This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2024.3392486

Secure Fault Localization in Path Aware Networking

Songtao Fu, Qi Li, *Senior Member, IEEE*, Xiaoliang Wang, Su Yao, Xuewei Feng, Ziqiang Wang, Xinle Du, Kao Wan and Ke Xu, *Fellow, IEEE*

Abstract—Secure data forwarding is critical for users to meet their requirements. In this paper, we propose D3 (Demon Detector in Data Plane), a source-driven, secure fault localization mechanism, which empowers the source to localize faulty link in Path Aware Networking, thus circumventing faulty link to guarantee secure data forwarding. D3 utilizes the source to instruct the on-path routers, thus empowering it to detect whether the on-path routers forward the packet as expected. Compared with existing schemes that are difficult to be deployed in practice due to the heavy storage, computation, and communication overhead, D3 offloads most of the on-path 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, thus having a low communication overhead. Besides that, the destination in D3 could keep stateless processing, thus having backward compatibility and eliminating the opportunity for DoS attacks toward a stateful destination. The BMv2 and Barefoot Tofino hardware evaluations show that D3 could achieve high fault localization accuracy and process the packet at line rate.

Index Terms—Path Aware Networking, Data Plane, Fault Localization.

1 INTRODUCTION

THE promise and potential of applications, ranging from Internet-of-Things (IoT) to Vehicular Networks [1] [2], have prompted users to be concerned about whether the network forwards the packet according to their requirements since the network protocols with flawed designs have been found to be a major cause of security issues [3]–[15]. As an Autonomous System (AS) is a trust and fate-sharing unit [16], we denote an AS as a domain. In an inter-domain context, the packet from the source to the destination user (e.g., end host) transits several ASes. The emerging Path Aware Networking (PAN) [17] [18] could facilitate the source to select the forwarding path. The PANs' architecture comprises two essential components: i) the neighboring nodes exchange path information in the control plane (with low-bandwidth, large storage and computation resources), and ii) the data packets (DAT) are forwarded in the data plane (with high-bandwidth, limited storage and computation resources [19]) along the discovered paths. The paths are exposed to hosts (in SCION [17]) or routers (in SR [18]), allowing them to select a path and embed it in the packet header [20]. Generally, the source-selected paths are a forwarding policy that each on-path router should follow. But an adversary in a specific AS might violate the forwarding policy. For example, a malicious AS might corrupt the packet by dropping, delaying, modifying, fabricating the packets, or redirecting the packets to an unexpected AS [21]. The source on the Internet needs a fault localization mechanism to localize the faulty link[1], then circumvent the faulty link to achieve secure data forwarding.

The fault localization mechanisms like secure traceroute provide some assistance in debugging network problems [22]. A drawback of them is that they only give feedback on probe packets, not data packets ($DATs$). Neither the destination nor the routers could verify the DAT and reply to the source with the feedback. On the other hand, fault localization mechanisms (e.g., AudIt [23] and Fatih [24]) aim to localize the faulty entity based on the DAT, lack the authentication of the DAT. As a result, they fail to meet the security requirements of the $detector$ (e.g., source or other entities) since on-path routers are unable to verify a Message Authentication Code (MAC) embedded in the packet by the $detector$.[2] To prevent the adversary from manipulating the packet, the on-path router needs to store the fingerprints (e.g., the hash) of packets, but it incurs prohibitively expensive storage overhead in the router.

There are some secure fault localization mechanisms, which not only localize the faulty link based on the DAT

- *Songtao Fu, Xuewei Feng, Xinle Du and Ke Xu are with the Department of Computer Science and Technology, Beijing National Research Center for Information Science and Technology (BNRist), Tsinghua University, Beijing, 100084, China, Ke Xu is also with Zhongguancun Laboratory, Beijing 100094, China (e-mail: {fust18, fengxw18, dxl18}@tsinghua.org.cn, xuke@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 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)*
- *Ziqiang Wang is with the Southeast University, Nanjing, 211189, China (e-mail: ziqiangwang@seu.edu.cn)*
- *Kao Wan is with the Peng Cheng Laboratory, Shenzhen, 518000, China (e-mail: sonicwk@hotmail.com)*

*Manuscript received **, 2022. This work was in part supported by the National Key R&D Program of China with No. 2022YFB3102301, 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. BJJWZYJH01201910003011. (*Corresponding authors: Ke Xu and Su Yao.*)*

1. The faulty link includes two neighbor AS border routers and the link between the two routers, at least one border router, or the link is faulty.

2. This paper calls the source user who localizes the faulty link on the forwarding path as $detector$. The fault localization is nearly the same as the failure localization [25]. Generally, the source or destination user is the end host.

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on January 14,2025 at 07:11:02 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2024.3392486

but also incorporate a per-packet MAC (i.e., $mark$ in this paper) to ensure a strong security guarantee. ShortMAC [26] is a secure fault localization mechanism, but it could not be utilized in the inter-domain context due to the extremely heavy storage overhead in the data plane. Faultprints [27] and RFL [28] are secure fault localization mechanisms in the inter-domain context. They have relatively low storage overhead in the data plane (e.g., 46.8 MB per 10 Gbps link). However, they still have high computation overhead in the data plane.

D3 (Demon Detector in Data Plane) in edge cloud [29] could meet the localization requirements between the servers in the edge and remote cloud with lightweight overhead. However, the requirements for secure fault localization in PAN differ from those in the edge cloud [30]. Consequently, *D3 in edge cloud* could not be utilized in PAN for the following reasons: (1) It lacks backward compatibility, impeding the ability to initiate localization with a legacy destination host as it assumes the destination has been upgraded. (2) It lacks the scalability to support a large number of source hosts since it presets the dynamic key [3] of source hosts in each on-path router, which incurs a high storage overhead. (3) It lacks robustness since it allows adversaries to launch DoS attacks toward the stateful destination.

In this paper, we propose a source-driven secure fault localization mechanism named D3 via extending the design of *D3 in edge cloud*. Compared with *D3 in edge cloud*, our contributions are as follows:

- We utilize a source-driven fault localization mechanism to maintain **backward compatibility** with legacy routers and destination hosts. D3 has the potential to empower the source to locate the faulty link solely depending on routers with the capability required by the D3 design. Furthermore, the source could achieve fault localization without the participation of the destination host, thus improving the **robustness** by preventing the DoS attacks towards the stateful destination host.
- We utilize a hierarchical key distribution system to decrease the storage overhead, thereby enhancing **scalability**. An edge router (named agent) manages the source hosts in the same subnet. Each on-path router only needs to store the dynamic keys for the agents, resulting in significantly lower storage overhead concerning dynamic keys in the data plane. Additionally, D3 has the same $proof$[4] storage overhead, communication overhead (20 B additional packet header), and computation overhead (one lightweight symmetric cryptographic operation) in the data plane compared to *D3 in edge cloud*.
- **Evaluation in BMv2 and Barefoot Tofino hardware.** The evaluation in BMv2 testifies that the localization accuracy could achieve 95%. Meanwhile, the evaluation in Barefoot Tofino hardware testifies that D3 could update $10,000$ dynamic keys within 1 second, and process the packet at line rate.

The rest of the paper organizes as follows: In the next section, we present the background, adversary model, and our assumptions. Sec. 3 presents the design of D3 at a high level, Sec. 4 details the D3, and sec. 5 analyzes its characteristics and benefits. Sec. 6 presents the implementation and the evaluation. Sec. 7 discusses the deployment of D3. Sec. 8 describes related works. We conclude the paper in the last section.

3. In this paper, the dynamic key is the symmetric key to calculate the $mark$.

4. The $proof$ is the statistic information in each on-path router, e.g., the counter of packets in an epoch.

2 PROBLEM SETTING

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

2.1 Background

To guarantee secure data forwarding, the source in the PAN could achieve secure fault localization based on the actual data traffic.

Barak et al. elaborate on the significance of a secure fault localization's key infrastructure, as highlighted in their work [25]. Specifically, secure fault localization plays a crucial role in defending against adversaries through the utilization of a symmetric key shared between the $detector$ (e.g., the source) and each on-path router (i.e., AS border router). The key infrastructure facilitates a confidential means of sharing dynamic keys between the routers and the $detector$. In essence, the $detector$ can localize the faulty link based on the $proof$ provided by each on-path router's reply packets. The policy governing these $proofs$ is pivotal for optimal performance. Barak et al. delve into the overhead associated with two $proof$ policies: per-packet and statistical fault localization. In per-packet fault localization, the destination and each router respond to the source with an ACK containing nested onion marks, resulting in significant computation and communication overhead. On the other hand, statistical fault localization (e.g., Symmetric Secure Sampling) offers a lower overhead by having the destination and each router reply to the source for portions of packets at specified intervals (referred to as epochs in this context). However, each on-path router in Symmetric Secure Sampling must store packet segments and respond to the source, leading to notable computation and storage overhead. To address these issues, ShortMAC [26] validates each packet using short marks and maintains flow counters as $proof$ in the data plane. It transmits the counters to the source at the end of each epoch. Despite these efforts, there remains heavy storage overhead in inter-domain scenarios, reaching up to 46 GB for a 100 Gbps link in extreme cases.

Faultprints [27] is the first secure fault localization in an inter-domain context. Fig. 1 (a) presents its basic procedure. Generally, The procedure enforced by the source (S) includes: ① Data (DAT) forwarding: S instructs each on-path router to record the $proof$ (A Bloom Filter to record whether processed a specific packet). ② End-to-end corruption ratio with ACK: 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). ③ $probe\ and\ reply$: If the end-to-end corruption ratio indicates there might be an adversary on the path, the source receives the $proof$ with the $probe\ and\ reply$

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on January 14,2025 at 07:11:02 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2024.3392486

TABLE 1
The properties of different fault localization mechanisms

	Storage overhead	*No ASY in data plane*	*Additional packet header*	*Inter-domain support*
ShortMAC [26]	46000	✗	2	✗
Faultprints [27]	468	✗	96	✓
RFL [28]	468	✗	58	✓
D3 in edge cloud [29]	$3.84 \times N$	✓	20	✓
D3	$3.84 \times L$	✓	20	✓

1. *Storage overhead* represents the data plane storage overhead (MB) per 100 Gbps link, *No ASY in data plane* represents no per-session [5] asymmetric cryptographic operation in data plane, *Additional packet header* represents the length of the additional packet header (B) with the path length of 5, *Inter-domain support* represents supportting inter-domain context.
2. N represents the number of hosts in an AS, L represents the number of agents in an AS, $L << N$.

