MASK: Practical Source and Path Verification Based on Multi-AS-Key

Songtao Fu, Qi Li, *Senior Member, IEEE*, Min Zhu, Xiaoliang Wang, Su Yao, Yangfei Guo, Xinle Du, and Ke Xu, *Senior Member, IEEE, Member, ACM*

***Abstract*— The source and path verification in Path-Aware Networking considers the two critical issues: (1) end hosts could verify that the network follows their forwarding decisions, and (2) both on-path routers and destination host could authenticate the source of packets and filter the malicious traffic. Unfortunately, the state-of-the-art mechanisms require heavy communication overhead in the network and computation overhead in the router; moreover, it is difficult to meet the dynamic requirements of the end host. We propose a user-driven mechanism, source and path verification based on Multi-AS-Key (MASK). MASK decreases the communication overhead by a short additional packet header and reduces the computation overhead by separating the control and data plane in terms of the cryptographic operation. Furthermore, it utilizes the stateful user to instruct the stateless routers to process the packet with a user-driven policy, thus satisfying the user's requirements such as detecting the packet drop and replay attack. With the plausible design, the communication overhead for realistic path lengths is 1/2 to 1/10 compared with the state-of-the-art mechanisms. We implement MASK in the BMv2 environment and commodity Barefoot Tofino programmable switch, testify that MASK introduces significantly less overhead than the state-of-the-art mechanisms, and demonstrate that MASK could achieve the verification in the programmable switch at line rate.**

***Index Terms*— Path-aware networking, data plane, source and path verification.**

Manuscript received 9 November 2021; revised 5 July 2022; accepted 15 October 2022; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor Y. Liu. This work was supported in part by the National Key Research and Development Program of China under Grant 2022YFB3102301; in part by the China National Funds for Distinguished Young Scientists under Grant 61825204; in part by the NSFC Project under Grant 61932016, Grant 62132011, and Grant 62132009; and in part by the Beijing Outstanding Young Scientist Program under Grant BJJWZYJH01201910003011. *(Corresponding authors: Ke Xu; Su Yao.)*

Songtao Fu, Min Zhu, and Xinle Du are with the Department of Computer Science and Technology, Beijing National Research Center for Information Science and Technology (BNRist), Tsinghua University, Beijing 100084, China (e-mail: fust18@tsinghua.edu.cn; minzhu@tsinghua.edu.cn; dxl18@tsinghua.edu.cn).

Qi Li is with the Institute for Network Sciences and Cyberspace, Beijing National Research Center for Information Science and Technology (BNRist), Tsinghua University, Beijing 100084, China, and also with the Zhongguancun Laboratory, Beijing 100094, China (e-mail: qli01@tsinghua.edu.cn).

Xiaoliang Wang is with the Information Engineering College, Capital Normal University, Beijing 100089, China (e-mail: wangxiaoliang@cnu.edu.cn).

Su Yao is with the Beijing National Research Center for Information Science and Technology (BNRist), Tsinghua University, Beijing 100084, China (e-mail: yaosu@tsinghua.edu.cn).

Yangfei Guo is with the Zhongguancun Laboratory, Beijing 100094, China (e-mail: guoyangfei@zgclab.edu.cn).

Ke Xu is with the Department of Computer Science and Technology, Beijing National Research Center for Information Science and Technology (BNRist), Tsinghua University, Beijing 100084, China, and also with the Zhongguancun Laboratory, Beijing 100094, China (e-mail: xuke@tsinghua.edu.cn).

Digital Object Identifier 10.1109/TNET.2022.3222610

I. INTRODUCTION

NETWORK attack threats on Internet have been increasing at an alarming rate in recent years. Flawed designs of network protocols lead to many security issues [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14]. In particular, packet forwarding anomalies become the major threat on the Internet [15]. Therefore, it is necessary to design the secure data plane to ensure correctness of packet forwarding on the Internet [16].

Path-Aware Networking (PAN) [17] has been developed to address this issue above. It allows end hosts to enforce their forwarding policies on the data plane of the Internet to ensure the correctness of the forwarding decisions. It thus provides an opportunity for establishing a secure data plane in the network layer. Based on PAN, the source and path verification in the data plane could fundamentally improve the network's trustworthiness, which achieves two fundamental security properties: (1) The end hosts (the source and the destination users) could verify that the network follows their forwarding decisions. (2) The on-path routers and destination host could authenticate the packets' source while filtering the malicious traffic as early as possible.

To achieve source and path verification, the source, on-path routers, and destination should calculate a Message Authentication Code (MAC)[1] with the dynamic key shared between them [15]. The granularity of the verification is at the AS level since that an AS is a fate-sharing unit [18]. Existing verification mechanisms have been proposed to empower the users to achieve the verification with a strong security guarantee. However, the following deficiencies significantly impact the practicality of these state-of-the-art verification mechanisms:

Heavy communication overhead. The mechanisms, such as ICING [19], OPT [15], and EPIC [20], which verify the packet in each on-path router, incur heavy communication overhead in the network by the long additional packet header. The probabilistic mechanism (e.g., PPV [21]) only updates the $mark$ in partial on-path routers. It decreases the communication overhead. But it also degrades the security performance, e.g., cannot filter the malicious traffic in routers.

[1]We name the MAC as $mark$ in this paper.

Heavy computation overhead. The mechanism utilizes asymmetric encryption to share the dynamic keys between the end hosts and routers (e.g., DRKey [15]), which is impractical with today's hardware. EPIC decreases this computation overhead with a practical hierarchical key derivation mechanism [22]. But the on-path routers need to recreate dynamic keys on the fly, and the symmetric cryptographic operations with a strong security guarantee (e.g., AES) in the data plane affect the line rate performance to a large extent.

Difficulty in meeting user requirements. Some users need to meet their requirements by detecting the packet drop or replay attack. The state-of-the-art mechanisms cannot detect the packet drop attack since they only check the received packets, and cannot verify the forwarding history according to the policy of a session (e.g., the ratio of the successful packet forwarding over a period). Meanwhile, some mechanisms utilize the timestamp to resist the replay attack, which needs synchronization between the routers and end hosts [23]. It is also impractical to keep the precise data plane synchronization between these entities in an inter-domain context.

High deployment barriers. The Internet is a network of networks connecting billions of hosts in more than 70,000 ASes [24], and the current Internet architecture is firmly entrenched [25]. That means a backward-compatible verification mechanism is necessary. The legacy routers need to communicate with updated routers and vice versa. The state-of-the-art mechanisms, which utilize the $mark$ with a strong security guarantee (e.g., AES), take more computation resources in the router. It allows legacy users to overwhelm the upgraded router by sending a huge number of false packets that need to be verified. On the other hand, a practical verification mechanism should benefit different PANs, e.g., to decrease their source and path verification overhead (e.g., Nebula [26]), or benefit the networks which have no verification mechanism (e.g., SRv6 [27]).

This paper proposes MASK, a practical source and path verification with an efficient additional packet header and lightweight overhead in the router. The design of MASK is based on two insights: One is the observation that a router should calculate at least one $mark$ to prove it had processed a packet. The other is that the MAC operation with a strong security guarantee (e.g., AES) is unnecessary in the data plane since the $mark$ is ephemeral (the duration of thousands of packets). The main contributions of this paper are as follows:

- We utilize a probabilistic marking mechanism based on a specific policy to keep a short and constant additional packet header, thus having a low communication overhead. With the policy shared between the source and destination, MASK keeps the ability to filter the malicious traffic in on-path routers.
- We reduce the computation overhead of the router by separating the control and data plane in terms of the cryptographic operation. In the control plane, the router prepares the dynamic keys calculated by a MAC operation with a strong security guarantee (e.g., AES); in the data plane, the on-path router only needs to perform 1 lightweight MAC operation to derive an ephemeral $mark$.
- We design a user-driven mechanism that the stateful users could instruct and verify the processing at stateless on-path routers. The source and destination users share the per-session policy without inter-domain negotiation. It could facilitate the destination to verify each packet with the policy-based marking. Then it could detect the packet drop and replay attacks, which could not or are hard to be achieved by the per-packet verification.
- We implement MASK with different MAC operations in the BMv2 environment and the Barefoot Tofino programmable switch and compare MASK with the state-of-the-art mechanisms. The results show that the communication overhead is 1/2 to 1/10 compared with state-of-the-art mechanisms. The implementation of MASK in a programmable switch demonstrates that MASK could achieve the verification at a line rate.

The rest of the paper organizes as follows: In the next section, we analyze the adversary model and our design goals. Sec. III presents how can we achieve the verification at a high level, Sec. IV details the MASK protocol, Sec. V analyzes the security and performance characteristics, Sec. VI presents the implementation and the evaluation, Sec. VII discusses the incremental deployment mechanism as well as the limitations. Sec. VIII describes related works. We conclude the paper in the last section.

II. Problem Statement

We first present the adversary model and the background of source and path verification on the Internet. Then we explain our design goals.

A. Adversary Model and Background

The source and path verification focuses on secure forwarding in the data plane, considering a Dolev-Yao adversary could observe, drop, inject, replay, or alter packets in a compromised network [28]. We summarize various types of attacks as follows:

Packet damage. It means that an adversary might damage any part of the packet, e.g., the source address in the packet header, or delay, drop a packet, inject a fabricate packet toward the destination.

Path change. An adversary might forward the packet along the arbitrary alternative path not the same as the expected path (selected by the source). Assume the expected path is (R_1, R_2, R_3, R_4) in Fig. 1, the path of (R_1, R_3, R_4) represents the skipping path, the path $(R_1, R_2, R_6, R_3, R_4)$ represents the detoured path, and the path (R_1, R_3, R_2, R_4) represents the disordered path. It might decrease the forwarding performance or redirect the traffic to the unexpected AS that the users (source and destination) do not want to transmit.

Replay attack. An adversary might replay a legitimate packet to deceive the routers or the destination, which wastes the bandwidth resources or causes more severe security issues.

Colluding attack. More than one adversary might collude to frame other entities or evade the verification. For example, R_2 and R_3 in Fig. 1 might collude to redirect the traffic to another AS (R_6) not on the expected path.

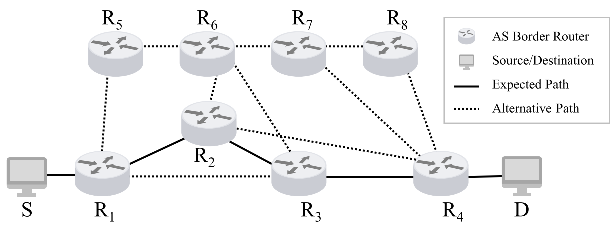

Fig. 1. The example of path change attacks in network.

