Robust and Lightweight Fault Localization

Bo Wu*† Ke Xu*† Qi Li *† Fan Yang *
*Department of Computer Science and Technology, Tsinghua University, Beijing, China
† Tsinghua National Laboratory for Information Science and Technology (TNList)
wub14@mails.tsinghua.edu.cn, xuke@tsinghua.edu.cn, qi.li@sz.tsinghua.edu.cn, y-f14@tinghua.org.cn

***Abstract*—The current network is vulnerable to various attacks, e.g., source spoofing and flow hijacking attacks, which can be constructed by misconfigurations or compromising routers. Unfortunately, both users and network operators are unable to localize these faults. Existing fault localization mechanisms detect such attacks under an assumption that localization is performed upon reliable communication channels. In this paper, we will relax the assumption and propose a robust and lightweight data-plane fault localization (RFL) protocol that aims to achieve source authenticity and path compliance in unreliable communication channels. RFL uses symmetric keys to build secure detection channels and samples packets for localization on the channels such that it can detect and localize faults. In particular, the localization performed is not impacted by the reliability of the communication channels, e.g., the packets that used to localize faults are dropped. We prototype of RFL on Click routers and the experiment results with the prototype demonstrate that RFL achieves more than 99.5% localization accuracy, while only incurring around 10% throughput degradation.**

I. Introduction

The reliable data delivery is highly desirable for users, e.g., for the purpose of providing security-critical services in modern ISP, enterprise, and datacenter networks [1], which requires correct delivery of packets along the purposed forwarding paths, and with the authentic origin. However, current networks always suffer from packet source spoofing and traffic hijacking attacks due to the existence of misconfiguration or attacks. Existing troubleshooting tools, e.g., ping and traceroute, cannot effectively identify and localize faults. In particular, an adversary may interfere with fault localization by dropping, modifying and redirecting packets. In this case, many attacks could be launched to counterfeit packet origins, change forwarding path and eavesdrop private information. Thus, data-plane fault localization for source and path verification is an essential remediation for securing data delivery.

In order to address this issue, end-to-end source authentications [2] [3] and path validations [4] [5] [6] have been extensively studied. They verify packet origins and forwarding paths by embedding cryptographic tags (markings). However, they assume that the packets used to verify markings can be correctly delivered. However, it is not always true. Specially, an adversary can easily interfere and drop the packets. Therefore, none of the existing schemes can accurately localize faults in unreliable networks without the help of centralized servers (e.g., the Internet). For example, OPT [7] and OSV [8] can perform source and path validation with either lightweight or efficient routers; without the exact acknowledgements (ACK) from intermediate router(s) and the receiver, the source host would not locate the misbehaving router. Although existing fault localization mechanisms [9] [10] could not localize the offending entity without reliable transmission channels. Centralization-based localization mechanisms [11] [12] only rely on central controller to localize the attacks.

Therefore, until now robust data-plane fault localization that resists to unreliable detection channels is not well addressed. Source and path verification could be still an effective way to localize faults in networks for packet source such that the packets can be detoured around the faults. However, it is difficult to achieve the goal. As we mentioned above, fault localization should be tolerant to reliable detection channels. More specifically, it should resist to interference from adversaries, e.g., they can drop, modify, and redirect attacks to interfere with the localization. Moreover, such process should not incur significant communication overhead in networks.

We propose RFL protocol, a robust and lightweight fault localization protocol, to ensure source authenticity and path compliance and enable fault localization in networks. RFL leverages a packet sampling mechanism to sign and verify packets, and applies an acknowledgement mechanism to evaluate the verified packets. Thereby, it ensures correct fault localization in unreliable networks. RFL enables a timer for each entity that will expire if the packet sent by the entity is dropped, modified, and redirected during packet verification. Thereby, each entity can effectively perform source and path verification to filter unwanted packets. Moreover, all packets will be sampled at each hop according a probabilistic sampling function. By collecting the sampling information in each entity, the packet source could identify and locate the misbehaved router. By such a sampling mechanism, RFL significantly reduces the overhead incurred by packet verification. Our theoretically analysis shows RFL introduces small overhead. Specially, it only incurs around 6.03% communication overhead, which outperforms the existing schemes. We prototype RFL based on the Click Modular Router, and use experiment results to demonstrate the performance of RFL. The experimental results show that RFL achieves more than 99.5% localization accuracy, and obtain more than 90% throughput and 85% goodput. Therefore, RFL incurs negligible performance overhead, while ensuring correctness of localization.

The contributions of this paper are four-fold:

- We propose RFL for source authenticity and path compliance verification, which resists to unreliable communication detection channel without involvement of central servers.
- We develop a robust fault localization to locate misbehaved entities during the source and path verification.
- We perform theoretical analysis of RFL and real experiment upon RFL prototype to demonstrate the performance of RFL.

978-1-5090-6468-7/17/$31.00 ©2017 IEEE

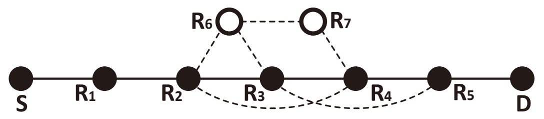


Fig. 1. Adversary model where R_2 is the misbehaved router and the purposed forwarding path is $\Psi = \langle R_1, R_2, R_3, R_4, R_5 \rangle$ between S and D.