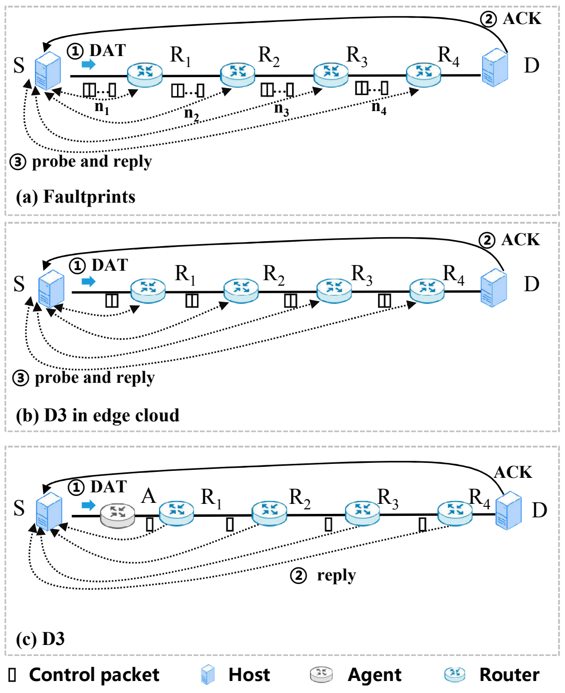

Fig. 1. An example of fault localization in different mechanisms. S and D represent the source and destination host, R_i represents an ingress border router of an AS, A represents an agent, which is an edge router of a subnet, n_1, n_2, n_3 *and* n_4 represent the number of control packets ($probe$ *and* $reply$ packets).

mechanism towards each router. The number of control packets ($probe$ and $reply$ packets) for R_1, R_2, R_3, R_4 are n_1, n_2, n_3 *and* n_4 respectively. Therefore, there have $O(n)$ control packets. An example of Faultprints operation with 4 ASes on the forward path includes: The source sends 4000 packets per epoch in the data plane, and instructs the on-path router to sample a packet (store whether it processed a packet in Bloom Filter) according to the $marks$ nested in the packet header. If the source receives a valid ACK within a certain timeout, e.g., a round-trip time (RTT), the source does nothing. Or else, the source sends 400 $probe$ packets (10% of 4000 DAT packets) along the forward path and receives 1600 $reply$ packets. The source localizes the faulty link based on the $reply$ packets. RFL has a similar procedure except that it records the $proof$ (Bloom Filter) for each flow, which incurs a little more $proof$ storage overhead, and $probe$ *and* $reply$ once at each epoch with asymmetric cryptographic operation, which incurs much more computation overhead.

In general, state-of-the-art secure fault localization mechanisms in PAN have limitations for the following reasons: Firstly, they are all incompatible with legacy destinations, making it nontrivial to upgrade each destination the source wants to communicate with. Secondly, as indicated in TABLE 1, they entail heavy storage (e.g., 46 GB per 100 Gbps link in ShortMAC) and computational overhead (e.g., Faultprints and RFL require per-session asymmetric cryptographic operations) in each router's data plane. Thirdly, the communication overhead is relatively high (e.g., The additional packet headers of Faultprints and RFL are 96 B and 58 B for a path length of 5, respectively). Lastly, there is little incentive for ISPs to upgrade their routers (e.g., MAC operations such as AES are challenging to implement in router hardware [31]).

The insight of *D3 in edge cloud* is it separates the control plane and data plane in terms of cryptographic operation and $proof$ storage. Compared with Faultprints, it decreases the computation overhead since the on-path router prepares the dynamic key for the source (e.g., servers in edge or remote cloud), then it does not need to achieve per-session asymmetric cryptographic operation in data plane. Besides that, it offloads the $proof$ storage ($Counter$ information) overhead to the control plane.

As shown in Fig. 1 (b), an example of *D3 in edge cloud* [29] compared with Faultprints is as follows: The source sends 4000 DAT packets at each epoch, and nests the instruction in the $mark$ to instruct the on-path router to count the number of sampling packet. Each on-path router only sends the packet header to the control plane after a lightweight verification if it should sample a specific packet. In the event that the source does not receive an ACK from the destination following a timeout (e.g., RTT), it dispatches 4 $probe$ packets towards the 4 on-path routers and retrieves 4 $reply$ packets. Subsequently, it localizes the faulty link. It generates only $O(1)$ control packets per epoch.

However, *D3 in edge cloud* lacks the following properties: (1) backward compatibility, (2) scalability, and (3) robustness. It could not be directly implemented in PAN due to these limitations. Therefore, D3 improves the design of *D3 in edge cloud* as follows: (1) Source-driven localization to keep backward compatibility. D3 does not need the ACK and $probe$. It achieves the localization only based on each on-path router's $reply$ at each end of an epoch. (2) Hierarchical architecture to support scalability. In D3, an agent first verifies the source's instruction with the symmetric key between the source and the agent, then updates the instructions with the dynamic key between the agent and the on-path router. Each on-path router only needs to store the dynamic key for the agents. (3) Flexible localization to improve the robustness. In *D3 in edge cloud*, the destination

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on January 14,2025 at 07:11:02 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2024.3392486

needs to keep stateful for all the sources. It gives the chance for an adversary to launch the DoS attack in PAN. The destination in D3 keeps stateless to avoid the DoS attack from the adversaries, it could also keep stateful to send the ACK with the end-to-end corruption ratio for some authenticated source (e.g., in the same enterprise) to defend against the coward attacks.

As shown in Fig. 1 (c), an example of D3 compared with *D3 in edge cloud* is as follows: The source sends 4000 DAT packets at each epoch, and nested the sampling instruction in the $mark$. The agent (an edge router of a subnet) updates the $mark$ and instructs the on-path router to sample a specific packet (recording the $Counter$ of each sampling packet) after successful verification. In order to reduce $proof$ storage overhead in the data plane, considering that each epoch takes hundreds of milliseconds, D3 records per-session counters in the control plane. Each on-path router sends the packet header to the control plane after a lightweight verification and sends 4 $reply$ packets with $Counter$ information to the source at each end of an epoch. Then the source localizes the faulty link based on the 4 $reply$ packets.

2.2 Adversary model

Considering a network consisting of (1) key distribution server (KDS), (2) host (source and destination), (3) agent, and (4) on-path routers. We assume that the KDS, host, and agent are honest. It seems like a strong assumption since the KDS, host, and agent could also be compromised. But the scope is limited since it only affects the security between itself and other entities. In the case that the destination host is legacy, the last entity is the destination border router. We also assume the last entity is honest. It is reasonable since the destination router has little benefit to being faulty. For example, a faulty destination router receives 500 packets, but it only replies to the source with the counter information of 200. The source then localize the link between the penult on-path router and the destination as faulty and select another penult on-path router. But in PAN, the destination router could negotiate the forwarding path in the control plane, it is not necessary for the destination to reply with the wrong information.

In the case that the destination router and host are legacy, e.g., R_4 and D in Fig. 1 do not support the localization, R_3 is the last entity. The last entity does not need to be honest. At first glance, there is no reply from destination AS, R_3 might drop the packets and reply with the correct counter information if it knows that it is the last entity on the path. But it does not hurt the localization for the following reasons: Firstly, R_3 does not know whether the destination is legacy or upgraded. Secondly, considering that R_3 could evade the localization, the source could get the end-to-end performance from the upper layer as a reference (e.g., the performance of TCP), to defend against the attack from the last entity.

An adversary in the Dolev-Yao model can compromise any on-path routers. 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 an unexpected AS. They can also launch a coward attack if they know they cannot be accurately localized, or frame other ASes. Furthermore, several colluders could exchange information (e.g., secret keys or link information) and launch the above-mentioned attacks. We summarize various types of attacks as follows:

Packet corruption. An adversary might corrupt any part of the packet, e.g., the source address in the packet header, or delay, drop a packet, inject a fabricated packet toward the destination.

Path inconsistency. An adversary might forward the packet along the arbitrary alternative path not the same as the expected path selected by the source.

Coward attack. An adversary might launch the coward attack [32], e.g., only drop the packets it sampled since that other routers on the path did not sample them.

Framing attack. The adversary might incriminate other routers by packet corruption (e.g., change other router's $mark$) or path inconsistency (e.g., redirect the packet).

Colluding attack. Colluders might exchange the dynamic key or statistical information to avoid or violate the localization [27], e.g., framing other ASes.

Replay attack. The adversary might launch a replay attack [33], which waste the network resources or brings more serious problems at the upper layer.

New DoS attack. The adversary might aggressively send packets to put a computational strain or storage overhead to overwhelm the router or host [27].

2.3 Requirements and assumptions

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

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

High fault localization accuracy. For a certain number of epochs, we denote the localization accuracy as $accuracy = K/N$, in which K is the number of epochs the source successfully localizes the faulty link (without false positive and false negative), N is the total number of epochs. A localization accuracy higher than 95% is practical [27].

We assume that the source knows an AS level path to the destination. And the 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. Each pair of KDS exchange the AK_{SI} with asymmetric cryptography in the control plane (e.g., with DH key exchange [16]). The KDS could derive the router's dynamic key (RDK) and the host's dynamic key (HDK) with a MAC operation (e.g., AES) based on the AK_{SI} in the control plane.

Secure dynamic key distribution. The host and router have a secure channel to get the dynamic keys from KDS in the local AS [34] [35]. Each router presets the dynamic keys from the control plane, thus decreasing the computation overhead to derive the dynamic key on the fly. The agent could also securely distribute the symmetric key between itself and the source host.

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on January 14,2025 at 07:11:02 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2024.3392486

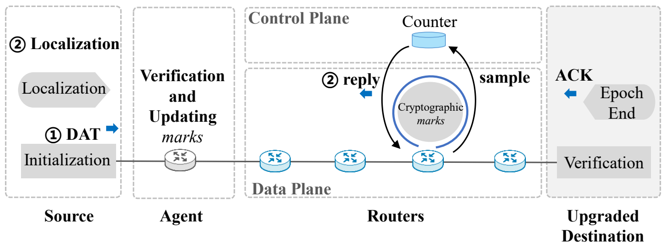

Fig. 2. Processing at different entities.

