and P. Tague, "Shortmac: Efficient data-plane fault localization," in *19th Annual Network and Distributed System Security Symposium, NDSS 2012, San Diego, California, USA, February 5-8, 2012*, The Internet Society, 2012.

[5] C. Basescu, Y. Lin, H. Zhang, and A. Perrig, "High-speed inter-domain fault localization," in *IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 22-26, 2016*, pp. 859–877, IEEE Computer Society, 2016.

[6] B. Wu, K. Xu, Q. Li, B. Liu, S. Ren, F. Yang, M. Shen, and K. Ren, "RFL: robust fault localization on unreliable communication channels," *Comput. Networks*, vol. 158, pp. 158–174, 2019.

[7] K. J. Argyraki, P. Maniatis, O. Irzak, S. Ashish, and S. Shenker, "Loss and delay accountability for the internet," in *Proceedings of the IEEE International Conference on Network Protocols, ICNP 2007, October 16-19, 2007, Beijing, China*, pp. 194–205, IEEE Computer Society, 2007.

[8] B. Barak, S. Goldberg, and D. Xiao, "Protocols and lower bounds for failure localization in the internet," in *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings* (N. P. Smart, ed.), vol. 4965 of *Lecture Notes in Computer Science*, pp. 341–360, Springer, 2008.

[9] X. Chen, "Implementing AES encryption on programmable switches via scrambled lookup tables," in *Proceedings of the 2020 ACM SIGCOMM 2020 Workshop on Secure Programmable Network Infrastructure, SPIN@SIGCOMM 2020, Virtual Event, USA, August 14, 2020* (A. Chen and L. Vanbever, eds.), pp. 8–14, ACM, 2020.

[10] L. Wang, H. Kim, P. Mittal, and J. Rexford, "Programmable in-network obfuscation of traffic," *CoRR*, vol. abs/2006.00097, 2020.

[11] M. Zhang, G. Li, S. Wang, C. Liu, A. Chen, H. Hu, G. Gu, Q. Li, M. Xu, and J. Wu, "Poseidon: Mitigating volumetric ddos attacks with programmable switches," in *27th Annual Network and Distributed System Security Symposium, NDSS 2020, San Diego, California, USA, February 23-26, 2020*, The Internet Society, 2020.

[12] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, and D. Walker, "P4: programming protocol-independent packet processors," *Comput. Commun. Rev.*, vol. 44, no. 3, pp. 87–95, 2014.

[13] J. Meng, H. Tan, C. Xu, W. Cao, L. Liu, and B. Li, "Dedas: Online task dispatching and scheduling with bandwidth constraint in edge computing," in *2019 IEEE Conference on Computer Communications, INFOCOM 2019, Paris, France, April 29 - May 2, 2019*, pp. 2287–2295, IEEE, 2019.

[14] F. Zhang, L. Jia, C. Basescu, T. H. Kim, Y. Hu, and A. Perrig, "Mechanized network origin and path authenticity proofs," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014* (G. Ahn, M. Yung, and N. Li, eds.), pp. 346–357, ACM, 2014.

[15] J. Wu, J. Bi, M. Bagnulo, F. Baker, and C. Vogt, "Source address validation improvement (SAVI) framework," *RFC*, vol. 7039, pp. 1–14, 2013.

[16] J. Wu, G. Ren, and X. Li, "Source address validation: Architecture and protocol design," in *Proceedings of the IEEE International Conference on Network Protocols, ICNP 2007, October 16-19, 2007, Beijing, China*, pp. 276–283, IEEE Computer Society, 2007.

[17] S. E. Deering and R. M. Hinden, "Internet protocol, version 6 (ipv6) specification," *RFC*, vol. 8200, pp. 1–42, 2017.

[18] B. Liu, J. T. Chiang, J. J. Haas, and Y. Hu, "Coward attacks in vehicular networks," *ACM SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 14, no. 3, pp. 34–36, 2010.

[19] P. F. Syverson, "A taxonomy of replay attacks," in *Seventh IEEE Computer Security Foundations Workshop - CSFW'94, Franconia, New Hampshire, USA, June 14-16, 1994, Proceedings*, pp. 187–191, IEEE Computer Society, 1994.

[20] M. Legner, T. Klenze, M. Wyss, C. Sprenger, and A. Perrig, "EPIC: every packet is checked in the data plane of a path-aware internet," in *29th USENIX Security Symposium, USENIX Security 2020, August 12-14, 2020* (S. Capkun and F. Roesner, eds.), pp. 541–558, USENIX Association, 2020.

[21] T. Datta, N. Feamster, J. Rexford, and L. Wang, "SPINE: surveillance protection in the network elements," in *9th USENIX Workshop on Free and Open Communications on the Internet, FOCI 2019, Santa Clara, CA, USA, August 13, 2019* (S. E. McGregor and M. C. Tschantz, eds.), USENIX Association, 2019.

[22] B. Wu, K. Xu, Q. Li, Z. Liu, Y. Hu, M. J. Reed, M. Shen, and F. Yang, "Enabling efficient source and path verification via probabilistic packet marking," in *26th IEEE/ACM International Symposium on Quality of Service, IWQoS 2018, Banff, AB, Canada, June 4-6, 2018*, pp. 1–10, IEEE, 2018.

[23] S. Fu, K. Xu, Q. Li, X. Wang, S. Yao, Y. Guo, and X. Du, "MASK: practical source and path verification based on Multi-AS-Key," in *2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS) (IWQoS 2021)*, (Tokyo, Japan), June 2021.

[24] F. Yang, K. Xu, Q. Li, R. Lu, B. Wu, T. Zhang, Y. Zhao, and M. Shen, "I know if the journey changes: Flexible source and path validation," in *28th IEEE/ACM International Symposium on Quality of Service, IWQoS 2020, Hangzhou, China, June 15-17, 2020*, pp. 1–6, IEEE, 2020.

[25] X. Zhang, C. Lan, and A. Perrig, "Secure and scalable fault localization under dynamic traffic patterns," in *IEEE Symposium on Security and Privacy, SP 2012, 21-23 May 2012, San Francisco, California, USA*, pp. 317–331, IEEE Computer Society, 2012.

[26] X. Zhang, Z. Zhou, G. Hasker, A. Perrig, and V. D. Gligor, "Network fault localization with small TCB," in *Proceedings of the 19th annual IEEE International Conference on Network Protocols, ICNP 2011, Vancouver, BC, Canada, October 17-20, 2011*, pp. 143–154, IEEE Computer Society, 2011.