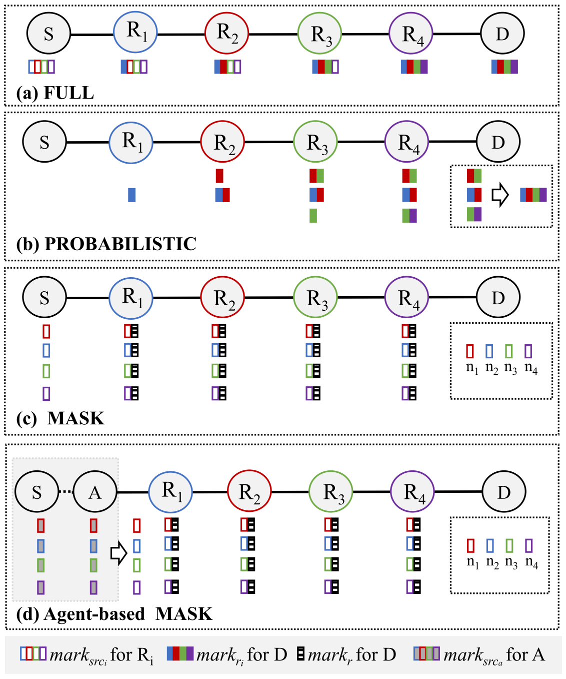

Fig. 2. The overview of different verification mechanisms. For the path length of n, the FULL mechanism has at least n $marks$ for on-path routers (R_i) in each packet, while the PROBABILISTIC has 2 $marks$ no matter the path length, MASK has 2 $marks$ and could verify the packet number at each end of an epoch. Agent-based MASK is nearly the same as MASK except that R_i only presets the dynamic keys for the agent (A).

Packet flooding. An adversary might exhaust the router's computation or storage resource by packet flooding if the verification incurs the router's storage and computation overhead.

Therefore, a secure source and path verification mechanism based on the dynamic key sharing between the end hosts and the routers is critical. The state-of-the-art secure source and path verification mechanisms include FULL and PROBABILISTIC processing.

As shown in Fig. 2 (a), the router and the destination in the FULL operation verify each packet with at least one $mark$. ICING [19], OPT [15], and EPIC [20] have the greatest impact. In ICING, each router calculates both a hash and a $mark$ over the payload and in addition performs n symmetric cryptographic operations (one for each router), which is two orders of magnitude slower than EPIC. To design a practical verification mechanism, we only compare with the lightweight counterparts (OPT and EPIC) in this paper. The $mark_{src_i}$ provides source authentication to a single on-path router, and $mark_{r_i}$ is the $mark$ updated by the R_i and verified by the destination. Besides that, we call the $mark$ for the end-to-end authentication as $mark_{pkt}$, which is generated by the source and verified by the destination. We omit it in Fig. 2 since there is one $mark_{pkt}$ for each packet in the three types of mechanisms.

As shown in Fig. 2 (b), the state-of-the-art PROBABILISTIC mechanism [21] verifies the packet based on a session. Two adjacent on-path routers probabilistically updates the $mark_{r_i}$. After receiving a certain number of packets (e.g., 3 packets in Fig. 2 (b)), the destination reconstructs the actual path based on the $marks$.

B. Design Goals and Assumptions

To achieve a practical source and path verification, the design goals of MASK are achieved as following:

Low communication and computation overhead. To achieve low communication overhead in the network, as shown in Fig. 2 (c), MASK utilizes the probabilistic $mark_{src_i}$ and accumulated $mark_r$ to keep the constant additional packet header. To achieve low computation overhead in the router, each router presets dynamic key from the control plane.

Satisfaction of user requirements. The source and the destination share a per-session policy to support the verification. As shown in Fig. 2 (c), assuming the policy is each on-path router verifies n_1, n_2, n_3, n_4 packets respectively, the destination could verify the number of packets based on the received packets. This user-driven mechanism empowers the destination to detect packet drop attack and resist the replay attack without the synchronization in the data plane.

Deployability. The emerging programmable switch [7], which supports packet processing using domain specific languages (e.g., P4 [29]), provides an exciting opportunity to design protocols that manipulate arbitrary parts of packet header at line rate [30]. It still has some challenges to utilize the programmable switch since the ASIC constraints, e.g., the P4 language did not support the heavy computation [31]. However, MASK's design is devoted to addressing these challenges by minimizing the verification's operation.

To preset the dynamic keys in the router, we begin with the following assumptions:

Hierarchical key establishment. We leverage the Key Distribution Server (KDS) to establish the two levels key system. The AS-Key (AK) is the top-level key infrequently renewed (e.g., days), which is only shared among the KDSes [32]. The KDS could derive the router's dynamic key (RDK) and the host's dynamic key (HDK). We will detail how to derive dynamic keys (second-level key) based on the Multi-AS-Key in section IV-A.

Secure intra-AS communication. The end host (source and destination) and the router could get the relative keys from the local KDS. We assume a secure channel to communicate between a host (router) and KDS in local AS to get the keys. In particular, an AS may prevent source address spoofing within its network using a specific anti-spoofing method [33], [34].

III. Overview

To achieve the verification, we design the additional MASK header. We could load the MASK header in a specific PAN packet header. For example, in SRv6, we could load the

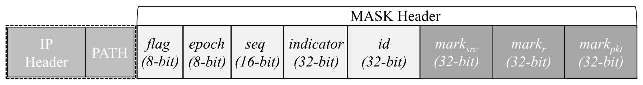

Fig. 3. The packet header of MASK.

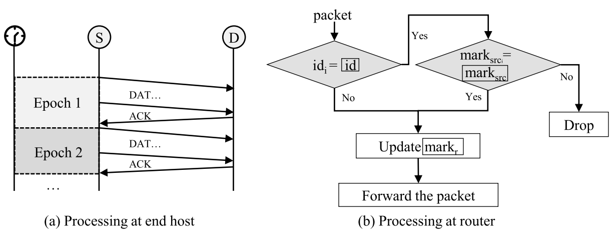
S
D
Epoch 1
Epoch 2
DAT...
ACK
DAT...
ACK
...
(a) Processing at end host
packet
id$_i$ = $\boxed{\text{id}}$
Yes
No
mark$_{src_i}$ = $\boxed{\text{mark}_{src}}$
Yes
No
Drop
Update $\boxed{\text{mark}_r}$
Forward the packet
(b) Processing at router

Fig. 4. The MASK overview.

MASK header as the IPv6 Extension Header [35] as Fig. 3; while in SCION, we could embed the MASK header as the $\boxed{\text{VALHD}}$ in EPIC. In the MASK header, $flag$ is the flag field, $epoch$ and seq are the epoch number and sequence number of a packet respectively, $indicator$ includes 16-bit time slot information (ts) and 16-bit user identifier (uid) to indicate which RDK should be selected, id is the identifier of an on-path router selected by the source to verify this packet. Besides that, there are 3 $marks$ in the MASK header:

- $mark_{src}$: The $mark$ calculated by the source and verified by the selected router.
- $mark_r$: The $mark$ updated by each on-path router.
- $mark_{pkt}$: The $mark$ calculated by the source and verified by the destination.

At a high level, MASK offloads most of the router's overhead to the end host. As shown in Fig. 4, MASK includes the processing at the end host and router.

Processing at end host. As shown in Fig. 4 (a), the source (S) requires the dynamic keys from the KDS in the local AS. It sends data packets (DAT) towards the destination, nests the $\boxed{epoch}$[2] and $\boxed{seq}$, with the $\boxed{id}$ (an on-path router's id_i) determined by the policy shared with the destination, as well as the $\boxed{mark_{src}}$ and $\boxed{mark_{pkt}}$ in the packet header. The destination (D) requires the dynamic keys from the KDS after receiving the first packet and achieves the verification of each packet. At the end of each epoch, the destination validates the processing of this epoch according to the policy, then sends an acknowledgment packet (ACK) to the source.

Processing at router: As shown in Fig. 4 (b), after receiving a packet, a router with the same id_i verifies the $\boxed{mark_{src}}$ to achieve the source verification. It drops the packet which fails the source verification. Then it updates $\boxed{mark_r}$ and forwards the packet to the next hop.

IV. Protocol Description

We first introduce the dynamic key between the user and the on-path routers, then describe the processing of MASK.

[2]In this paper, $\boxed{\text{FIELD}}$ represents the FIELD in the packet header.

A. Deriving the Dynamic Key

The KDS derives the dynamic keys from the AK between two ASes. The end hosts and the routers could require the relative keys from the KDS in the local AS.

We utilize the IP address of the end hosts, the path information, and the timestamp to represent a session. As shown in Equation (1):

$$session = H(IP_{src}||IP_{dst}||id_1||\cdots||id_n||TS) \qquad (1)$$

In which IP_{src} and IP_{dst} are the IP addresses of the source and the destination, id_n represents the identifier of an on-path router, TS defines the time slot of the beginning of the session. All these fields are from the IP, PATH, and MASK header. "||" represents the concatenation operation, and $H(.)$ represents a specific hash function (a non-secure hash function is enough).

The dynamic keys include the end host's HDK and the router's RDK_{si}. The HDK is calculated as Equation (2):

$$HDK = MAC_{AK_{sd}}(session) \qquad (2)$$

In which AK_{sd} is the AK between the source AS and the destination AS. As a user-driven verification mechanism, the HDK derived from a cryptographic operation with strong security (e.g., AES), guarantees that an adversary could only guess the dynamic key with negligible probability.

We do not utilize the on-path routers recreates RDK on the fly since it incurs the heavy computation overhead to achieve the MAC operation with a strong security guarantee [36]. For example, to achieve the AES operation, the recirculation bandwidth and the connection between the data plane and dedicated extern encryption hardware could be the bottleneck of the forwarding [37]. However, presetting the per-session dynamic keys incurs each router's heavy storage overhead. We utilize a time-slot-based dynamic key deriving mechanism, ensuring the confidentiality of RDK with no computation overhead and constant storage overhead. The RDK_{si} utilized between the source and a specific router i is calculated as Equation (3):

$$RDK_{si} = MAC_{AK_{si}}(id_i||indicator) \qquad (3)$$

In which AK_{si} is the AK between the source AS and the AS of router i. id_i is on-path router i's id. In practice, an AS could use one id (as virtual id) to decrease the scale of the dynamic key. The $indicator$ includes 16-bit time slot ts and 16-bit user identifier uid. We assume that the KDS distributes AK_{si} to the border routers, and each router derives the RDK_{si} of M (more than one) slots in the control plane. The user could require the on-path router's RDK_{si} of M slots from the local KDS. Then it could utilize the $indicator$ to instruct each router to select the relative RDK_{si}.

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

We will detail in Sec. VI that the length of a dynamic key is 24 B (192-bit). Suppose there are 70,000 ASes, each AS has N uid, and each router presets the key of 2 time slots, the storage overhead in an on-path router is $70 * 10^3 * 2 * 24 * N = 3.36 * N\ MB$. The number of uid affects the dynamic key storage overhead in the router. That means each on-path router only stores the dynamic keys for a certain number of uid. We utilize two measures to keep low and constant storage overhead.

Firstly, the users in a group could use the same uid. For example, the servers in the edge cloud communicate with the remote cloud servers [38]. In this case, the servers controlled by a certain company could utilize the same uid. It is sustainable for several important users.

Secondly, one uid is for a subnet instead of an end host. The border router of a subnet as an agent. The agent could manage the symmetric key for each end host in the same subnet. With the symmetric key between the end host and the agent, the end host sends the MASK packet to the agent. After verifying the packet, the agent operates the packet header with the dynamic keys between itself and the on-path routers. The on-path routers and the destination eventually achieve the verification. Compared with MASK, the difference is the agent first verifies the source, then calculates the $marks$ for the on-path routers according to the packet header.