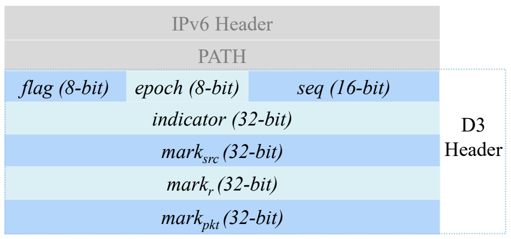

Fig. 3. D3 packet header in IPv6 packet.

3 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 on-path router. With the cryptographic $marks$ in each packet (section 4.1), D3 achieves fault localization with 2 procedures: ① Data (DAT) forwarding, and ② Localization based on $reply$.

① Data forwarding. The data forwarding includes packet initialization at the source and the agent (section 4.2), as well as lightweight processing at the router (section 4.3). During packet initialization, the source embeds instructions in the $marks$ and transmits a specific number of $DATs$ to the destination in each epoch. The agent verifies the $marks$ and updates the $marks$ in the packet header. In terms of lightweight processing, each stateless router only processes the Cryptographic $marks$ in the data plane. If the Cryptographic $marks$ instruct the router to sample the packet, it records the $Counter$ as $proof$ in the control plane. The D3-supported destination verifies each packet to filter the malicious traffic and records the end-to-end $Counter$ information of the session (section 4.5). Notably, the upgraded (D3-supported) destination may transmit an ACK containing the end-to-end $Counter$ information back to the source, enabling the source to achieve more robust localization.

② Localization based on $reply$. At each end of an epoch, each on-path router replies to the source with a $reply$ containing $Counter$ information. Subsequently, the source accomplishes fault localization (section 4.4). It localizes the faulty link based on $replies$. As shown in Fig. 1 (c), each on-path router has 1 $reply$ towards the source.

We utilize the additional packet header of D3 as part of the routing header in the IPv6 packet [36]. Fig. 3 shows our designed D3 packet header in IPv6 packet. We utilize the two bits of $flag$ to indicate the length of $mark_r$ for each on-path router, $epoch$ and seq are the epoch number and sequence number of a specific packet respectively, $indicator$ is the tag to instruct each router to select the relative RDK, includes the time slot (16-bit ts) and the user (agent) identifier (16-bit uid). Furthermore, the packet header contains 3 $marks$:

- $mark_{src}$ (32-bit): The $mark$ calculated by the source, then updated by the agent after verification, which instructs a selected on-path router to sample the packet.
- $mark_r$ (32-bit): The $mark$ calculated by the agent which instructs the on-path routers to verify and filter the packet.
- $mark_{pkt}$ (32-bit): The $mark$ calculated by the source and verified by the destination.

4 PROTOCOL DESIGN

We first detail how to calculate the cryptographic $marks$ 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 $reply$.

4.1 Cryptographic $marks$ in each packet

The $marks$ are calculated by a specific MAC operation with RDK_{si} and HDK_{sd}, to guarantee that an adversary could guess the correct $marks$ with negligible probability. The KDS initially generates the dynamic key between the source and router i using the fixed key (AK_{SI}) shared between the source's AS and router $i's$ AS:

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

In which $MAC(.)$ 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. The $indicator$ is an identifier to distinguish the time slot (ts) and the user (uid), "||" represents the concatenation operation.

We could adjust the key switching frequency with ts. A 16-bit length 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.

Therefore, we could nest the ts in the packet header to instruct the dynamic key switching. The KDS sends the AK to the routers, each router could derive the key of two time slots (ts_{i-1} and ts_i) in the control plane, and gets the relative key with the different ts in the packet header. After one time slot, it clears the key of ts_{i-1} and derives the key of ts_{i+1}. Therefore, the synchronization with the granularity of the second in the control plane is enough for D3.

The length of uid (K) 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 hosts in an AS utilize the same dynamic key in a specific time slot, allows an adversary to get the dynamic key towards a specific router i, then subverts the secure localization. We utilize a hierarchical mechanism to overcome it.

In our hierarchical mechanism, the agent is an edge router of one group (with the exact security requirement)

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on January 14,2025 at 07:11:02 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2024.3392486

which has a specific uid, e.g., the edge router of a subnet. The KDS could only distribute the dynamic key toward the agent instead of the host. With an intra-domain secure channel, the host sends the D3 packet to the agent. The agent processes the packet header with the dynamic keys between itself and the on-path routers.

With this design, an AS could customize its dynamic key policy with other ASes, including the length of uid and the switch step of ts. Each on-path router then presets the dynamic keys in the control plane.

The agent and the router could get the fixed key from the KDS and derive the symmetric key in the control plane. Each agent has 1 symmetric key SK_u with each source host which needs to achieve fault localization. The SK_u between the source and the agent could also switch with the ts in the packet header. For example, $SK_u = MAC_{SK}(IP_s||ts)$, in which SK is the fixed key, and IP_s is the IP address of the source.

With SK_u and RDK_{si}, the source and the agent instruct each router to probabilistically sample a packet with $marks$. The agent calculates one MAC operation to get a $mark$ for an on-path router. The input of the MAC includes the link information (id_{i-1}) and CST_{PH}. CST_{PH} is the constant packet header of D3, which includes the source IP address, the path information, the $flag$, $epoch$, seq, and $indicator$. It could also include a partial packet payload for packet integrity checking. Router i and agent calculate $mark_i$ with Equation (2).

$$mark_i = MAC_{RDK_{si}}(id_{i-1}||CST_{PH}) \tag{2}$$

The KDS calculates the dynamic key (HDK_{sd}) between the source and destination:

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

In which AK_{SD} is the AK between the source AS and destination AS, $IP||path||TS$ represents a session, IP is the concatenation of the source and the destination's IP addresses, $path$ is the concatenation of all the routers' id on the path, TS is the information of the first time slot in a session. With the HDK_{sd}, the source calculates the $mark_{pkt}$ as Equation (4), where PL denotes the initial four bytes of the packet payload and ins_{pkt} represents an instruction guiding the packet type, which will be elaborated on in Sec. 4.2. The $[0:32]$ represents truncating the rightmost 32-bit, ensuring that the destination can authenticate the source and verify the packet's integrity.

$$\begin{aligned} mark_{pkt} = & MAC_{HDK_{sd}}(CST_{PH}||PL)[0:32] \\ & + ins_{pkt} \end{aligned} \tag{4}$$

4.2 Packet initialization at source and agent

As shown in Algorithm 1, the source first gets the $path$ and SK_u, HDK_{sd}, determines the number of packets in an epoch, e.g., $N = 4000$, and initializes a counter for each on-path router with $Counter_{src}[I] = \{0\}$. 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 $marks$ ($mark_{pkt}, mark_{src_a}, mark_{r_a}$), and nests them in the packet header (line 5). It increases the $Counter_{src}[i]$ if it instructs

Algorithm 1: packet initialization at source

```
input : $path, SK_u, HDK_{sd}, N, Counter_{src}[I] = \{0\}$
1 for each epoch do
2     $seq = 0$
3     while $seq < N$ do
4         nest $CST_{PH}$ in the packet header
5         calculate and nest marks
6         if router $i$ sample the packet then
7             $Counter_{src}[i]++$
8         forward packet and $seq = seq + 1$
9     epoch = (epoch+1)%256
```

the on-path router i to sample the packet (line 6-7). Then it forwards the packet to the first router and increases the seq (line 8). At the end of each epoch, the source processes the packet in the next epoch if it needs to send another packet (line 9).

If the source selects router i to sample the packet, assuming that $mark_{sa} = MAC_{SK_u}(CST_{PH})$, it nests the $mark_{src}$ and $mark_r$ in the packet header as:

$$\begin{aligned} mark_{src_a} &= mark_{sa}[0:32] + ins_{sa} \\ mark_{r_a} &= mark_{sa}[32:64] \end{aligned} \tag{5}$$

The ins_{sa} is an instruction that indicates the sampling policy. Particularly, $ins_{sa} == n$ represents a DAT that does not need to be sampled (DAT_{UNS}), $ins_{sa} == 0, 1, 2 \ldots n-1$ represents a DAT that needs hop i to sample (DAT_{SAM}).[6] We name this mechanism the $blind\ policy$ since only the source, agent and the selected router know the sampling policy. An adversary could not distinguish the packet type from the confidential $mark$, thus only corrupting these packets with negligible probability. We utilize $mark_{r_a}$ to extend the $mark$ to 64 bits.

The source nests the ins_{pkt} in $\boxed{mark_{pkt}}$ to indicate the sampling policy as Equation (4). Particularly, $ins_{pkt} == 0, 1, 2 \cdots n-1$ represents a DAT that needs hop i to record the sampling counter ($Counter[i]$) at the destination, $ins_{pkt} == n$ represents a DAT_{UNS}, and $ins_{pkt} == n+1$ represents an ACK or a $reply$. It is worth mentioning that the $\boxed{mark_{pkt}}$ is unnecessary when communicate with legacy destination, and the source nests $\boxed{mark_{pkt}}$ with a random string.

Algorithm 2: processing at agent

```
input : $pkt, SK_u, RDK_{path}$
1 calculate $ins_{sa} = \boxed{mark_{src_a}}$-$mark'_{src_a}$ and $mark_{r_a}$
2 if $ins_{sa} <= n$ and $\boxed{mark_{r_a}}$==$mark'_{r_a}$ then
3     calculate $mark_{src_i}, mark_{r_i}$ with $RDK_{si}$
4     $\boxed{mark_{src}} \leftarrow mark_{src_i}$, $\boxed{mark_r} \leftarrow mark_{r_i}$
5 else
6     drop the packet
```

The agent has the SK_u and RDK_{path} (RDK_{si} of all the on-path routers). After receiving a D3 packet

6. We record the counter of sampling packets as $proof$.

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on January 14,2025 at 07:11:02 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2024.3392486

from the source in the same subnet, it first calculates the $mark'_{src_a} = MAC_{SK_u}(CST_{PH})[0:32]$ and $mark'_{r_a} = MAC_{SK_u}(CST_{PH})[32:64]$, and calculates ins_{sa} with $\boxed{mark_{src_a}}$-$mark'_{src_a}$ (line 1). If $ins_{sa} <= n$ and $\boxed{mark_{r_a}}$==$mark'_{r_a}$, the agent calculates the $mark_i$ as Equation (2), and the $mark_{src_i}$ with Equation (6). It is worth mentioning that the agent randomly selects hop i if $ins_{sa} == n$.

$$mark_{src_i} = mark_i[0:32] + ins \tag{6}$$

In which ins is an instruction to instruct each on-path router to sample the packet. Particularly, $ins == 0$ (SAM) indicates that hop i needs to sample the packet and verify the packet (DAT_{SAM}), $ins == 1$ (UNS) indicates that hop i only needs to verify the packet (DAT_{UNS}).