With this design, an AS could customize its dynamic key policy with other ASes, including the number of uid and the step size of ts. Assuming each AS has 16 agents, the storage overhead is at most 53.76 MB.

B. Processing at End Host

As shown in Equation (4), the source in PAN first plans the $PATH$ information, then it gets the RDK_{path} of routers, and the HDK from the KDS.

$$\begin{aligned} PATH &= (id_1, id_2, \ldots, id_n) \\ RDK_{path} &= (RDK_{s1}, RDK_{s2}, \ldots, RDK_{sn}) \end{aligned} \tag{4}$$

The source first determines the policy of the verification. The policy includes the number of packets in a specific epoch and the probability of each hop to verify a packet. We utilize the uniform probability as default, which means each on-path router has the probability of $1/n$ (n is the path length) to verify a packet. When it comes to the number of packets in a specific epoch, the source could denote the default number, e.g., 4096, then calculate the adjustment factor with the HDK and the $epoch$:

$$\lambda = MAC_{HDK}(epoch)\ MOD\ K \tag{5}$$

In which K represents an integer, assuming K is 4096, the value range of λ is from 0 to 4095. Then the number of packets in an epoch is from 4096 to 8192. The number in different epochs constitutes the Checkpoint Sequence Hopping (CSH). The $CSH[M]$ is an array which includes M sequences for M epochs, e.g., $CSH[M] = \{4500, 5001, \ldots, CSH[m], \ldots\}$ it means that there are 4500 packets in epoch 0, 5001 packets in epoch 1, and $CSH[m]$ packets in epoch m. It is worth mentioning that there is 8-bit $epoch$ in the packet header, but the source and the destination could utilize arbitrarily length of $epoch$ in equation (5). The destination could also calculate λ with Equation (5). With this design, the source and the destination share the same CSH without inter-domain negotiation.

With a specific policy, in each epoch, the source processes the packet as Alg. 1. In which $\leftarrow$ is an initialization or assignment.

Algorithm 1 Procedure at Source Host

```
input : $PATH, RDK_{path}, HDK, CSH[M]$
1 for $CSH[m]$ in $CSH[M]$ do
2     while $CSH[m] >= 0$ do
3         embed the $\boxed{flag}$ $\boxed{epoch}$ $\boxed{seq}$
4         determine $id_i$ and calculate $marks$
5         $\boxed{\text{id}} \leftarrow id_i$, $\boxed{mark_{src}} \leftarrow mark_{src_i}$
6         $\boxed{mark_r} \leftarrow mark_r$, $\boxed{mark_{pkt}} \leftarrow mark_{pkt}$
7         $CSH[m] \leftarrow CSH[m] - 1$
8         send packet to the first hop
```

In Alg. 1, the source has the $PATH$, RDK_{path}, HDK, and calculates the $CSH[M]$ for each epoch as Equation (5). It selects $CSH[m]$ from $CSH[M]$ (e.g., 5000) in each epoch (line 1). Then it embeds the $\boxed{flag}$ (the default value is 0), $\boxed{epoch}$ ($m\ MOD$ 256) and $\boxed{seq}$ ($CSH[m]$) in the packet header (line 2-3). After that, it selects one on-path router according to the marking probability (e.g., uniform marking) and calculates the $\boxed{marks}$ as Equation (7), (8), (9), then embeds them into the packet header (line 4-6). In Equation (6), the CST_{pkt} represents the constant packet header. It includes source IP address, $\boxed{PATH}$, $\boxed{flag}$, $\boxed{epoch}$, $\boxed{seq}$, and $\boxed{id}$, id_i and id_{i-1} are the link information. The $AUTH$ is the path authorization of a specific PAN, e.g., the hop authenticator (in SCION), or HMAC (in SRv6). In Equation (7), [0:32] represents that we truncate the lowest 32-bit of the $mark_i$.

$$\begin{aligned} mark_i &= MAC_{RDK_{si}}(CST_{pkt}||id_{i-1}||id_i||AUTH) \quad (6) \\ mark_{src_i} &= mark_i[0:32] \end{aligned} \tag{7}$$

In Equation (8), $hash$ is a 32-bit string calculated by the source, which is the hash of the upper-layer payload, such as the TCP sequence number and (or) the packet payload. It could facilitate the destination to verify the packet integrity. The source performs the XOR operation with each $mark_i$'s highest 32-bit string. Each on-path router will perform the XOR operation hop by hop, then at the destination, $mark_r = hash$, if all the routers perform as the source expected.

$$mark_r = hash \oplus mark_1[32:64] \cdots \oplus mark_n[32:64] \tag{8}$$

The purpose of the $mark$ is to defend against the adversary evading the verification. From this perspective, EPIC utilized a short $mark$ (24-bit) to improve the goodput. With the 32-bit $mark$ in MASK, an attacker has to send at most $2^{32} \approx 4.29 \times 10^9$ probe packets to find a correct $mark$ of a packet, which takes around 40 minutes on a gigabit link.

As each packet has its unique $mark$, it is enough to guarantee the security of verification. The reason we generate 64-bit $mark$ is that the on-path router can truncate this 64-bit $mark$ for the source authentication and path validation. With the truncation, each on-path router only needs to perform 1 MAC operation, which improves the practicality. We have detailed that the $mark$ in the data plane is an ephemeral $mark$, and in Sec. VI we will testify that the Two-round Even-Mansour (2EM) [36] operation could satisfy the security requirement.

The source calculates the $mark_{pkt}$ as Equation (9). With $\boxed{mark_r}$ in the destination's received packet header, the destination could verify the processing at each hop.

$$mark_{pkt} = MAC_{HDK}(CST_{pkt}||mark_r)[0:32] \qquad (9)$$

In MASK, we utilize the XOR operation and one $mark_r$ to load the $mark_{r_i}$ instead of the onion $mark_{r_i}$ for each on-path router. In practice, the XOR operation of $mark_r$ has the same security guarantee as the onion $mark_r$ in terms of path verification. The reasons are as follows: Firstly, considering a single adversary launching the path change attack, it has to alter the $\boxed{PATH}$. It could not evade the verification since the honest downstream router will calculate the wrong $mark_i$ with the wrong $\boxed{PATH}$, e.g., in Fig. (1), the adversary changes the path from "$R_1 - R_2 - R_3$" to "$R_1 - R_3 - R_2$". If R_2 and (or) R_3 are honest, the $mark_i$ of R_2 and (or) R_3 will wrong. Secondly, for the colluding attack, the colluders can exchange their information. If R_2 and R_3 are colluders, even the onion $mark_r$ could not verify the disorder since the colluders (R_2 and R_3) could calculate the $mark_r$ with the path information of "$R_1 - R_2 - R_3$". Thirdly, as the destination is the verifier in the path verification, the mechanism that each on-path router i processes its own onion $mark_{r_i}$ could not improve the security performance since an adversary could change other router's $mark$ or even drop the packet. We will discuss this limitation in Sec. VII-B.

Then the source decreases the $CSH[m]$ and sends the packet to the first hop (line 7-8). The source turns to the next epoch with the number of $CSH[m+1]$ after $CSH[m]$ decreases to 0.

The destination achieves the source and path verification with Alg. 2.

The destination initializes the counters, which record the number of packets that passed its verification. The $sCounter[M][I]$ is a two-dimensional array, and $sCounter[m][i]$ represents the counter for on-path router i, epoch m. The destination receives a packet in epoch m (line 1). It calculates the $mark'_{pkt}$ with Equation (9), then compares it with $\boxed{mark_{pkt}}$ (line 2). After a successful verification, it increases the relative router's $sCounter[m][i]$, or else filters the packet (line 3-5). When it receives a certain number (e.g., 200) of packets in epoch m, the destination determines that it is the end of epoch $m-1$ if $m > 0$. The destination verifies whether each $sCounter[m-1][i]$ matches the policy as Equation (10). In epoch m, N is the desirable value of $sCounter[m][i]$, $N = CSH[m] \times P_i$. P_i is the verification probability of router i, the default value is $1/n$ with the path length of n. We set $\beta_1 = \beta_2 = 5\%$, which means that the packet damage and the packet replay ratio should be less than 5%. If router i violates the policy, the verification failed at router i (line 6-9). That does not mean router i is a false router. For example, an adversary might frame router i by dropping the packet verified by router i. As a verification mechanism, MASK is devoted to verifying whether the network forwards the packet as the user expected. It could not localize the false router.

Algorithm 2 Procedure at Destination Host

```
input : $pkt, HDK, CSH[M]$,
        $sCounter[M][I] = \{\{0\}, \cdots \{0\}\}$
1  while receive a packet in epoch $m$ do
2      if $\boxed{mark_{pkt}} == mark'_{pkt}$ then
3          $sCounter[m][i]++$
4      else
5          filter the packet
6      if the end of epoch $m-1$ then
7          for $sCounter[m-1][i]$ in $sCounter[m-1][I]$ do
8              if $sCounter[m-1][i]$ violate the policy then
9                  verification failed at router i
10         $sCounter[m-1][I] \leftarrow \{0\}$
```

$$N \times (1-\beta_1) < sCounter[m][i] < N \times (1+\beta_2) \qquad (10)$$

We do not assume an exact report mechanism towards the source. The destination could reply to the source with an ACK, which includes the $sCounter[m-1][i]$ encrypted with the HDK. The ACK from the destination to the source on the reverse path is similar to the forward path. To prevent an adversary drop the ACK, the destination could set the $\boxed{mark_{pkt}}$ with $mark_{pkt}+1$ to implicitly indicate it is an ACK.

The destination records the counters of two epochs for each session. It clears the $sCounter$ of epoch $m-1$ after replying with the ACK (line 10). To satisfy that the users accomplish the verification of the epoch $m-1$ within epoch m, each epoch should take more than one RTT. The RTT on the Internet peaks at 100 ms and 200 ms.[3] For a packet size of 1500 B, 4000 packets in an epoch make up 48 Mb. The forwarding delay of the packets in an epoch is nearly 500 ms with an end-to-end bandwidth of 100 Mbps. One epoch is enough to receive the ACK. Therefore, MASK utilizes 4096-8192 packets in an epoch. The user could increase or decrease it if necessary.

C. Processing at Router

When a router receives a packet, it first checks that the packet was received through the correct interface according to the $\boxed{id_{i-1}}$, to filter the packet without the correct $\boxed{PATH}$. Then it processes the packet as Alg. 3.

[3]https://www.caida.org/catalog/software/walrus/rtt/

Algorithm 3 Procedure at Router

```
input : $pkt, RDK_{si}, id_i, id_{i-1}$
1 calculate $mark_{src_i}$ and $mark_{r_i}$
2 if $\boxed{id} == id_i$ then
3     if $\boxed{mark_{src}}$ != $mark_{src_i}$ then
4         drop the packet
5 $\boxed{mark_r} \leftarrow mark_{r_i}$
6 send packet to the next hop
```

Each on-path router i at most performs one MAC operation in the data plane. It calculates the $mark_i$ with the router's dynamic key RDK_{si} as Equation (6), then derives the $mark_{src_i}$ with Equation (7) and $mark_{r_i}$ with Equation (11) (line 1). If the id_i in the packet header is the same as $\boxed{id}$, it achieves the source verification by comparing the $mark_{src_i}$ with the $\boxed{mark_{src}}$. The router drops the packet if the source verification failed, to filter the malicious traffic (line 2-4). Then it updates the $\boxed{mark_r}$ with $mark_{r_i}$ (line 5), and sends the packet to the next hop (line 6).

$$mark_{r_i} = mark_{r_{i-1}} \oplus mark_i[32:64] \tag{11}$$

D. Agent-Based MASK

The motivation of Agent-based MASK is offloading the AS border router's storage overhead for each session. This hierarchical design is consistent with the Internet. As shown in Fig. 2 (d), each agent is a border router of a subnet in an AS. It has 1 symmetric key SK_u with each user (end host or other entities) which wants to achieve the verification. The source and agent process the packet as Alg. 4 and 5.

In Alg. 4, the processing at source host is nearly the same as Alg. 1 except that the source host only has the SK_u instead of RDK_{si}. Therefore, the $\boxed{mark_{src}}$ is $mark_{src_a} = MAC_{SK_u}(CST_{PKT})[0:32]$, the $\boxed{mark_r}$ is $mark_r = hash$, and the $\boxed{mark_{pkt}}$ is the same as Equation (9).

Algorithm 4 Procedure at Source Host (Agent-Based MASK)

```
input : $PATH, SK_u, HDK, CSH[M]$
1 for $CSH[m]$ in $CSH[M]$ do
2     while $CSH[m] >= 0$ do
3         embed the $\boxed{flag}$ $\boxed{epoch}$ $\boxed{seq}$
4         determine $id_i$ and calculate $marks$
5         $\boxed{\text{id}} \leftarrow id_i$, $\boxed{mark_{src}} \leftarrow mark_{src_a}$
6         $\boxed{mark_r} \leftarrow hash$, $\boxed{mark_{pkt}} \leftarrow mark_{pkt}$
7         $CSH[m] \leftarrow CSH[m] - 1$
8         send packet to the agent
```

In Alg. 5, the agent has the SK_u and RDK_{path}. After receiving a MASK packet. It first calculates the $mark_{src_a} = MAC_{SK_u}(CST_{PKT})[0:32]$ (line 1). Then it compares it with the $\boxed{mark_{src}}$ in the packet header (line 2). After successful verification, it calculates the $mark_{src_i}$ with Equation (7), $mark_r$ with Equation (8), and updates the $\boxed{mark_{src}}$ and $\boxed{mark_r}$ (line 3-4). Or else, it drops the packet (line 5-6). With this design, the agent is transparent to the on-path routers and the destination host, the routers and the destination process the packet as Alg. 3 and 2.

Algorithm 5 Procedure at Agent

```
input : $pkt, SK_u, RDK_{path}$
1 calculate $mark_{src_a}$
2 if $\boxed{mark_{src}} == mark_{src_a}$ then
3     calculate $mark_{src_i}, mark_r$
4     $\boxed{mark_{src}} \leftarrow mark_{src_i}$, $\boxed{mark_r} \leftarrow mark_r$
5 else
6     drop the packet
```

To improve security, the SK_u between the source and the agent could switch with the ts in the packet header. For example, $SK_u = MAC_{SK}(IP_{src}||ts)$, in which SK is the root key, and IP_{src} is the IP address of the source.

In Alg. 5, the agent needs to perform n MAC operations to calculate $mark_r$. To decrease the computation overhead of the agent (as well as the on-path router), the user could set the lowest bit of $flag$ with 1. In this case, the agent only updates the $\boxed{mark_r}$ with $mark_r = hash \oplus mark_i[32:64]$. Then only the selected R_i on the path needs to calculate the $mark_i$, other on-path routers forward the packet directly. At the destination, $\boxed{mark_r}$ is $hash$ in the two cases. It decreases the computation overhead of the agent from $O(n)$ to $O(1)$, and an unselected on-path router has no additional computational overhead. We will detail in Sec. V that the colluding attackers might change the path of partial packets and avoid the verification at the destination.

V. Analysis

In this section, we analyze the security and performance properties of MASK.

A. Security Analysis

An overview of the security properties in OPT [15], PPV [21] and EPIC [20] is shown in Tab. [I]. In which ✓, (✓), and × represent satisfied, partially satisfied, and unsatisfied, respectively. We first describe the honesty assumptions in different scenarios. Then we detail the security properties. In general, path authorization is the ability that packets to traverse the network only along paths authorized by all honest on-path ASes (AS_i). The packet authentication for destination (D) is that the destination agrees with the source on the packet source address, path, and payload. The source authentication for AS_i means that the on-path ASes agree with the source on the packet source address. The path validation for the destination means that the destination can verify that the

TABLE I
THE SECURITY PROPERTIES IN OPT, PPV AND EPIC

	Who	Honesty assumption				Security property			
		OPT	EPIC	PPV	MASK	OPT	EPIC	PPV	MASK
Path authorization	AS_i	-				(✓)	✓	×	×
Packet authentication	D	S, AS_s, AS_d			$S\ (agent), KDS_s, KDS_d,$	✓	✓	✓	✓
Source authentication	AS_i	S, AS_s, AS_d, D	S, AS_s	-	$S\ (agent), KDS_s$	✓	✓	×	(✓)
Path validation	D	AS_s, AS_d, S		AS_d	$KDS_s, KDS_d, S\ (agent)$	✓	✓	(✓)	✓
Path validation	S	AS_s		-	KDS_s, KDS_d, D	✓	✓	×	✓
Packet drop resilience	D	-			S	×	×	×	✓
Reply attack resilience	AS_i/D	-				(✓)	(✓)	×	(✓)

packet traversed all honest ASes on the path from the source to the destination intended by the source, while the validation for the source (S) represents that the source can verify that the original packet traversed all honest ASes on the intended path by receiving a reply from the destination. The packet drop is for the destination to detect whether the on-path router drops (or delays) the packets. In the last, the replay attack resilient means that packets are uniquely identifiable and cannot be replayed.

Honesty Assumptions. We assume the adversary could generate the correct $mark$ with negligible probability without the dynamic key. Unlike an entity using asymmetric encryption can prove its honesty through private keys, in symmetric encryption mechanisms, one of the entities needs to be assumed to be honest since two entities keep the same key. In EPIC and OPT, the source (S) and the destination (D) require the dynamic key from the source AS (AS_s) and destination AS (AS_d), respectively. In MASK, the users require the keys from their local KDSes (KDS_s and KDS_d).

For packet authentication at the destination, the destination assumes that the S, AS_s, and AS_d are honest in OPT, EPIC, and PPV. It is the same in MASK, the honesty of KDS_s and KDS_d guarantee that the $S\ (agent)$ and D could get the correct dynamic keys, and the honesty of the $S\ (agent)$ guarantee that the original $mark$ is correct.

For source authentication at the AS_i, OPT assumes that the $S,\ AS_s$ and the AS_d, D are honest, while EPIC assumes that the S and AS_s are honest. PPV does not achieve source authentication. In MASK, it assumes that the $S\ (agent)$ and KDS_s are honest.

For path validation at the destination, OPT and EPIC assume the $AS_s,\ AS_d,\ S$ are honest, PPV only needs the AS_d to be honest, while MASK assumes the $KDS_s,\ KDS_d,\ S\ (agent)$ are honest. For path validation at the source, OPT and EPIC need the AS_s to be honest, MASK assumes that the $KDS_s,\ KDS_d,\ D$ are honest.

For packet drop resilience at destination (D), MASK assumes that the source (S) is honest. Thus the destination could defend against the packet drop attack with the per-session policy negotiated with the source.

In a word, MASK is a mechanism to empower the users (source and destination) to verify whether the network forwards the packet as their expectation. Therefore, we assume that the KDS, the agent, the source, and the destination are all honest. The honesty of the KDS and agent is promised by the infrastructure (e.g., PKI), while the KDS and agent manage the local users in a centralized way. However, it might appear like a strong assumption. An adversary might compromise any of these entities (KDS, agent, source, or destination) to launch active attacks. But a compromised entity could only avoid the verification between itself and other entities. It could not impersonate other entities since it only has its keys. An adversary might manipulate a huge number of source hosts to launch a DDoS attack, which is out of this paper.

Path authorization. OPT could satisfy the path authorization with a long $mark$, but it has limited security since many packets utilize a specific authenticator. EPIC achieves the path authorization by utilizing the whole hop authenticator as an input of the $mark$, and the whole hop authenticator is not in the packet header. PPV and MASK do not support path authorization. We maintain that the path authorization has already existed in a specific PAN (e.g., in SCION, SRv6). We could utilize it in MASK in a probabilistic way. For example, we utilize the hop authenticator in EPIC as an input of the Equation (7). As the hop authenticator needs another MAC operation, we could cache it in practice [39].

Packet and source authentication. OPT and EPIC could authenticate the packet and source with the $mark$ calculated by the shared key. But OPT assumes that the source authentication on the router is based on the assumption that the destination is honest, which weakens the source authentication property compared to EPIC. PPV achieves packet authentication at the destination but does not support the on-path router to authenticate the source.

In MASK, the destination could authenticate the packet, while the on-path router authenticates the source in a probabilistic way. It is relatively weaker than EPIC, in which each on-path router authenticates each packet. And it might incur attacks on the probabilistic authentication at the first glance. We now detail that it does not hurt the verification.

Considering two types of attacks: (1) Attack from a single adversary, it could modify the $\boxed{id}$ for the downstream router to evade the probabilistic authentication. But as the $\boxed{id}$ is part of the input of the $mark$, the $mark_{r_i}$ of downstream routers are wrong. The destination will filter this packet with the $\boxed{mark_{pkt}}$. (2) The colluding attack, for example, R_1 and R_3 in Fig. 1, R_1 modifies the $\boxed{id}$ from R_2 to R_4, R_2 does not authenticate the source, and R_3 modifies the $\boxed{id}$ back to R_2 again. This packet can reach the destination, but as the $\boxed{id}$ is part of the input of the $mark$, the $mark_{r_2}$ is wrong. Then the destination will filter the packet based on the $\boxed{mark_{pkt}}$. At the end of each epoch, the destination will find that the probability is not the same as the policy negotiated with the source.

In a word, probabilistic authentication only lacks the ability to filter the malicious traffic at the first honest router. But as MASK keeps the backward compatibility, and it incurs little computation and storage overhead in the router, this attack is the same as an attacker sending the legacy packet into the network, the colluders have no benefit to launching this attack. And the packet was eventually filtered by the destination.

It is worth mentioning that the colluders could evade the authentication if only the selected router updates $mark_r$ since it could only guarantee that a packet was processed by the selected router, not all the on-path routers. It cannot defend against colluding attacks. From this perspective, to defend against colluding attacks, each on-path router must update the $mark_r$.

Path validation. OPT and EPIC satisfy the path validation properties through the relative validation field. PPV does not allow the source to validate the path and only probabilistically validates individual links at the destination. MASK could satisfy the path validation at the destination. And the destination replies to the source with an ACK at each end of an epoch. We have discussed the honest assumption of the destination.