The $mark_{r_i}$ is calculated with Equation (7). The agent updates the two $marks$ as $\boxed{mark_{src}}$ and $\boxed{mark_r}$ in the packet header (line 2-4). If the packet is illegitimate, the agent drops the packet (line 5-6).

$$mark_{r_i} = mark_i[32:64] \tag{7}$$

The agent could utilize $mark_r$ to facilitate each on-path router to filter the malicious traffic. We could utilize the rightmost 2 bits of $flag$ to indicate the length of $mark_{r_i}$ for each hop. The $flag$ of "0,1,2,3" indicates the "2,4,8,32" bits of $mark_{r_i}$ respectively. As the average path length at AS level is less than 5, and the majority path length level is less than 10,[7] 4 bits $mark_{r_i}$ could satisfy that each hop on the path could filter the malicious packet. In this case, $mark_r = mark_1[32:36]||\cdots||mark_n[32:36]$. The downside is that the agent needs to calculate n MAC operations for n hops. Therefore, we utilize 32 bits $mark_{r_i}$ as default, to keep lightweight processing at the agent.

With this design, the agent needs not process the control packets ($reply$ and ACK), and only performs 2 MAC operations to calculate $mark_{src_a}$ and $mark_i$. Considering that the agent drops the illegitimate packet after the first MAC operation, and the inter-domain traffic in an agent is relatively lower than the AS border routers, it incurs little computation overhead on the agent. We will detail in Sec. 6 that the throughput of an agent is sustainable.

4.3 Lightweight processing at router

Algorithm 3: processing at on-path router

```
Input : $pkt, RDK_{si}, id_i, id_{i-1}$
1 calculates $mark'_i$
2 $ins = \boxed{mark_{src}} - mark'_{src_i}$
3 if ($ins == UNS$) or ($flag[0:2]! = 3$) then
4     verify $\boxed{mark_{r_i}}$
5 else if $ins == SAM$ then
6     verify $\boxed{mark_{r_i}}$ and $Counter++$
7 forward packet to next router
```

Each router keeps one $Counter$ for the sampling packet (DAT_{SAM}) in the control plane. As shown in Algorithm 3, when receiving a packet, each router first calculates $mark_i{}'$ with Equation (2) (line 1). Then it derives $mark'_{src_i} = mark_i{}'[0:32]$, and calculates the ins (line 2). It then processes the packet with different values of ins (line 3-6).

For DAT_{UNS} or each on-path router need to verify the packet ($flag[0:2]! = 3$), it verifies the $\boxed{mark_{r_i}}$ with $mark_{r_i}{}'$. The $mark_{r_i}{}'$ is calculated according to the $flag[0:2]$, e.g., $mark_{r_i}{}' = mark_i{}'[32:64]$ if $flag[0:2] == 3$, $mark_{r_i}{}' = mark_i{}'[32:36]$ if $flag[0:2] == 1$. We set $flag[0:2] = 3$ as default. For DAT_{SAM}, it verifies the $\boxed{mark_{r_i}}$ with $mark_{r_i}{}'$ and clones the packet header (the IP, PATH, and D3 packet header) to the control plane. The control plane increases the $Counter$ in the relative epoch. Meanwhile, it forwards the packet to the next router (line 7).

At each end of an epoch (e.g., after receiving 200 packets of the new epoch), the control plane sends a $reply$ with the information of the $Counter$ to the source. As $reply$ is a control packet, the on-path router only specifies the path information in the packet header. It does not embed the $\boxed{mark_{src}}$ and $\boxed{mark_r}$ for other on-path routers. In practice, router i utilizes the $\boxed{mark_{src}}$ and $\boxed{mark_r}$ to load the $Counter$ information, and set $flag[0:2] = 3$. From this perspective, D3 could work well in both symmetric and asymmetric paths.

The $reply$ from router i utilize router i's address as the default source IP address. But there comes a challenge that an adversary might distinguish the $reply$ with the source address (the router's address), then corrupt this $reply$. To prevent this, we modify the source address's suffix with a random string. Router i utilizes a specific virtual IP address for a session. The adversary could not distinguish it from the DAT with the source address. The router then utilizes the new virtual source address to derive the dynamic key (HDK_{is} between router i and the source) with Equation (3), and utilizes the HDK_{is} to calculate $mark_{pkt}$ with Equation (4), and $ins = n + 1$ indicates its a $reply$.

To prevent the adversary observe the $Counter$ information, router i nests the $Counter$ information in the $mark_r$:

$$mark_r = ENC_{HDK_{is}}(Counter) \tag{8}$$

In which ENC indicates the encryption with the HDK_{is}. In practice, router i could utilize $mark_r = MAC_{HDK_{is}}(CST_{PH}||PL)[32:64] \oplus Counter$, in which PL is a random string. This design allows router i and the source to perform only one MAC operation with Equation (4). After receiving the $reply$, the source first requires the HDK_{is} from the KDS. It can then store the HDK_{is} locally for any subsequent $reply$ from router i.

4.4 Fault localization at source

With the $reply$ from each on-path router, the source could localize the faulty link even if the destination is legacy.

At each end of an epoch, it decrypts the $Counter$ information if it receives a $reply$, and calculates the reputation of each on-path router. It calculates the reputation with $repu_i = Counter/Counter_{src}[i]$. The reputation gap is calculated as Equation (9). The gap between two neighbor routers larger than the gap caused by natural packet loss (e.g., 1%) possibly signals the faulty link.

$$gap = repu_i - repu_{i+1} \tag{9}$$

7. https://bgp.potaroo.net/as6447

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on January 14,2025 at 07:11:02 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2024.3392486

The setting of the threshold of gap is based on the source's requirement, e.g., an application requires an end-to-end corruption ratio of no more than 5%. 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 $\rho = 5\%$, the reputation gap between R_2 and R_3 is within $\rho \pm \rho/2$ (e.g., $5\% \pm 2.5\%$), 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 6.

4.5 Fault localization with D3-supported destination

As we described in Sec. 1, the legacy destination of D3 needs to do nothing about the localization. But the D3-supported destination could benefit the source to get a comprehensive end-to-end forwarding performance. From this perspective, we design the processing at the D3-supported destination.

Algorithm 4: processing at D3-supported destination

```
Input : $pkt, HDK_{sd}$
1  initialize $sCounter = 0, Counter[I] = \{0\}$
2  calculate $ins_{pkt} = \boxed{mark_{pkt}} - mark'_{pkt}$
3  if ( $ins_{pkt} <= n$ ) then
4      $sCounter++$
5      if ($ins_{pkt} < n$) then
6          $Counter[ins_{pkt}]++$
7  else if ($ins_{pkt} == n+1$) then
8      process the packet as $ACK$ or $reply$
9  else
10     filter the packet
11 if the end of an epoch then
12     forward ACK toward the source
```

The destination processes the packet with Algorithm 4. It first initializes the counters ($Counter[I]$ for each hop and $sCounter$ for this session) with 0 (line 1).

When the destination receives a packet, it calculates $mark'_{pkt}$ with $mark'_{pkt} = MAC_{HDK_{sd}}(CST_{PH}||PL)[0:32]$, then calculates the ins_{pkt} (line 2). The ins_{pkt} indicates that this packet is legitimate ($ins_{pkt} <= n$) or sampled ($ins_{pkt} < n$), the destination performs the $sCounter{++}$ and $Counter[ins]++$ respectively (line 3-6); $ins_{pkt} == n+1$ represents that its an ACK or a $reply$, and this "destination" is a "source" of a specific session (line 7-8). If the $\boxed{mark_{pkt}}$ header is wrong, it filters the packet (line 9-10).

At each end of an epoch, the destination forwards the counter information to the source with an ACK (line 11-12). An ACK has the same packet header as the $reply$ except that the destination sends all the $Counter[i]$ of an epoch to facilitate the source to get a comprehensive corruption ratio. As detailed in the processing of the $reply$, it utilized the $mark_r$ to load the $sCounter$ information.

The destination calculates the end-to-end corruption ratio based on the $sCounter$ information:

$$repu = sCounter/N \tag{10}$$

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, and the theoretical corruption ratio incurred by the natural packet loss is $\theta_n = 1-(1-\rho_n)^k$. As the $sCounter$ records all the packets received by the destination, we denote the end-to-end corruption ratio threshold as:

$$\theta_{threshold} = \rho + \theta_n \tag{11}$$

If the end-to-end corruption ratio is less than a threshold (e.g., $\rho + \theta_n = 5\% + 0.5\% = 5.5\%$ for $\rho = 5\%$, $\rho_n = 0.1\%$, and $k = 5$), it only sends the $sCounter$ to the source. If the end-to-end corruption ratio is more than the threshold, it nests the $Counter[i]$ in the packet payload as:

$$ACK_{info} = ENC_{HDK_{sd}}(Counter[i]) \tag{12}$$

The source gets the end-to-end corruption ratio $corr_e$ with the $sCounter$. If the corruption ratio is no more than the threshold $\theta_{threshold}$ (e.g., 5.5%), it needs not to localize the faulty link. Or else, the source calculates the $repu'_i$ of each on-path router as:

$$repu'_i = Counter[i]/Counter_{src}[i] \tag{13}$$

In which $Counter_{src}[i]$ is the counter in the source. The $repu'_i$ is not an indicator of the actual reputation of each hop since the downstream routers might corrupt the packets. For example, R_2 in Fig. 1 forwards 100% of the packets, R_4 drops 5% of the packets, the $repu'_i$ is less than 100% for R_2 since the destination could not record the packets dropped by R_4. In Sec. 5.1, we will detail that $repu'_i$ is an indicator to defend against the coward attack.