Packet drop resilience. A malicious router might drop packets (or delay a packet more than a certain number of epochs) during the processing; only MASK could detect the packet drop attack according to the policy. Others could not achieve that since they do not count the number of packets for a session. The stateful counter might incur the DDoS attack towards the destination, especially for the communication from the client to the server. As the number of sessions in the network layer is less than the number of sessions in the transport layer, a DDoS attack detection and mitigation mechanism [40] can mitigate these attacks effectively. Besides that, the destination can simply clear the counter information of a specific session after time out.

Replay attack resilience. To prevent an adversary replay a legitimate packet, MASK utilizes the unique seq to distinguish the replay packets at the destination. At each end of an epoch, the destination compares the number of legitimate packets with the threshold and reports to the relative entity (e.g., the source) if there's a replay attack in this epoch. AS_i in MASK could not detect the replay attack since the on-path router is stateless. OPT has limited support for replay suppression, and EPIC could achieve it with a replay-suppression system. However, the two mechanisms rely on the packet timestamp, which is impractical in today's inter-domain context. PPV does not support replay attack resilience.

TABLE II
THE COMPARISON OF OVERHEAD WITH OPT, PPV AND EPIC

	Computation overhead			Communication overhead (B)		
	No asy.	No deriving	MAC operations	n	n=5	n=10
OPT	×	×	2n	16n+68	148	228
PPV	×	×	4	64	64	64
EPIC	✓	×	n	5n+24	49	74
MASK	✓	✓	n(1)	24	24	24

Based on the above analysis, we can conclude that the relative entity in MASK could defend against different types of attacks effectively. Specifically, the adversaries, which launch the packet damage (including packet drop), path change, replay attack, and colluding attack, could not evade the verification at the destination and the source. Besides that, as MASK has minimum computation overhead and constant storage overhead in the router, packet flooding has little effect on it.

B. Performance Analysis

We have detailed the constant storage overhead in Sec. IV-A. In this section, we analyze the computation overhead in the router and the communication overhead in the network, which affects the performance of the verification. As shown in Tab. [II], MASK has lower computation and communication overhead compared with OPT [15], EPIC (EPIC L3 as default) [20] and PPV [21].

Computation overhead. The cryptographic operation accounts for most of the computation overhead. "No asy." represents that the router needs not to perform the per-session asymmetric cryptography. EPIC and MASK could derive the dynamic keys from the key server; others have to negotiate the dynamic keys by asymmetric cryptography, which incurs the computation overhead in the data plane. "No deriving" means that the router needs not to derive the dynamic key for each packet on the fly. Only MASK satisfies it, which significantly minimizes the computation overhead. In MASK, each on-path router only operates 1 MAC calculation in the data plane. Therefore, MASK needs n per-packet MAC operations. And it could decrease the number of MAC operations to 1 if only the selected router updates the $mark_r$. While OPT, PPV, and EPIC L3 need to derive the dynamic keys on the fly. Besides the key deriving, OPT operates at least $2n$ MAC operations for one packet. PPV operates 4 MAC operations. EPIC L3 processes 1 MAC operation and truncates it into two $marks$ to achieve the verification, thus having n MAC operations. We do not take account into the MAC operation for the hop authenticator. And we will discuss in Sec. VI that we could cache the hop authenticator in the control plane.

Communication overhead. MASK and PPV have stable additional packet header sizes, while the OPT and EPIC headers depend on the path length. We illustrate the communication overhead with 5 and 10 hops since the current average path length on the Internet is less than 5 AS-level hops, and the vast majority of Internet AS hops is at most 10 [41]. The communication overhead of MASK for realistic path lengths is 1/2 to 1/10 compared with other mechanisms. Besides that, the user in MASK requires the dynamic key from the local KDS. It only sends one ACK at each end of an epoch, thus having little communication overhead incurred by the control packets.

VI. Evaluation

To show that MASK is practical, we evaluated MASK on both software (Bmv2[4]) and hardware (Barefoot Tofino programmable switch).

A. Evaluation in BMv2

We first implement MASK and the state-of-the-art mechanisms in the BMv2 environment, the testbed hosts in a virtual machine with Ubuntu 16.04, with the Intel Core i5-6200U CPU, 2.4 GHz, and 4 GB RAM.

To achieve the verification in each router, we instantiate MAC operation with the HMAC based on MD5. The BMv2 could not handle the HMAC operation in the data plane. We implement it as an $extern$ function. As shown in Fig. 6, there are three Match-Action tables. The first ((a) FromController) is to judge whether the $extern$ has processed the packet. It sends a processed packet to the third table ((c) Forward) for forwarding. In contrast, it sends an unprocessed packet to the second table ((b) CompareID) to judge whether it needs to verify the packet and update the $mark$. In OPT and EPIC, each packet was verified and marked, while in PPV, two adjacent hops marked a packet with the probability of $1/k$, in which k is the number of remaining hops. In MASK, each hop compares the id_i with its own id to decide whether it sends the packet to $extern$ or forwards the packet according to the third table directly. In OPT, EPIC, and PPV, the $extern$ first derives the dynamic key, then calculates the relative $marks$, verifies and updates the packet header. In MASK, the $extern$ only calculates and updates the $mark$. We do not implement MASK with the mechanism that each on-path router updates its $mark_{r_i}$ since it has nearly the same delay as EPIC and OPT.

We also instantiate the MAC operation in MASK with SipHash [42] and 2EM [36] in the data plane. We implement the SipHash with SipHash-2-4-64,[5] and 2EM with 192-bit dynamic key and 64-bit output ($mark$). Then we truncate the $mark$ to 32-bit to achieve the verification.[6] We did not implement the verification in the data plane for OPT, EPIC, and PPV, since they need a MAC operation with strong security (e.g., AES) to derive the dynamic key. We will detail in Sec. VI-B that the AES operation is impractical with today's hardware.

As shown in Fig. 5 (a),(b),(c), we evaluate these mechanisms in terms of end-to-end delay with the path length from 2 to 10, which composes the computation and forwarding delay. The baseline is the IP (IPv6) packet with nearly 50 ms in different path lengths and packet sizes. MASK-E represents MASK with the extern HMAC operation. In contrast, MASK-D represents MASK with the SipHash-2-4-64 (or 2EM) in the data plane. The delay of MASK-D is nearly the same as the baseline. MASK-E and PPV have a constant delay of 110 ms and 300 ms, respectively. In contrast, OPT and EPIC have a relatively high delay (300 ms-800 ms) since they operate the extern HMAC operation at each hop. Although the BMv2 environment does not represent the actual processing ability, we can conclude from the simulation that (1) compares with per-hop MAC operation, probabilistic MAC operation could reduce the computation overhead of routers from $O(n)$ to $O(1)$, (2) packet processing in the data plane could decrease the computation overhead to a large extent.

B. Evaluation in Barefoot Tofino Hardware

We evaluate MASK for the desired performance in commodity Barefoot Tofino programmable switch S9180-32X.

The central aspect of MASK is the lightweight processing in each hop, which has significantly lower computation overhead than other mechanisms. We evaluate the computation overhead in terms of per-packet processing latency. The computation overhead on a certain hop includes (1) source verification (calculating and verifying the source's $mark$), and (2) path verification (calculating and updating the on-path router's $mark$). We first test the delay from the data plane to the local controller, which takes 3 ms for one packet. It means that the external hardware would result in a very high delay.

Therefore, we implement the MAC operation in the data plane. We first implement it as AES operation in data plane [37], each packet recirculates 4 times to complete an AES-128 operation (10 rounds). Then we implement the HalfSipHash-1-1-32 (the SipHash-2-4-64 could implement in the BMv2 but could not accomplish in the hardware in a single pipeline) and 2EM operation in the hardware. The two mechanisms could generate two 32-bit $marks$. The number of stages in one pipeline is limited (12 stages in Tofino and 20 stages in Tofino2). The implementation of HalfSipHash-1-1-32 takes 23 stages (12 stages in the ingress pipeline, and 11 stages in the egress pipeline). In contrast, the 2EM operation only takes 7 stages, it can be finished within one hardware pipeline.

Besides that, the security guarantee of 2EM operation is sufficient for an ephemeral $mark$. It has been proven to secure up to $2^{2n/3}$ queries under adaptive chosen-plaintext and ciphertext adversaries. As detailed in [36], the best-known practical attacks against 2EM, which are chosen-plaintext attacks, require more than 2^{89} bits of memory for $n = 64$ even if the adversary can send more than 1 trillion queries per second, which generate approximately 2.4 Pbps traffic (assuming the average packet size is 300 B). In practice, it is infeasible for an attacker to achieve it within 1 minute. And generating a large volume of traffic, if possible, can be easily flagged as DDoS attacks. Furthermore, if MASK utilizes the hop authenticator (which is not in the packet header) for one of the inputs of the $mark_i$, the adversary could not get the whole plaintext from the packet header. Therefore, the minute

[4]https://github.com/p4lang/behavioral-model
[5]https://github.com/veorq/SipHash/
[6]https://github.com/programmableNetwork/mask-BMv2

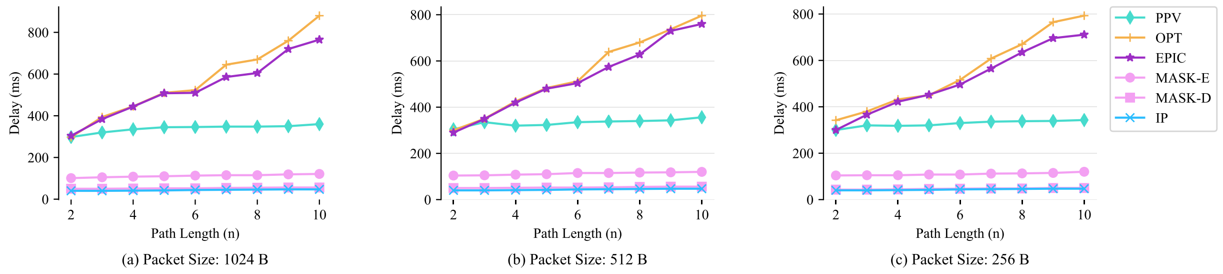

Fig. 5. The evaluation in BMv2 environment.

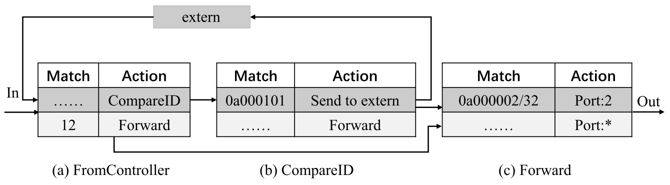

Fig. 6. MASK processing with extern MAC operation in BMv2.

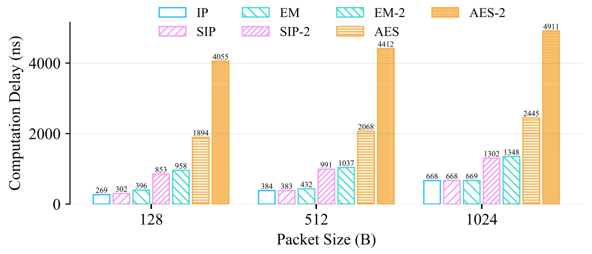

Fig. 7. The computation overhead in Barefoot with different packet size.

granularity synchronization in the control plane is enough for sharing the dynamic keys. 2EM operation is practical for the data plane MAC operation in MASK.

It is convenient to utilize the packet's timestamp leaving and entering the ingress pipeline to measure the delay in the hardware. The delay in different packet sizes is shown in Fig. 7, in which IP represents IPv6 packet forwarding, which is the baseline with the value of around 668 ns for the packet size of 1024 B, EM represents the 2EM operation, SIP represents the HalfSipHash-1-1-32 operation, EM-2 represents two 2EM operations, SIP-2 represents two HalfSipHash-1-1-32 operations, AES represents a single AES operation, while AES-2 represents two AES operations. The evaluation testifies that the EM and SIP operation has nearly the same computation delay, and nearly the same as the baseline when the packet size is more than 512 B, the SIP-2 has a delay of twice of SIP, which has nearly 1302 ns when the packet size is 1024 B, while the EM-2 has a delay of 1348 ns when the packet size is 1024 B. One AES operation has a delay of nearly 2445 ns for the packet size of 1024 B. In contrast, for the packet size of 1024 B, the two AES operations cost 4911 ns to accomplish the operation. The results confirm that the recirculation in the data plane could significantly increase the computation overhead in each hop.

The MAC operation in different mechanisms behaves the same on specific hardware, but not all the MAC operations

TABLE III
EACH HOP'S COMPUTATION OVERHEAD IN DIFFERENT MECHANISMS

	Total Operation/ number	Computation delay(μs) 512 B	1024 B
OPT	AES/2	4.41	4.91
PPV	2EM/2	1.04	1.35
EPIC	2EM/1	0.43	0.67
MASK	2EM/1	0.43	0.67

could satisfy the requirements of any particular mechanism. For example, the OPT needs the AES operation because the $mark$ in each hop is 128-bit. We enumerate the number of MAC operations and the computation delay in Tab. [III] based on the requirements of these mechanisms.

Then we compare the processing of source and path verification at each hop. In OPT, hop i validates the OPV_i and updates the PVF_i in the packet header. Only the AES-128 operation could satisfy the requirement. In other mechanisms, the 2EM could satisfy the verification requirements. In PPV, hop i only achieves the path verification, but it should also calculate the MF_i and updates the MVF in the packet header, thus needing 2 MAC operations. In EPIC L3, hop i only calculates C_i with 1 MAC operation. It utilizes $C_i^{[1]}$ to authenticate the source, then updates the HVF with $C_i^{[2]}$. In MASK, we truncate the 64-bit $mark$ to two $marks$ (32-bit $mark_{src}$ and 32-bit $mark_r$). Then we could achieve the source and path verification with 1 MAC operation. The results show that MASK incurs little computation delay compared with IP packet forwarding.

As the average path length of the Internet is nearly 5, and most of the path lengths are less than 10, we compare the computation overhead of various mechanisms with different path lengths. As shown in Fig. 8 (a), with a packet size of 512 B, MASK has a little more delay than the IP packet forwarding (baseline). While in Fig. 8 (b), with the packet size of 1024 B, MASK has nearly the same delay as the baseline. OPT has much more computation delay since it needs 9 recirculations to achieve 2 AES operations. PPV has less delay than OPT since it could utilize 2 2EM operations, which need 1 recirculation in each router, and PPV only processes the MAC operation in two hops for a specific packet. EPIC has the same delay since it could utilize the 2EM operation to generate C_i. The 2EM has weaker security guarantees compared with AES, MASK could switch the dynamic key

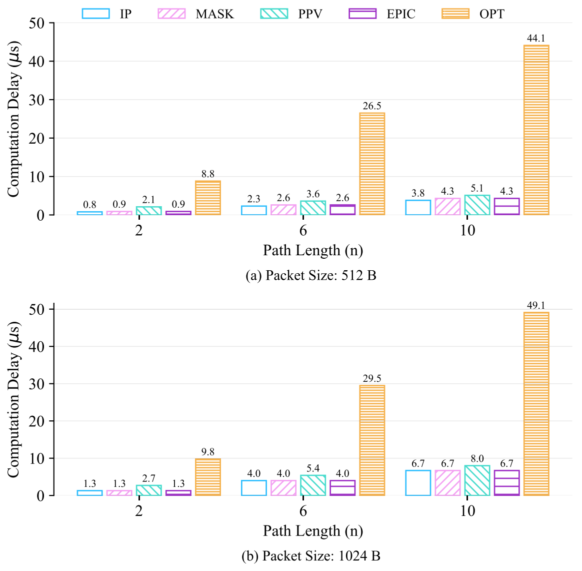

Fig. 8. The computation overhead with different path lengths. The operations include the negotiation of dynamic keys (asymmetric operation) in OPT, deriving the dynamic keys (an AES operation) on the fly in OPT, PPV, and EPIC, and deriving the hop authenticator in EPIC are not considered. It is impractical to achieve these operations in OPT, PPV, and EPIC with today's programmable hardware.

with the granularity of minutes. PPV and EPIC need to utilize a similar mechanism to guarantee the security of the $marks$. Or else, it needs to utilize a more secure MAC operation (e.g., AES), which has the same delay as OPT.

The operations include the negotiation of dynamic keys (asymmetric operation) in OPT, deriving the dynamic keys (an AES operation) on the fly in OPT, PPV, and EPIC, and deriving the hop authenticator in EPIC are not considered. The reason is that we focus on achieving the practical data plane processing of relative mechanisms. As detailed in [39], the authors detailed that the MAC operation is impractical with today's programmable hardware. And we assume the key and the hop authenticator could be cached in the hardware. We will detail that the AES could decrease the throughput to a large extent though it could achieve with recirculations. On the other hand, Intel's AES-NI hardware could reduce the computation latency [20]. MASK could also benefit from this dedicated component.

MASK could achieve higher throughput than other mechanisms since it accomplishes the processing in a single pipeline. As detailed in [37], the throughput of each hop with AES operation was affected by the hardware's recirculation bottleneck since a packet recirculates several times to accomplish one AES operation. In [37], the authors utilize 200 Gbps recirculation bandwidth. The throughput for an AES-128 operation (requiring 4 recirculations / 5 passes) on RMT programmable switch is 10.92 Gbps (85.3 Mpps), while 4.91 Gbps (38.4 Mpps) for 9 recirculations (10 passes), which could process two AES-128 operations. In practice, we could achieve a higher bandwidth by reserving more switch ports as loopback. For example, we could reserve $2n/(2n+1)$ times of total ports for n recirculations. It then could achieve $1/(2n+1)$ line rate. For example, to achieve 1 recirculation in S9180-32X (30 ports with 100 Gbps), we could reserve 20 ports $(2/3)$ for the recirculation, and the throughput could achieve 1 Tbps $(1/3)$. When $n > 5$, we could not reserve the ports with the ratio of $2n/(2n+1)$ since we need a certain number of ports to connect with other networks. MASK could achieve the verification in a single pipeline, which means it can process the packets at a hardware line rate, thus improving its deployability in the realistic network.

TABLE IV
THE GOODPUT RATIO IN DIFFERENT PATH LENGTH

n	2	3	4	5	6	10
OPT	90.91	89.61	88.34	87.11	85.91	81.43
PPV	93.98	93.98	93.98	93.98	93.98	93.98
EPIC	96.71	96.25	95.79	95.33	94.88	93.11
MASK	97.66	97.66	97.66	97.66	97.66	97.66

C. Communication Overhead

We evaluate the communication overhead of MASK and compare it with other mechanisms. We only consider the overhead owed to security since the routing headers (e.g., SRv6, SCION) depend on the underlying PAN. Therefore, we evaluate the goodput in different mechanisms. We use the additional packet header to load the security-related fields and define the additional packet header size as HD. The goodput ratio (GR) is the ratio of payload size (p) and total packet size (p+HD), namely GR = p/(p+HD). As shown in Tab. [IV], the goodput ratio is high for MASK. For the payload size of 1000 B in n = 5, the additional header in EPIC L3 is 49 B, which corresponds to a goodput ratio of 95.33%, while the value in OPT is 87.11%. In MASK, the ratio is stable at 97.66% regardless of the path length, while PPV has a stable ratio with a value of 93.98%.

D. Performance at End Host

In MASK, the end host requests dynamic keys from the KDS, and the KDS answers with the relative dynamic keys before a new session. We assume that the underlying PAN minimizes latency by locally caching public paths. And the Multi-AS-Key has been synchronized among the KDSes. Therefore, the additional latency incurred in MASK is insignificant since the paths and keys information is available at the local KDS. Moreover, an end host can cache both path information and keys, eliminating extra latency for subsequent packets. Meanwhile, with the policy shared by Equation (5), there's no inter-domain negotiation between the source and the destination.

We first evaluate the computation overhead at the end host. The source and destination need to perform the MAC operations for each packet, which comprises the heaviest overhead. We test different hash and MAC operations in the virtual machine with an Intel Core i5-6200U CPU. It could achieve more than 8,000,000 cycles for SipHash-2-4-64 (1,000,000 cycles for $2EM$) in one second, which could satisfy the calculation of $mark_{pkt}$. The computation delay in the end host is detailed in Tab. [V]. At the source, it needs a

TABLE V
THE COMPUTATION OVERHEAD AT END HOST

	SIP/ Delay(µs)	2EM/ Delay(µs)	Total delay (µs) n=5	n=10
S	1/0.125	n/1	5.125	10.125
D	1/0.125	0/0	0.125	0.125

SipHash-2-4-64 operation to calculate the $mark_{pkt}$, a $2EM$ operation to calculate the $mark_{src}$, and the $mark_r$ of each on-path router. In contrast, the destination needs to calculate an $mark_{pkt}$. The end host could calculate the hash of the payload with a computationally effective hash operation, e.g., the CRC32 which only costs less than 20 ns in our virtual machine. With the total delay of a certain packet, we could learn from Tab. [V] that an ordinary commodity host could support more than 1 Gbps with a packet size of 1024 B in a realistic AS-level path length. We can conclude that the overhead in the end host is sustainable for the source and path verification.

E. Performance at Agent

In Agent-based MASK, the performance of border routers and the destination is the same as in MASK. While the performance at the source is less than in MASK since it process only one MAC operation for the agent, then the delay is 1.125 μ s no matter the path length.

The agent first verifies the source with one $mark$, and initializes the $mark_r$ and $mark_{src}$ with n (path length) $marks$. For the path length of 4, it processes 5 2EM operations, which need 4 recirculations. The processing latency in S9180-32X is the same as AES, which takes 2445 ns for the packet size of 1024 B. For the path length of 9, it processes 10 2EM operations, which takes 4911 ns for the packet size of 1024 B. Considering that each packet needs 9 recirculations in S9180-32X, we could reserve 18 ports for 9 recirculations, the throughput could achieve at least 100 Gbps. In some scenarios which need the throughput at the magnitude of Tbps, we could utilize the $flag$ to instruct the agent only to process one MAC for the selected router. Then the agent only needs to perform 2 MAC operations for each packet, which could achieve 1 Tbps in S9180-32X. Therefore, the latency and the throughput are sustainable for the agent to process the packet in a subnet.