4.6 Parameters in D3

D3 utilizes the *blind policy* to achieve fault localization. The probabilistic sampling policy could facilitate each hop to offload the storage overhead from the data plane to the control plane. The parameters, which include the number of packets in each epoch (N) and the sampling ratio (sp_i), could influence the localization accuracy. In general, the larger N and the higher sp_i, the higher localization accuracy. But the larger N indicates more localization delay, while the higher sp_i indicates the higher bandwidth between the data plane and control plane. From this perspective, a reasonable parameter of N and sp_i is important for D3.

Consider that the corrupted packets are randomly distributed in each epoch. We perform uniform sampling according to different sp_i (e.g., the source instructs each hop to randomly sample 15% of the packets if $sp_i = 15\%$). To set reasonable sampling parameters, we evaluate the sampling consistency under different corruption ratios (ρ) with a Python program. The source samples packets according to a certain sp_i, and each hop corrupts packets according to a certain corruption ratio. For each hop, the total packets set is T, the sampling packets set is S, the actual corrupted packets set is D, and the sampling corrupted packets set is the intersection of S and D ($S \cap D$). For each hop, the sampling corruption ratio is $SCR = (S \cap D)/S$, and the actual corruption ratio is $ACR = D/T$. We define sampling

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on January 14,2025 at 07:11:02 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2024.3392486

TABLE 2
The sampling consistency in different parameters

N	$\rho = 10\%$	5%	3%	$\rho = 10\%$	5%	3%
		sp_i= 5%			sp_i= 10%	
1000	491	321	277	652	500	413
2000	648	485	390	821	651	537
3000	754	585	476	896	737	630
4000	816	654	556	944	819	714
8000	944	824	700	993	930	862
		sp_i= 15%			sp_i= 20%	
1000	753	551	442	800	636	543
2000	886	724	594	932	827	696
3000	957	825	711	978	886	797
4000	978	898	791	993	941	864
8000	999	980	924	1000	993	966

consistency as the difference between the two corruption ratios is less than 30% ($\frac{|SCR-ACR|}{ACR} < 30\%$). In this case, the sampling corruption ratio could represent the actual corruption ratio. For example, if the actual corruption ratio is 5%, the sampling corruption ratio of $3.5\% - 6.5\%$ meets the requirement of sampling consistency.

We evaluate the sampling consistency under different parameters (N, sp_i, and ρ). We test each case with 1000 epochs and record the number of epochs it meets the requirement of sampling consistency. As shown in TABLE 2, when $N = 1000$ in each epoch, only about 63.6% of epoch meet the consistency requirement even if $sp_i = 20\%$ and $\rho = 5\%$. When $N = 3000$, with $sp_i = 15\%$ and $\rho = 5\%$, the ratio of sampling consistency can reach 82.5%. When $N = 4000$, the ratio of sampling consistency can reach 89.8% with $sp_i = 15\%$ and $\rho = 5\%$. Therefore, we utilize $N = 4000$ and $sp_i = 15\%$ as default. With the larger N, the sp_i can be reduced linearly. For example, the parameters of $N = 8000$ and $sp_i = 5\%$ could achieve almost the same sampling consistency as $N = 4000$ and $sp_i = 10\%$.

It is worth mentioning that the source localizes the faulty link according to the gap of adjacent hops, and the localization accuracy is affected by the sampling consistency of adjacent hops. Sec. 6 will evaluate the localization accuracy with different N, sp_i, and ρ in each epoch.

5 ANALYSIS

We analyze the security and performance properties in this section. As *D3 in edge cloud* is the same as D3 except that D3 needs much less storage overhead in terms of dynamic key, we compare D3 with Faultprints and RFL. The analysis shows that D3 achieves strong security properties with relatively low overhead.

5.1 Security analysis

As shown in TABLE 3, we analyze 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 attacks, framing attacks, colluding attacks, and replay attacks. Finally, we explain why D3 does not create the opportunity for a new DoS attack.

Packet corruption and path inconsistency. If the adversary modifies the packet header, and the adversary delays or drops the packets, the $Counters$ in the downstream routers will decrease. If the adversary launches the path inconsistency attack, the $Counter$ in the downstream router will decrease since the $mark$ is calculated with the link information. The source localizes the link with the reputation gap more than the threshold as a faulty link. Faultprints and RFL have the same security property.

Coward attack. Faultprints and RFL could defend against it since an on-path router does not know whether other on-path routers should sample the packet. For the D3-supported destination, D3 could defend against coward attack with the $Counter$ information from the destination. As the adversary only drops the packets sampled by itself, the source could localize it with the $Counter[i]$ from the destination. For example, R_2 in Fig. 1 drops 5% of the packets sampled by R_2, R_3 and R_4 reply the correct $Counter$ to the source since they do not need to sample the dropped packets, then $repu_2 = repu_3 = repu_4 = 100\%$. The $Counter[i]$ from the destination will indicate the packet dropping, then $repu'_2 = 95\%, repu'_3 = repu'_4 = 100\%$. The source could eventually localize R_2.

For the legacy destination, the source could not get $Counter[i]$ from the destination. It could utilize $mark_r$ as another sampling instruction to instruct the sampling policy. That means, each on-path router needs to compare with $mark_r$ and $mark_{src}$, then decide whether sample a packet. Each on-path router could not launch the coward attack since it does not know whether the downstream on-path router will sample this packet. But it incurs the computation overhead of the agent, and two colluders still could launch the coward attacks.

Based on the source-driven localization, an alternative is the source clears the noise from other on-paths routers by an $exclusive\ sampling\ policy$. That means, only one on-path router to sample the packets in a specific epoch. The source gets the end-to-end performance from the upper layer as a reference (e.g., the performance of TCP). When the end-to-end performance is significantly lower than the $repu_i$ of the last on-path router, it instructs one on-path router to exclusively sample packets in an epoch. Assuming an epoch has N packets, the source instructs the first on-path router sample $N \times sp_1$ packets in epoch m, the second on-path router sample $N \times sp_2$ packets in epoch $m + 1$ (or another epoch). The rest on-path routers can sample $N \times sp_i$ packets in the same way. As each on-path router only drops the packets it sampled, the source could localize the fault entity with the end-to-end performance and the $repu_i$ in a specific epoch. For example, if R_2 launches the coward attack, it only drops the packets at epoch $m + 1$, the end-to-end performance in epoch $m + 1$ would be significantly lower than other epochs since the coward attacker (R_2) does not drop packets in other epochs.

Framing attack. To frame others with path inconsistency, R_1 in Fig. 1 changes the path information from R_3 to R_j, which is not the source anticipated. But as the path information is the partial input of the $mark_r$, R_2 could not calculate the correct $mark_i$. The $Counter$ information from R_2 will turn

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on January 14,2025 at 07:11:02 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2024.3392486

TABLE 3
Comparison of security with existing mechanisms

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

out to be wrong. The source would localize the link between R_1 and R_2 as a fault. Faultprints and RFL could also defend against this attack with the chained $marks$.

To frame others with packet corruption, assume that R_1 in Fig. 1 wants to modify the $mark_{src}$ of R_4. But with the $blind\ policy$, R_1 could not know the $mark_i$ of R_4 since it does not have the dynamic key between the agent and R_4, the $Counter$ from R_2 will turn to be lower, the source then gives R_1 and R_2 a low reputation. In Faultprints, the sampling policy is based on the chained $marks$ ($Auth_{modif,i}$). 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. While in RFL, to prevent an adversary from changing the downstream router's $mark$, each on-path router verifies the packet based on the $marks$ (M_i) of all the downstream routers. If R_1 changes the $mark$ for R_4 (or R_3), R_2 will drop the packet after the failure of verification. It could defend against the framing attack if each entity strictly enforces the verification. But in RFL, the sampling policy is determined by $PacketID$, not by the M_i. R_2 might not drop the packet considering the AS relationship on the Internet. The adversary could achieve the attack if R_3 or R_4 drops the packet.

Colluding attack. There are two situations of colluding attacks in violating the localization of D3: (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 $Counter$ from R_{i+1}.

For the case of adjacent colluders, e.g., R_{i+1} and R_i share the dynamic key. If R_i corrupted a packet should be sampled by R_{i+1}, and R_{i+1} recorded 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. Faultprints and RFL have the same characteristic as D3.

Replay attack. Assuming that an adversary launches a replay attack. Each router in D3 could mitigate this attack by recording and replying with the $Counter$. It could defend against the replay attack in coarse granularity. The D3-supported destination could observe the misbehavior with seq in fine granularity. It could filter this packet and announce it in the ACK with another $Counter$. Compared with Faultprints which utilizes 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 replay attacks.

New DoS attack. The adversary could aggressively send packets to exhaust the computation and storage resources of each router or destination. It would not hurt D3 since we minimize the computation overhead and have constant storage overhead in the data plane. The adversary could also aggressively send DAT_{SAM} to overwhelm the channel between the data plane and the control plane. But the router only clones the packet header to the control plane after the verification of $mark_{src}$. The adversary had little chance to fabricate the correct $mark$ since it did not know the dynamic key. The D3-supported destination only stores the $Counter$ information for a small number of sources, and it could directly filter the fabricate packet with the $mark_{pkt}$, the adversary has little chance to overwhelm its storage space.

The source in $D3\ in\ edge\ cloud$ needs to achieve localization with the $probe$ and $reply$ packets. If an adversary launch DoS attacks by frequently probing and receiving replies from on-path routers, it would incur a potential increase in network communication overhead. Besides that, as the $reply$ packet was processed in the control plane, it would incur extremely high computation overhead in each on-path router, and generate too much network traffic between its control plane and data plane. In extreme cases, it might overwhelm an on-path router. Therefore, D3 cancels the $probe$ packet from the source. Each on-path router in D3 only sends one $reply$ packet toward the source at each end of an epoch. From this perspective, the adversary could not overwhelm the on-path router in D3. Therefore, the control ($reply$) packets have negligible impact on network traffic.

5.2 Performance analysis

As shown in TABLE 4, compared with the state-of-the-art mechanisms, D3 has a significantly lower overhead in terms of storage, computation, and communication overhead.