VII. DISCUSSION

In this section, we discuss incremental deployment issues and the limitations of MASK. We first discuss a high-level approach for implementing MASK on the Internet, then discuss attacks that MASK does not entirely defend.

A. Incremental Deployment

As the Internet's architecture is a flexible composition of many networks [43], the dedicated network could use some new designs [44], [45]. From this perspective, we insist that MASK could first deploy in some dedicated network. For example, MASK could be integrated with SRv6 [46] to verify the source and the segment node in an efficient way. The integration could directly utilize the segment lists as the path information and the Type Length Value (TLV) field to load the additional packet header of MASK [47]. MASK could also be used in SCION to enjoy lightweight processing in the routers. For example, MASK can be implemented as the substitute of $\boxed{\text{VALHD}}$ in the EPIC packet header. Some other dedicated networks, such as satellite networks that need more security guarantees, could adopt MASK to enjoy a secure data plane. As MASK could achieve in programmable hardware with line rate, the operator in a dedicated network could directly implement it. The legacy router could also benefit from it. It could utilize the Policy Based Routing to connect the traffic to an implemented programmable hardware.

Besides PAN, MASK could also use in other networks. For example, the ASes could exchange the AS-PATH with BGP. We name it MASK without source authentication in an on-path router. It has no path information at the source and packet header. The source initializes the MASK header with $mark_{pkt}$ and $indicator$. Only one on-path router randomly updates the $\boxed{id}$ with id_i and relative $\boxed{mark_r}$, $\boxed{mark_{src}}$. The destination could require the dynamic keys from the KDS, and reconstruct the actual path based on the verified packets. MASK without source authentication in an on-path router is just like PPV [21], except that MASK need not derive the dynamic keys on the fly, thus only processing 1 MAC operation in the data plane of the router, and could benefit from the policy shares between the source and the destination to detect the packet drop and packet replay attack. It could achieve the verification even with a partial deployment, thus benefiting the early adopted ISP to draw traffic towards them and appeal to existing and new clients who desire adequate protection.

B. Limitations

MASK could not verify the attack from adjacent colluders, e.g., the on-path routers, R_i and R_{i+1} forward the packet to another colluding router (R_j), not on the expected path. This weakness exists in any path verification mechanism; even every packet verified at each hop could not verify the fault since R_i and R_{i+1} could calculate the $marks$ pass any entities' verification [15].

Besides that, the verification cares about whether the network forwards the packet according to the end hosts' policy. It could not localize the fault entity if the adversary frames other entities or drops the packet. It needs a fault localization mechanism to localize the fault entities, in which each on-path router records the testimony and returns it to the source (or other entities) [48].

VIII. RELATED WORK

Path-Aware Networking. The PANs, such as NEBULA [26], and SCION [17], give transparency and choices to end hosts, while SR [49] provides a practical way to program a dedicated network. These mechanisms are orthogonal with MASK and provide the basis of path verification by allowing the hosts to embed a path of their choice in the packet header.

Cryptographic operation in programmable data plane. The ability of cryptographic operation in programmable hardware is important for verification in the data plane. Previous works mostly focus on dedicated hardware, which might have scalability or compatibility issues [50]. SPINE [51] implements SipHash in the BMv2 environment. It needs at least three packet recirculations on a hardware switch due to resource constraints, which degrades the performance. P4-AES [37] requires at least four recirculation on hardware switch, and the highest throughput only achieves 10.92 Gbps. PINOT [36] could encrypt the IP address with 2EM operation in a single pipeline on commercial hardware. MASK stores the dynamic keys, utilizes 2EM operation to calculate two 32-bit $marks$ in a single pipeline, then achieves the lightweight source and path verification with the two $marks$.

Source and path verification. Source and path verification has been extensively studied in the literature [52], [53], [54], [55]. The mechanisms like Pi [52] and SNAPP [56] utilize path identifier to detect the DDoS attack or to pin the forwarding path, which could achieve path verification. Still, the attacker could easily forge the identifier. In ICING [19], the router computes a MAC for each downstream on-path router. It requires each router to compute $n + 1$ MAC operations per packet. OPT [15] utilizes DRKey to share the dynamic keys between the hosts and the routers. RFL [57] could further achieve this verification on unreliable communication channels. EPIC [20] proposes a sophisticated mechanism to improve the security and efficiency of the verification. It provides a hierarchical security level (named L0-L4) for different security requirements. Atlas [58] extends the verification to multipath routing. Atomos [54] achieves the verification with constant-size proofs by a scheme of noncommutative homomorphic asymmetric-key encryption. But these mechanisms incur relatively heavy computation overhead, while the timestamp synchronization between the routers and end hosts is also impractical [20]. PPV [21] reduces both computation and communication overhead by probabilistic marking, but it cannot achieve the source verification in the router. OSV [59] creates a more efficient system by using orthogonal sequences instead of the cryptographic operation. It incurs complexity in the control plane, thus lacking the capability of deployability. Meanwhile, vSFC [60], [61] achieves the verification in the cloud to verify the real enforcement of service function chains as expected, extending the verification to a dedicated network. DYNAPFV [62] proposes a lightweight packet forwarding verification mechanism to detect attacks against packet forwarding in SDN. PSVM [63] utilizes Credible Guarantee Agent (CGA) in SDN to perform a flexible source and path verification. These mechanisms are not tailored for inter-domain contexts. In MASK, the verification is a service toward the end host, which offloads most of the router's overhead to the end host and KDS to achieve a practical source and path verification in inter-domain contexts.

IX. Conclusion

Existing source and path verification mechanisms faced a dilemma between security and efficiency in computation and communication overhead. Our efficient MASK resolves this dilemma. In contrast to performing per-packet verification in each entity on the path, MASK offers a dramatically different design point by leveraging the end host's ability to establish a per-session verification, ensuring security and reducing overhead in the router, as well as providing the customized capability to meet the end host's requirements. It suits Internet inter-domain communication and facilitates security guarantees in a dedicated network. In a word, MASK enables on-path routers and the destination to achieve the source and path verification in the data plane efficiently, thus providing a secure foundation for the data plane in the network layer.

References

[1] X. Feng, C. Fu, Q. Li, K. Sun, and K. Xu, "Off-path TCP exploits of the mixed IPID assignment," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2020, pp. 1323–1335.

[2] H. Birge-Lee, Y. Sun, A. Edmundson, J. Rexford, and P. Mittal, "Bamboozling certificate authorities with BGP," in *Proc. 27th USENIX Secur. Symp.*, 2018, pp. 833–849.

[3] J. Zheng, Q. Li, G. Gu, J. Cao, D. K. Y. Yau, and J. Wu, "Realtime DDoS defense using COTS SDN switches via adaptive correlation analysis," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 7, pp. 1838–1853, Jul. 2018.

[4] Q. Li et al., "Dynamic network function enforcement via joint flow and function scheduling," *IEEE Trans. Inf. Forensics Security*, vol. 17, pp. 486–499, 2022.

[5] J. Cao, M. Xu, Q. Li, K. Sun, and Y. Yang, "The loft attack: Overflowing SDN flow tables at a low rate," *IEEE/ACM Trans. Netw.*, to be published.

[6] R. Xie et al., "Disrupting the SDN control channel via shared links: Attacks and countermeasures," *IEEE/ACM Trans. Netw.*, vol. 30, no. 5, pp. 2158–2172, Oct. 2022.

[7] M. Zhang et al., "Poseidon: Mitigating volumetric DDoS attacks with programmable switches," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2020, pp. 1–18.

[8] G. Li et al., "Enabling performant, flexible and cost-efficient DDoS defense with programmable switches," *IEEE/ACM Trans. Netw.*, vol. 29, no. 4, pp. 1509–1526, Aug. 2021.

[9] C. Fu, Q. Li, M. Shen, and K. Xu, "Realtime robust malicious traffic detection via frequency domain analysis," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2021, pp. 3431–3446.

[10] X. Feng et al., "PMTUD is not panacea: Revisiting IP fragmentation attacks against TCP," in *Proc. NDSS*. Reston, VA, USA: ISOC, 2022, pp. 1–18.

[11] X. Feng, Q. Li, K. Sun, C. Fu, and K. Xu, "Off-path TCP hijacking attacks via the side channel of downgraded IPID," *IEEE/ACM Trans. Netw.*, vol. 30, no. 1, pp. 409–422, Feb. 2022.

[12] X. Feng et al., "Off-path network traffic manipulation via revitalized ICMP redirect attacks," in *Proc. 31st USENIX Secur. Symp. (USENIX Secur.)*, 2022, pp. 2619–2636.

[13] C. Fu, Q. Li, M. Shen, and K. Xu, "Frequency domain feature based robust malicious traffic detection," *IEEE/ACM Trans. Netw.*, early access, Aug. 8, 2022, doi: 10.1109/TNET.2022.3195871.

[14] C. Fu, Q. Li, and K. Xu, "Detecting unknown encrypted malicious traffic in real time via flow interaction graph analysis," in *Proc. NDSS*. Reston, VA, USA: ISOC, 2023, pp. 3431–3446.

[15] T. H.-J. Kim, C. Basescu, L. Jia, S. B. Lee, Y.-C. Hu, and A. Perrig, "Lightweight source authentication and path validation," in *Proc. ACM Conf. SIGCOMM*, Aug. 2014, pp. 271–282.

[16] D. Wendlandt, I. C. Avramopoulos, D. G. Andersen, and J. Rexford, "Don't secure routing protocols, secure data delivery," in *Proc. 5th ACM Workshop Hot Topics Netw. (HotNets-V)*, 2006, pp. 1–15.

[17] A. Perrig, P. Szalachowski, R. M. Reischuk, and L. Chuat, *SCION: A Secure Internet Architecture*. Cham, Switzerland: Springer, 2017.

[18] X. Liu, A. Li, X. Yang, and D. Wetherall, "Passport: Secure and adoptable source authentication," in *Proc. 5th USENIX Symp. Networked Syst. Design Implement. (NSDI)*, 2008, pp. 365–376.

[19] J. Naous, M. Walfish, A. Nicolosi, D. Mazières, M. Miller, and A. Seehra, "Verifying and enforcing network paths with icing," in *Proc. 7th Conf. Emerg. Netw. Exp. Technol.*, 2011, p. 30.

[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 *Proc. 29th USENIX Secur. Symp.* Berkeley, CA, USA: USENIX Association, 2020, pp. 541–558.

[21] B. Wu et al., "Enabling efficient source and path verification via probabilistic packet marking," in *Proc. IEEE/ACM 26th Int. Symp. Quality Service (IWQoS)*, Jun. 2018, pp. 1–10.

[22] B. Rothenberger, D. Roos, M. Legner, and A. Perrig, "PISKES: Pragmatic internet-scale key-establishment system," in *Proc. 15th ACM Asia Conf. Comput. Commun. Secur.*, Oct. 2020, pp. 73–86.

[23] D. L. Mills, J. Martin, J. L. Burbank, and W. T. Kasch. *Network Time Protocol Version 4: Protocol and Algorithms Specification*, document RFC 5905, 2010, pp. 1–110.