Storage overhead. In Faultprints and RFL, each router derives the dynamic key on the fly to keep 1 key ($16B$) storage overhead. They record the sampling packet with the Bloom Filter, which is a relatively sustainable way (468 MB for a bandwidth of 100 $Gbps$) to store the $proof$ 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.[8] In the extreme case, a router needs to preset the dynamic keys for all the other ASes. With the key length of 24 B and each router presets the key for 2 time slots, each AS has N agents, the storage overhead is $80 \times 10^3 \times 2 \times 24 \times N\ B$, which is around $3.84 \times N\ MB$.

8. https://www.cidr-report.org/as2.0/

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on January 14,2025 at 07:11:02 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2024.3392486

TABLE 4
Comparison of performance with existing mechanisms

	Data plane storage overhead (B)[1]	Additional packet header (B)[2]	Control packet	No ASY in data plane	No deriving key on the fly	MAC operation	localization delay (pkts)
Faultprints	$16 + 468 \times 10^6 \times BW$	$56 + 8 \times n$	2400	✗	✗	2	4000
RFL	$16 + 468 \times 10^6 \times BW$	$38 + 4 \times n$	2	✗	✗	2	4000
D3	$3.84 \times 10^6 \times L$	20	5	✓	✓	1	4000

[1] BW represents the bandwidth in metric of 100 Gbps, L represents the number of agents in an ASes.
[2] n represents the path length.

D3 offloads the $proof$ 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) with a hash table (the data index calculated by the hash of the session information). Two epochs are enough since each router clears the $Counter$ of epoch $i-1$ at the end of epoch i, which takes 4 B for a session. Consider the load factor of 0.75, and each router's average number of flows per second is 39.73 K.[9] The total storage overhead of D3 in the control plane is $N = 39.73 \times 4/0.75\ KB$, which is around 212 KB.

Communication overhead. The additional packet header is the most important communication overhead. Faultprints and RFL have 96 B and 58 B additional packet headers with a path length of 5 respectively and will grow with the increase of path length ($(56 + 8 \times n)$ and $(38 + 4 \times n)$ with the path length of n). D3 has the constant additional packet header of 20 B to embed the probabilistic and cryptographic sampling policy, which is significantly lower than Faultprints and RFL. Besides that, the number of $probe$ and $reply$ packets also has an impact on network traffic. Considering forward and reverse paths of 5 ASes, in Faultprints, the ACK from the destination can be piggybacked on TCP acknowledgments, therefore, there is no control packet if the packet forwarding is the same as the source anticipated. In contrast, if the corruption ratio is larger than the threshold, the source sends 1 $probe$ packet and receives 5 $reply$ packets for each data packet. Assuming that the P_{probe} is 10% for 4000 packets, the number of control packets is $6 * 400 = 2400$. RFL has 2 control packets (1 $ReqProb$ packet and 1 $AckProb$ packet) since each on-path router only replies one packet with the $proof$ (encrypted Bloom Filters) and the signature calculated with its private key, which incurs heavy computation overhead in asymmetric cryptography. D3 does not need ACK and $probe$ packets, it utilizes a source-driven localization. Then each on-path router only sends 1 $reply$ packet toward the source at the end of each epoch. The number of control packets is only 5.

Computation overhead. Faultprints and RFL need to negotiate the dynamic keys with asymmetric encryption, which is prohibitively expensive in the data plane. Moreover, they need to derive the dynamic key on the fly for each packet, which takes 1 MAC operation with a strong security guarantee (e.g., AES). Besides that, Faultprints accomplishes the processing of $Auth_{modif,i}$ with 1 MAC operation, verifies Con_i with 1 MAC operation if the $Auth_{modif,i}$ indicates that it should store the $proof$ in the Bloom Filter. It utilizes 1 MAC operation to achieve the delay localization. Therefore, it needs at least 2 MAC operations. RFL achieves the delay localization in each $epoch$, it needs to calculate per-packet M_i with 1 MAC operation, and calculate the sampling policy with 1 MAC operation, thus having 2 MAC operations. D3 presets the dynamic key in the control plane, each router only performs 1 lightweight 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 on the Internet peaks at 100 ms and 200 ms.[10] For a packet size of 1500 B, 4000 packets in an epoch make up 48 Mb. 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 $reply$, and the evaluation in Sec. 6 testifies that the localization accuracy could achieve 95% with 4000 packets in an epoch. Therefore, D3 utilizes 4000 packets in an epoch as default. In Faultprints [27], the evaluation results show that the probabilistic model can achieve a high localization accuracy, i.e., the localization accuracy is more than 0.95 with a P_{Probe} of 0.15 and 4000 data packets. RFL utilizes the same storage mechanism except that each on-path router only replies one packet with the $proof$ (encrypted bloom filters) and the signature, thus also having around 4000 packets in each epoch.

6 EVALUATION

6.1 Implementation

We implement D3 in the BMv2 environment[11] and Barefoot Tofino hardware. Our BMv2 testbed hosts in a virtual machine with Ubuntu 16.04, Intel Core i5-8250U CPU, 2.4 GHz, and 4 GB RAM. We instantiate the MAC operation with 2EM [37]. As the 2EM has been proven secure up to $2^{2n/3}$, we utilize the 24 B dynamic key to achieve $n = 64$. The implementation includes the terminals (source and destination), the agent, and the on-path routers. The processing of agent and on-path routers includes nearly 900 lines $P4_{16}$ program in the data plane. We configure a router as an agent or an on-path router from the control plane with $P4\ Runtime$. The agent verifies the packet and updates the $marks$ with the source's policy. In contrast, the on-path router performs the data plane operations to sample and forward the packets. We implement nearly 500 lines of Python program in the on-path router's control plane, including preparing the $reply$. We instantiate two terminals as the

9. https://www.caida.org/catalog/datasets/trace_stats/

10. https://www.caida.org/catalog/software/walrus/rtt/
11. https://github.com/p4lang/behavioral-model

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on January 14,2025 at 07:11:02 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2024.3392486

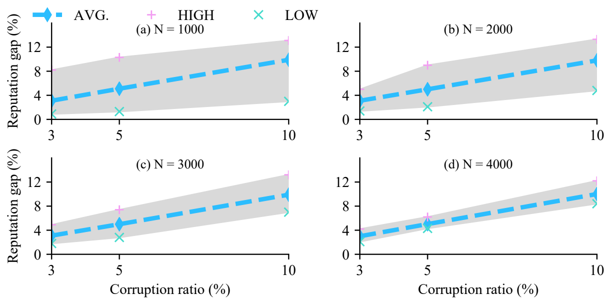

Fig. 4. The range of reputation gap with different parameters.

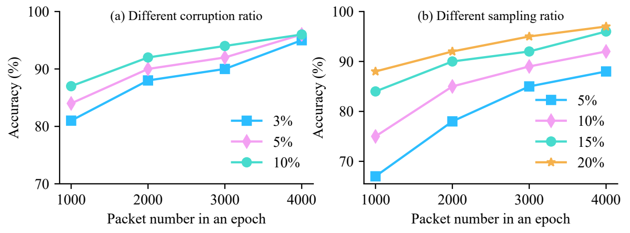

Fig. 5. The localization accuracy with different parameters.

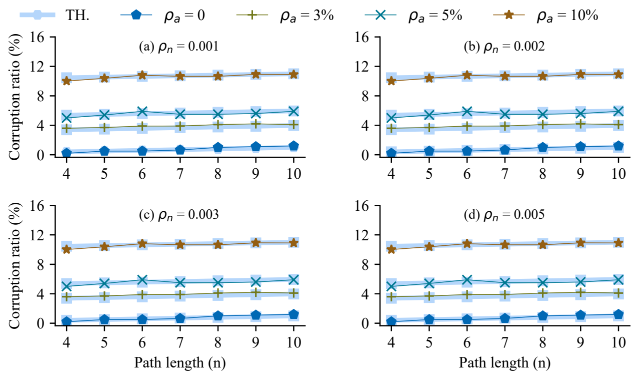

Fig. 6. The end-to-end corruption ratio with different parameters.

source and the destination host, implement the processing of different packets, and the fault localization with nearly 800 lines of Python program. In commodity Barefoot Tofino programmable switch S9180-32X, we implement the key switching in the control plane, and the data plane operation of D3, which includes nearly 900 lines $P4_{16}$ program with 2EM operation as MAC operation. Finally, we test the packet processing performance at the host.

6.2 Evaluation in BMv2

Reputation gap. We evaluate the reputation gap of the faulty link with $sp_i = 15\%$, the path length is 4, and $\rho_n = 0.001$. The source sends $N = 1000, 2000, 3000, 4000$ packets in each epoch. An adversary's corruption ratio is $\rho = 3\%,\ 5\%,\ 10\%$, respectively. As shown in Fig. 4 (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. 4 (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. 5 (a) shows the localization accuracy under different corruption ratios with the sp_i of 15%. It shows that with the corruption ratio of 10%, the accuracy is around 90%, and 95% with 2000, and 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 4000. This testifies that D3 can achieve fault localization even with the adversary's corruption ratio of 3%. Fig. 5 (b) is the localization accuracy under different sp_i with a corruption ratio of 5%. It shows that with 1000 packets in an epoch and a sp_i of 5%, the accuracy is only around 65%. It is around 85% with a packet number of 2000 and a sp_i of 10%. And it is more than 95% with 4000 packets and a sp_i of 15%. The baseline of the state-of-the-art mechanisms (Faultprints) is to achieve a localization accuracy of 95% for 4000 packets in an epoch under the weaker attackers ($\rho = 5\%$), 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 ($reply$) with the same ratio as DAT. Each router could forward more than 1 $reply$ to facilitate the source receiving at least 1 $reply$ from each on-path router. We evaluate the number of $replies$ in different corruption ratios. The results show that under the corruption ratio of 3%, the source receives more than 98% of the $replies$ if each router forward one $reply$; under the corruption ratio of 5%, two $reply$ from each router could satisfy that the source receives 95% of the $replies$; under the corruption ratio of 10%, two $replies$ could meet that the source receives 90% of the $replies$. Considering each epoch has 4000 packets, and the control packet size is less than $100\ B$, assuming that the average size of DAT is $1000\ B$, each router forwards 2 $replies$ at each end of an epoch, the extra communication overhead incurred by D3's control packets is around 0.025% for the path length of 5. In Faultprints, the corresponding overhead is 1.81%.

End-to-end corruption ratio from the D3-supported destination. 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 $DATs$ in an epoch. An adversary corrupts packets with the corruption ratio of $\rho = 0, 3\%, 5\%, 10\%$. Fig. 6 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 all the packets. It testifies that the D3-supported destination makes the localization more robust. This is especially useful for an enterprise sending many of its flows along the same path, e.g., between two remote offices. A transit AS that tries to violate the communication is detected quickly. It is worth

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on January 14,2025 at 07:11:02 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2024.3392486

mentioning that in the case of the communication from the client to the server, the server could only keep stateful for parts of the clients, to avoid the DoS attack towards the server.

6.3 Evaluation in hardware

As the dynamic key is critical for the localization, we first test the delay of dynamic key switching in the control plane in switch S9180-32X. Then we implement 2EM [37] in S9180-32X as MAC operation, and testify that the hardware could accomplish the processing of D3 within one pipeline. Besides that, as D3 sends parts of the packet headers from the data plane to the control plane to store the $proof$, it incurs a certain latency cost. We first test the latency from the data plane to the control plane, which takes 3 ms for one packet. It takes around 1 μs to store the $Counter$ in the control plane. As each epoch is about a few hundred milliseconds, this latency is acceptable for epoch-based fault localization. In practice, as we only transmit the relevant packet header to the control plane, we could utilize 1 port of the switch to connect with a more powerful server, or even with a stateful per-flow packet processor system [19].

Key switching. In the beginning, each router needs to preset the dynamic keys of the first two time slots and ensure that there are dynamic keys of ts_i and ts_{i+1} at each time slot. After the end of ts_i, the hardware updates the key of ts_i with the key of ts_{i+2}. Therefore, we test the delay of key switching (presetting and clearing) in the control plane. As shown in TABLE 5, when the number of keys is 10^4, the key switching can be completed within 1 second, that is, the router clears the key of ts_i, and presets the key of ts_{i+2} in the control plane, while takes about 10 seconds for 10^5 keys. When the granularity of the time slot is minute, each router could keep synchronization at the minute level and preset the dynamic keys of two time slots to ensure the key sharing between the agent and the router, which is sustainable in the hardware.

TABLE 5
Key distribution and clearing delay

Presetting(ms)				Clearing(ms)			
1	10^2	10^4	10^5	1	10^2	10^4	10^5
0.28	12.09	808.35	10454.78	0.15	0.81	39.375	327.37

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 symmetric keys with the first MAC operation and then accomplish 2 MAC operations (as detailed in Sec. 5). As the hardware accomplishes the MAC operation with strong security guarantee (e.g., AES) takes relatively higher computation resources and significantly decreasing the throughput (e.g., the throughput for an AES-128 operation on RMT programmable switch is 10.92 Gbps) [31]. We could cache the symmetric key in the hardware

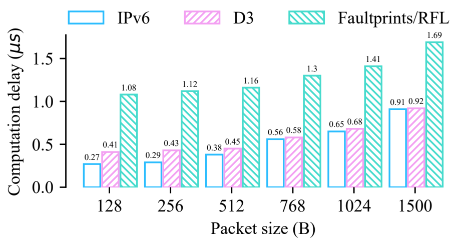

Fig. 7. The computation delay in Barefoot Tofino hardware.

[38] (though it incurs high storage overhead), and achieve 2 MAC operations with one recirculation.

The computation delay in different packet sizes is shown in Fig. 7, in which IPv6 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, Faultprints and RFL need 2 MAC operations. We implement 2 MAC operations with one recirculation. We test the computation delay of 2 MAC operations, which is 1.08 µs with a packet size of 128 B, and 1.69 µs with a packet size of 1500 B.

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

6.4 Performance at agent

In D3, the agent first verifies the source with the first MAC operation and updates the $marks$ with the second MAC operation. The first MAC operation determines whether the packet is legitimate, and the second MAC operation only updates the $marks$. Considering that the agent is stateless, in S9180-32X, we could achieve the two MAC operations in the ingress pipeline and egress pipeline, thus achieving line rate processing.

6.5 Performance at host

In D3, the host requests path information and dynamic keys from the KDS. The additional latency is insignificant since the paths and keys information is available at local KDS. Moreover, a host 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 the most computation overhead. We test the MAC operations with 2EM in the Intel Core i5-8250U CPU virtual machine. It could achieve more than 1,200,000 2EM operations in one second. The source needs a 2EM operation to calculate the $mark_{pkt}$, and a 2EM operation to calculate the $mark_{src_a}$ for the

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on January 14,2025 at 07:11:02 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2024.3392486

agent. In comparison, the D3-supported destination needs to calculate the $mark_{pkt}$. We could learn that an ordinary commodity host could support up to 600,000 packets for the source and 1,200,000 for the destination, which is 7.2 Gbps and 14.4 Gbps for the packet size of 1500 B. We can conclude that the computation overhead at the host is sustainable.

We test the localization delay in the virtual machine, which is the delay of each source to figure out the faulty link with replies from on-path routers. With $1,000$ runs, the average delay for the source to calculate the $repu_i$ for each on-path router is 1.10 ms. For a path length of 5, the localization delay is around 5.50 ms. As each epoch takes more than 100 ms, it is sustainable for each source to localize the faulty link.

7 DISCUSSION

7.1 Incremental deployment

As the deployment of PAN is an incremental process [39], the deployment of D3 in PAN is also an incremental process. Fortunately, D3 could benefit the early adopter. Firstly, the source could get benefits from D3. For example, R_2 in Fig. 1 is a legacy router, the source could nest the path information with $R_1 - R_3 - R_4$ in the packet header with a specific routing header, and achieve the localization with the $replies$ from R_1, R_3, R_4. This mechanism could achieve by a specific PAN which supports the loose source routing (e.g., SRv6). Secondly, the ISP could benefit from D3 since it could attract traffic from an upgraded source which need more security guarantees. In general, for the network which has an administrator (e.g., the enterprise network), D3 could support the source host or a gateway to achieve fault localization as the administrator's expectation. While for some emerging new scenarios (e.g., the source is an automated vehicle or a smart wearable for cardiac monitoring), with a hierarchical design, an agent in D3 could establish the fault localization after a packet verification, thus empowering the source to improve its security in the network data plane.

7.2 D3-supported destination could benefit fault localization

The D3-supported destination could improve the robustness of the localization. We described in Sec. 4 that the adversary might launch a coward attack if the legacy destination does not return the ACK. A countermeasure is that we could utilize $exclusive\ sampling\ policy$ to defend against it (The source needs to get the end-to-end performance from the upper layer). A D3-supported destination host could benefit D3 to defend against the coward attack. It is worth mentioning that the destination could only keep stateful for the authenticated source (e.g., the hosts in the same enterprise), to prevent the adversary from launching a DoS attack toward the stateful destination.

8 RELATED WORK

Data plane MAC operation. There are several mechanisms available to achieve an intelligent network data plane using programmable hardware [40], [41]; however, accomplishing MAC operation in the data plane remains challenging. SPINE [42] achieves the MAC operation with SipHash in the BMv2 environment, but the hardware could not accomplish it within one single pipeline [43]. P4-AES [31] requires at least four recirculations to accomplish a MAC operation. PINOT [37] 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 fault localization with the two $marks$.

Path availability and verification. The PAN, exemplified by SCION [17], offers transparency and user choices, while SR [18] provides a practical network programming approach. These mechanisms form the foundation for constructing a PAN that enables hosts to embed their policies in packet headers. For path verification, EPIC [20] employs efficient symmetric cryptographic operations during forwarding, whereas PPV [44] and MASK [45], [46] enhance path verification efficiency through probabilistic packet marking. These mechanisms can only verify the forwarding path, and cannot localize the faulty link.

Fault localization. AudIt [23] and Network Confessional [47] lack packet authentication. DynaFL [48] and DYNAPFV [49] focus on detecting attacks against packet forwarding in SDN. TrueNet [50] utilizes trusted computing technology to establish a trusted network-layer architecture. ShortMAC [26] provides strong security guarantees but comes with significant storage overhead. These mechanisms encounter challenges in inter-domain applications. Faultprints [27] is the first fault localization mechanism designed for inter-domain context, employing per-packet $proof$ recording with Bloom Filter. RFL [28] addresses fault localization in the context of unreliable communication channels. Unfortunately, both mechanisms still involve significant overhead. In contrast, D3 achieves high localization accuracy with minimal overhead in on-path routers through a source-driven mechanism.

9 CONCLUSION

It is difficult to deploy the existing secure fault localization mechanisms in the PAN for the heavy storage, communication, and computation overhead. This paper designs D3, which offloads the router's overhead to the host and control plane. D3 has low storage overhead in the data plane and communication overhead in the network. With the 2EM operation, each on-path router could accomplish all the MAC operations within a single pipeline on commodity hardware, thus significantly lowering computation overhead. And it has strong security guarantee characteristics with the $blind\ policy$. Furthermore, D3 keeps backward compatibility with the source-driven design. With D3, the source could program the router on the path and build a more reliable network to meet its requirements. It thus brings us closer to localizing the malicious ASes in the PAN.

REFERENCES

[1] Y. Zhao, K. Xu, H. Wang, B. Li, M. Qiao, and H. Shi, "Mec-enabled hierarchical emotion recognition and perturbation-aware defense in smart cities," *IEEE Internet Things J.*, vol. 8, no. 23, pp. 16 933–16 945, 2021.

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on January 14,2025 at 07:11:02 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2024.3392486

[2] M. Shen, J. Zhang, L. Zhu, K. Xu, and X. Tang, "Secure SVM training over vertically-partitioned datasets using consortium blockchain for vehicular social networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 6, pp. 5773–5783, 2020.

[3] X. Feng, C. Fu, Q. Li, K. Sun, and K. Xu, "Off-path TCP exploits of the mixed IPID assignment," in *ACM SIGSAC Conference on Computer and Communications Security*, 2020, pp. 1323–1335.