[24] (2021). *Resource Reports*. [Online]. Available: https://labs.apnic.net/dists/ascc.html

[25] J. McCauley, Y. Harchol, A. Panda, B. Raghavan, and S. Shenker, "Enabling a permanent revolution in internet architecture," in *Proc. ACM Special Interest Group Data Commun.*, Aug. 2019, pp. 1–14.

[26] T. Anderson et al., "The NEBULA future internet architecture," in *The Future Internet—Future Internet Assembly, Validated Results and New Horizons* (Lecture Notes in Computer Science), vol. 7858, A. Galis and A. Gavras, Eds. Berlin, Germany: Springer, 2013, pp. 16–26.

[27] C. Filsfils, P. Camarillo, J. Leddy, D. Voyer, S. Matsushima, and Z. Li. *Segment Routing over IPv6 (SRv6) Network Programming*, document RFC 8986, 2021, pp. 1–40.

[28] D. Dolev and A. Yao, "On the security of public key protocols," *IEEE Trans. Inf. Theory*, vol. IT-29, no. 2, pp. 198–207, Mar. 1983.

[29] P. Bosshart et al., "P4: Programming protocol-independent packet processors," in *Proc. SIGCOMM Comput. Commun. Rev.*, vol. 44, Jul. 2014, pp. 87–95.

[30] N. Foster, N. McKeown, J. Rexford, G. Parulkar, L. Peterson, and O. Sunay, "Using deep programmability to put network owners in control," in *Proc. SIGCOMM Comput. Commun. Rev.*, vol. 50, no. 4, Oct. 2020, pp. 82–88.

[31] M. Hogan, S. Landau-Feibish, M. T. Arashloo, J. Rexford, D. Walker, and R. Harrison, "Elastic switch programming with P4All," in *Proc. 19th ACM Workshop Hot Topics Netw.*, Nov. 2020, pp. 168–174.

[32] B. I. Jun, L. Bingyang, W. U. Jianping, and S. Yan, "Preventing IP source address spoofing: A two-level, state machine-based method," *Tsinghua Sci. Technol.*, vol. 14, no. 4, pp. 413–422, Aug. 2009.

[33] J. Wu, G. Ren, and X. Li, "Source address validation: Architecture and protocol design," in *Proc. IEEE Int. Conf. Netw. Protocols*, Oct. 2007, pp. 276–283.

[34] G. Hu, K. Xu, J. Wu, Y. Cui, and F. Shi, "A general framework of source address validation and traceback for IPv4/IPv6 transition scenarios," *IEEE Netw.*, vol. 27, no. 6, pp. 66–73, Nov./Dec. 2013.

[35] S. E. Deering and R. M. Hinden. *Internet Protocol, Version 6 (IPv6) Specification*, document RFC 8200, 2017, pp. 1–42.

[36] L. Wang, H. Kim, P. Mittal, and J. Rexford, "Programmable in-network obfuscation of traffic," 2020, *arXiv:2006.00097*.

[37] X. Chen, "Implementing AES encryption on programmable switches via scrambled lookup tables," in *Proc. Workshop Secure Program. Netw. Infrastruct.*, Aug. 2020, pp. 8–14.

[38] S. Yao et al., "Blockchain-empowered collaborative task offloading for cloud-edge-device computing," *IEEE J. Sel. Areas Commun.*, early access, Oct. 31, 2022, doi: 10.1109/JSAC.2022.3213358.

[39] J. de Ruiter and C. Schutijser, "Next-generation internet at terabit speed: SCION in P4," in *Proc. 17th Int. Conf. Emerg. Netw. EXperiments Technol.*, Dec. 2021, pp. 119–125.

[40] K. Xu, Y. Zheng, S. Yao, B. Wu, and X. Xu, "NetSpirit: A smart collaborative learning framework for DDoS attack detection," *IEEE Netw.*, vol. 35, no. 6, pp. 140–147, Nov. 2021.

[41] (2021). *As6447 BGP Routing Table Analysis Report*. [Online]. Available: https://bgp.potaroo.net/as6447/

[42] J. Aumasson and D. J. Bernstein, "SipHash: A fast short-input PRF," in *Proc. Prog. Cryptology—INDOCRYPT, 13th Int. Conf. Cryptol. India* (Lecture Notes in Computer Science), vol. 7668. Berlin, Germany: Springer, 2012, pp. 489–508.

[43] P. Zave and J. Rexford, "The compositional architecture of the internet," *Commun. ACM*, vol. 62, no. 3, pp. 78–87, Feb. 2019.

[44] S. Yao, J. Guan, Y. Wu, K. Xu, and M. Xu, "Toward secure and lightweight access authentication in SAGINs," *IEEE Wireless Commun.*, vol. 27, no. 6, pp. 75–81, Dec. 2020.

[45] X. Wang, K. Xu, W. Chen, Q. Li, M. Shen, and B. Wu, "ID-based SDN for the Internet of Things," *IEEE Netw.*, vol. 34, no. 4, pp. 76–83, Jul. 2020.

[46] C. Filsfils, S. Previdi, L. Ginsberg, B. Decraene, S. Litkowski, and R. Shakir. *Segment Routing Architecture*. document RFC 8402, 2018, pp. 1–32.

[47] C. Filsfils, D. Dukes, S. Previdi, J. Leddy, S. Matsushima, and D. Voyer. *IPv6 segment routing header (SRH)*, document RFC 8754, 2020, pp. 1–27.

[48] S. Fu et al., "D3: Lightweight secure fault localization in edge cloud," in *Proc. IEEE 42nd Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2022, pp. 515–525.

[49] R. Hartert et al., "A declarative and expressive approach to control forwarding paths in carrier-grade networks," in *Proc. ACM Conf. Special Interest Group Data Commun.*, Aug. 2015, pp. 15–28.

[50] F. Hauser, M. Schmidt, M. Haberle, and M. Menth, "P4-MACsec: Dynamic topology monitoring and data layer protection with MACsec in P4-based SDN," *IEEE Access*, vol. 8, pp. 58845–58858, 2020.

[51] T. Datta, N. Feamster, J. Rexford, and L. Wang, "SPINE: Surveillance protection in the network elements," in *Proc. 9th USENIX Workshop Free Open Communications Internet (FOCI)*, 2019, pp. 1–7.

[52] A. Yaar, A. Perrig, and D. X. Song, "Pi: A path identification mechanism to defend against DDoS attack," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2003, pp. 1–15.

[53] Q. Li, Y. Liu, Z. Liu, P. Zhang, and C. Pang, "Efficient forwarding anomaly detection in software-defined networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 11, pp. 2676–2690, Mar. 2021.

[54] A. He, K. Bu, Y. Li, E. Chida, Q. Gu, and K. Ren, "Atomos: Constant-size path validation proof," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 3832–3847, 2020.

[55] Q. Li, Y. Chen, P. P. C. Lee, M. Xu, and K. Ren, "Security policy violations in SDN data plane," *IEEE/ACM Trans. Netw.*, vol. 26, no. 4, pp. 1715–1727, Aug. 2018.

[56] B. Parno, A. Perrig, and D. Andersen, "SNAPP: Stateless network-authenticated path pinning," in *Proc. ACM Symp. Inf., Comput. Commun. Secur. (ASIACCS)*, 2008, pp. 168–178.

[57] B. Wu et al., "RFL: Robust fault localization on unreliable communication channels," *Comput. Netw.*, vol. 158, pp. 158–174, Jul. 2019.

[58] L. Ma, K. Bu, N. Wu, T. Luo, and K. Ren, "Atlas: A first step toward multipath validation," *Comput. Netw.*, vol. 173, May 2020, Art. no. 107224.

[59] H. Cai and T. Wolf, "Source authentication and path validation with orthogonal network capabilities," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Apr. 2015, pp. 111–112.

[60] X. Zhang, Q. Li, J. Wu, and J. Yang, "Generic and agile service function chain verification on cloud," in *Proc. IEEE/ACM 25th Int. Symp. Quality Service (IWQoS)*, Jun. 2017, pp. 1–10.

[61] X. Zhang, Q. Li, Z. Zhang, J. Wu, and J. Yang, "VSFC: Generic and agile verification of service function chains in the cloud," *IEEE/ACM Trans. Netw.*, vol. 29, no. 1, pp. 78–91, Feb. 2021.

[62] Q. Li, X. Zou, Q. Huang, J. Zheng, and P. P. C. Lee, "Dynamic packet forwarding verification in SDN," *IEEE Trans. Dependable Secure Comput.*, vol. 16, no. 6, pp. 915–929, Nov. 2019.

[63] F. Yang et al., "I know if the journey changes: Flexible source and path validation," in *Proc. IEEE/ACM 28th Int. Symp. Quality Service (IWQoS)*, Jun. 2020, pp. 1–6.

Songtao Fu received the M.E. degree from the Department of Computer Science and Technology, Sichuan University, Sichuan, China, in 2013. He is currently pursuing the Ph.D. degree with Tsinghua University. His research interests include internet architecture and network security.

Qi Li (Senior Member, IEEE) received the Ph.D. degree from Tsinghua University, Beijing, China. He is currently an Associate Professor with the Institute for Network Sciences and Cyberspace, Tsinghua University. He worked at ETH Zurich and The University of Texas at San Antonio. His research interests include network and system security, particularly in internet and cloud security, mobile security, and big data security. He is the Editorial Board Member of the IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING and *ACM DTRAP*.

Min Zhu received the Ph.D. degree from the Department of Computer Science and Technology, Tsinghua University, Beijing, China, in 2014. She is currently a Senior Engineer at the Department of Computer Science and Technology, Tsinghua University. Her research interests include internet architecture and network security.

Xiaoliang Wang received the Ph.D. degree from the Department of Computer Science and Technology, Tsinghua University, Beijing, China, in 2017. He is currently a Lecturer at the Information Engineering College, Capital Normal University. His research interests include network architecture and network security.

Su Yao received the Ph.D. degree from the National Engineering Laboratory for Next Generation Internet Interconnection Devices, Beijing Jiaotong University, Beijing, China. Currently, he serves at the Beijing National Research Center for Information Science and Technology (BNRist), Tsinghua University, as an Assistant Research Fellow. His research interests include future network architecture, the IoT security, and artificial intelligence for network systems.

Yangfei Guo received the M.E. degree from the Institute for Network Sciences and Cyberspace, Tsinghua University, Beijing, China, in 2022. He is currently working at the Zhongguancun Laboratory. His research interest includes inter-domain source address validation.

Xinle Du received the B.E. degree from the Department of Computer Science and Technology, Xidian University, Xi'an, China, in 2014. He is currently pursuing the Ph.D. degree with Tsinghua University. His research interests include data-driven networks, data center network transport protocol, and AQM.

Ke Xu (Senior Member, IEEE) received the Ph.D. degree from the Department of Computer Science and Technology, Tsinghua University, Beijing, China. He serves as a Full Professor at Tsinghua University. He has published more than 200 technical papers and holds 11 U.S. patents in the research areas of next-generation internet, blockchain systems, the Internet of Things, and network security. He is a member of ACM. He served as the Steering Committee Chair for IEEE/ACM IWQoS. He has guest-edited several special issues in IEEE and Springer journals. He is the Editor of IEEE INTERNET OF THINGS JOURNAL.