[4] J. Zheng, Q. Li, G. Gu, J. Cao, D. K. Yau, and J. Wu, "Realtime ddos defense using cots sdn switches via adaptive correlation analysis," *IEEE Trans. Inf. Forensics Secur.*, vol. 13, no. 7, pp. 1838–1853, 2018.

[5] Q. Li, X. Deng, Z. Liu, Y. Yang, X. Zou, M. Xu, and J. Wu, "Dynamic network function enforcement via joint flow and function scheduling," *IEEE Trans. Inf. Forensics Secur.*, vol. 17, pp. 486–499, 2022.

[6] 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.*, vol. 31, no. 3, pp. 1416–1431, 2023.

[7] R. Xie, J. Cao, Q. Li, K. Sun, G. Gu, M. Xu, and Y. Yang, "Disrupting the sdn control channel via shared links: Attacks and countermeasures," *IEEE/ACM Trans. Netw.*, vol. 30, no. 5, pp. 2158–2172, 2022.

[8] 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, San Diego, California, USA, February 23-26*, 2020, pp. 1–18.

[9] G. Li, M. Zhang, S. Wang, C. Liu, M. Xu, A. Chen, H. Hu, G. Gu, Q. Li, and J. Wu, "Enabling performant, flexible and cost-efficient ddos defense with programmable switches," *IEEE/ACM Trans. Netw.*, vol. 29, no. 4, pp. 1509–1526, 2021.

[10] C. Fu, Q. Li, M. Shen, and K. Xu, "Realtime robust malicious traffic detection via frequency domain analysis," in *2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19*, 2021, pp. 3431–3446.

[11] X. Feng, Q. Li, K. Sun, K. Xu, B. Liu, X. Zheng, Q. Yang, H. Duan, and Z. Qian, "Pmtud is not panacea: Revisiting ip fragmentation attacks against tcp," in *29th Annual Network and Distributed System Security Symposium, San Diego, California, USA, April 24-28*, 2022.

[12] 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, 2022.

[13] X. Feng, Q. Li, K. Sun, Z. Qian, G. Zhao, X. Kuang, C. Fu, and K. Xu, "Off-path network traffic manipulation via revitalized icmp redirect attacks," in *31st USENIX Security Symposium, Boston, MA, USA, August 10-12*, 2022, pp. 2619–2636.

[14] C. Fu, Q. Li, M. Shen, and K. Xu, "Frequency domain feature based robust malicious traffic detection," *IEEE/ACM Trans. Netw.*, vol. 31, no. 1, pp. 452–467, 2023.

[15] C. Fu, Q. Li, and K. Xu, "Detecting unknown encrypted malicious traffic in real time via flow interaction graph analysis," in *30th Annual Network and Distributed System Security Symposium, San Diego, California, USA, February 27 - March 3*, 2023, pp. 3431–3446.

[16] X. Liu, A. Li, X. Yang, and D. Wetherall, "Passport: Secure and adoptable source authentication," in *5th USENIX Symposium on Networked Systems Design & Implementation, San Francisco, CA, USA, April 16-18, 2008*, 2008, pp. 365–376.

[17] A. Perrig, P. Szalachowski, R. M. Reischuk, and L. Chuat, *SCION: A Secure Internet Architecture*, ser. Information Security and Cryptography. Springer, 2017.

[18] C. Filsfils, P. Camarillo, J. Leddy, D. Voyer, S. Matsushima, and Z. Li, "Segment routing over ipv6 (srv6) network programming," *RFC*, vol. 8986, pp. 1–40, 2021.

[19] M. Scazzariello, T. Caiazzi, H. Ghasemirahni, T. Barbette, D. Kostic, and M. Chiesa, "A high-speed stateful packet processing approach for tbps programmable switches," in *20th USENIX Symposium on Networked Systems Design and Implementation, Boston, MA, April 17-19, 2023*, pp. 1237–1255.

[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, August 12-14*, 2020, pp. 541–558.

[21] H. Birge-Lee, L. Wang, J. Rexford, and P. Mittal, "SICO: surgical interception attacks by manipulating BGP communities," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, London, UK, November 11-15*, 2019, pp. 431–448.

[22] V. N. Padmanabhan and D. R. Simon, "Secure traceroute to detect faulty or malicious routing," *Comput. Commun. Rev.*, vol. 33, no. 1, pp. 77–82, 2003.

[23] 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, Beijing, China, October 16-19*, 2007, pp. 194–205.

[24] A. T. Mizrak, Y. Cheng, K. Marzullo, and S. Savage, "Fatih: Detecting and isolating malicious routers," in *2005 International Conference on Dependable Systems and Networks Yokohama, Japan, 28 June - 1 July*, 2005, pp. 538–547.

[25] 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, pp. 341–360.

[26] 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, San Diego, California, USA, February 5-8*, 2012, pp. 1–19.

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

[28] 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.

[29] S. Fu, Q. Li, X. Wang, S. Yao, X. Feng, Z. Wang, X. Du, K. Wan, and K. Xu, "D3: lightweight secure fault localization in edge cloud," in *42nd IEEE International Conference on Distributed Computing Systems, Bologna, Italy, July 10-13*, 2022, pp. 515–525.

[30] S. Yao, M. Wang, Q. Qu, Z. Zhang, Y.-F. Zhang, K. Xu, and M. Xu, "Blockchain-empowered collaborative task offloading for cloud-edge-device computing," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 12, pp. 3485–3500, 2022.

[31] X. Chen, "Implementing AES encryption on programmable switches via scrambled lookup tables," in *Proceedings of the 2020 ACM SIGCOMM Workshop on Secure Programmable Network Infrastructure, Virtual Event, USA, August 14*, 2020, pp. 8–14.

[32] 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.

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

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

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

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

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

[38] J. de Ruiter and C. Schutijser, "Next-generation internet at terabit speed: SCION in P4," in *The 17th International Conference on emerging Networking EXperiments and Technologies, Virtual Event, Munich, Germany*, 2021, pp. 119–125.

[39] Y. Tian, Z. Wang, X. Yin, X. Shi, Y. Guo, H. Geng, and J. Yang, "Traffic engineering in partially deployed segment routing over ipv6 network with deep reinforcement learning," *IEEE/ACM Trans. Netw.*, vol. 28, no. 4, pp. 1573–1586, 2020.

[40] G. Zhou, Z. Liu, C. Fu, Q. Li, and K. Xu, "An efficient design of intelligent network data plane," in *32nd USENIX Security Symposium, Anaheim, CA, USA, August 9-11*, 2023, pp. 6203–6220.

[41] J. Yan, H. Xu, Z. Liu, Q. Li, M. X. Ke Xu, and J. Wu, "Brain-on-switch: Towards advanced intelligent network data plane via nn-driven traffic analysis at line-speed," in *NSDI*, 2024, pp. 1–16.

[42] 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, Santa Clara, CA, USA, August 13*, 2019, pp. 1–7.

[43] S. Yoo and X. Chen, "Secure keyed hashing on programmable switches," in *Proceedings of the ACM SIGCOMM 2021 Workshop on Secure Programmable network INfrastructure, Virtual Event, USA, 27 August 2021*, pp. 16–22.

[44] 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, Banff, AB, Canada, June 4-6, 2018*, 2018, pp. 1–10.

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on January 14,2025 at 07:11:02 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2024.3392486

[45] 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 *29th IEEE/ACM International Symposium on Quality of Service, Tokyo, Japan, June 25-28*, 2021, pp. 1–10.
[46] S. Fu, Q. Li, M. Zhu, X. Wang, S. Yao, Y. Guo, X. Du, and K. Xu, "MASK: practical source and path verification based on multi-as-key," *IEEE/ACM Trans. Netw.*, vol. 31, no. 4, pp. 1478–1493, 2023.
[47] K. J. Argyraki, P. Maniatis, and A. Singla, "Verifiable network-performance measurements," in *Proceedings of the 2010 ACM Conference on Emerging Networking Experiments and Technology, Philadelphia, PA, USA, November 30 - December 03*, 2010, pp. 1–12.
[48] X. Zhang, C. Lan, and A. Perrig, "Secure and scalable fault localization under dynamic traffic patterns," in *IEEE Symposium on Security and Privacy, San Francisco, California, USA, 21-23 May*, 2012, pp. 317–331.
[49] Q. Li, X. Zou, Q. Huang, J. Zheng, and P. P. C. Lee, "Dynamic packet forwarding verification in SDN," *IEEE Trans. Dependable Secur. Comput.*, vol. 16, no. 6, pp. 915–929, 2019.
[50] 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, Vancouver, BC, Canada, October 17-20*, 2011, pp. 143–154.

Songtao Fu received the Ph.D. degree from Tsinghua University, Beijing, China. 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 an Editorial Board Member of the IEEE TDSC and ACM DTRAP.

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

Su Yao received his Ph.D. degree from the National Engineering Laboratory for Next Generation Internet Interconnection Devices, Beijing Jiaotong University, Beijing, China. Currently, he serves in 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, IoT security, and artificial intelligence for network system.

Xuewei Feng received the Ph.D. degree with Tsinghua University, Beijing, China. His research interests include Internet architecture and network security.

Ziqiang Wang received B.E. degree from the School of Artificial Intelligence Xidian University, Xi'an, China, in 2020. Currently, He is pursuing his Ph.D. in the School of Cyber Science and Engineering at Southeast University supervised by Prof. Ke Xu. His research interests include network architecture and security.

Xinle Du received the Ph.D. degree with Tsinghua University. His research interests include data-driven network, data center network transport protocol and AQM.

Kao Wan received his B.S degree from Peking University in 2008 and his Ph.D. degree from Tsinghua University in 2018. He now works as an assistant researcher in Peng Cheng laboratory of China. His research interests include networking, IoT, SDN, AI and network security..

Ke Xu (Fellow, IEEE) received his Ph.D. degree from the Department of Computer Science and Technology, Tsinghua University, Beijing, China, where he serves as a full professor. He has published more than 200 technical papers and holds 11 US patents in the research areas of next-generation Internet, Blockchain systems, Internet of Things, and network security. He is a member of ACM and senior member of IEEE. He has guest-edited several special issues in IEEE and Springer Journals, and also served as Steering Committee Chair of IEEE/ACM IWQoS. He is an editor of IEEE IoT Journal.

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on January 14,2025 at 07:11:02 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.