II. Problem Statement

In this section, we formalize the adversary model, problem statement and our research scope. For generality, we consider an end-to-end communication in multi-hop network where the packets are delivered through routers from a source to a destination. We denote the source by S, the routers in a path by R_i ($1 \leq i \leq n$) and the destination by D, where n is the path length (not including S and D). We define Ψ as the purposed forwarding path and the packets should be transmitted along it, ideally.

A. Adversary Model

In this paper, we regard both compromised and misconfigured router as the misbehaved router as the compromised router is controlled by an adversary for launching attacks, and the misconfigured router has security threat for data-plane path consistency. More intuitively, Fig. 1 is used to understand adversary model, where we assume R_2 is the misbehaved router and $\Psi = \langle R_1, R_2, R_3, R_4, R_5 \rangle$. In this case, R_2 could modify the source address of IP packets originating from S for launching **source spoofing** attack. Besides, R_2 could make the packets delivered along a path that differs from Ψ, e.g., $\langle R_2, R_6, R_3 \rangle$, $\langle R_2, R_6, R_7, R_4 \rangle$ and $\langle R_2, R_4 \rangle$ for the purpose of **path inconsistency** attack. If R_2 colludes with others, the packets could be transmitted along the unordered routers on Ψ, e.g., $\langle R_2, R_4, R_3, R_5 \rangle$.

During the verification of source authenticity and path consistency, localizing the misbehaved router is a higher requirement if any error occurs. However, in order to evade the fault localization, the misbehaved router could launch **sophisticated attacks** that can be divided into two categories. Firstly, the misbehaved router could destroy or disturb fault localization by unexpectedly discarding, modifying and redirecting some messages used to localize the fault. For example, when S tries to obtain verification messages from intermediate routers on Ψ, R_2 could drop *request* packets (from S to R_i) or ACK packets (from R_i to S) to destroy this procedure; or when R_3 reports ACK packet to S, R_2 could modify this packet data to disturb the fault localization. Secondly, the misbehaved router could frame others of their unrealistic misbehavior by launching frame attack. For example, when receiving a packet, R_2 greatly reduces the TTL value to 2, causing the illusion that R_4 is regarded as the fault due to its dropping the packet.

B. Problem Formulation

This paper focuses on the data-plane fault localization for source authenticity and path compliance. We formulate the several problems as the following definition shows.

Definition 1. We divide the end-to-end communication session into consecutive ***epoch***s, varying with different sessions and their phases. For one session, after a period of time or sending an amount of packets, the *epoch* value would be switched.

Definition 2. The ***timer*** $\mathcal{T}_S$ and $\mathcal{T}_i$ are running on S and R_i on Ψ, respectively. They starts when the request package arrives and expires after a certain timeout, called ***timer threshold*** that can be evaluated by a *round-trip time* (RTT).

Definition 3. In this paper, we denote ***fault localization*** as to localize the misbehaved router as well as its one neighbor, because accurately localizing the misbehaved router is impossible according to the research [13]. Of course, for the higher requirements, centralization-based mechanisms (e.g., VeriDP [11]) in SDN, could be employed, which is not suitable for end-to-end communication.

Definition 4. We define ***positive ratio*** denoted by $\mathcal{P}_i$ and $\mathcal{P}_D$ for R_i and D, which illustrates the probability that the corresponding entity is a misbehaved router. When $\mathcal{P}_\tau$ is greater than ***positive ratio threshold*** (denoted by ζ), and $\mathcal{P}_1$, $\cdots$, $\mathcal{P}_{\tau\text{-}1}$ are all less than ζ, we could identify R_τ or $R_{\tau\text{-}1}$ as the misbehaved router (detailed in Section V).

C. Scope and Assumptions

Since we focus on the data-plane fault localization for source authenticity and path consistency while the packet is intended to be delivered through Ψ, we assume the end hosts (S and D) are all trusted, because it is meaningless for a malicious source to detect the security of source and forwarding path. As we perform data-plane detection, we assume S knows Ψ, which could be learned from the existing control-data plane routing protocols [14] [15]. As for the secret key distribution, we assume the public/private key of each entity on Ψ is long-lived and the public key could be gained and verified by others. In this paper, we employ Dynamically Recreatable Key (DRKey) protocol [7] to establishes symmetric keys shared between routers and endhosts, in which routers don't need to store the keys that prevents state exhaustion DoS attacks.

III. Overview of RFL

We now describe the overview of RFL protocol that localizes faults even in unreliable transmission channels, e.g., in the presence of interference from adversaries. Fig. 2 shows the workflow of RFL protocol. Before each packet's departure, S firstly inserts and initializes RFL header. Each entity would verify and probabilistically sample the packets on receiving them. At the end of each epoch, S would perform fault localization according to the sampled information of entities.

RFL protocol initialization. S precomputes a marking for each entity on Ψ before sending out a data packet. All markings are inserted into a new header called RFL header between IP header and TCP/UDP header.

Source and path verification. Once receiving the packet, R_i recalculates its own marking using symmetric key K_{S,R_i} and compares it with the inserted one on RFL header. Only these two values are equal, could R_i forward the packet to downstream routers.

Fault localization. RFL samples packets to localize faults. S uses the packet sampling information of each entity on Ψ to localize the fault. Our developed probabilistic packet sampling function $\mathcal{F}$ (described in Section V) determines which entity would sample the packet in one epoch. Each entity's packet sampling results can be only predicted by S, but unknowable to others. At the end of each epoch, the message carrying the encrypted sampling information would be delivered from D to S. Thereby, based on the received ACK message, S determines where a fault occurs (detailed in Section V).

However, it is challenging to implement RFL because of the following issues:

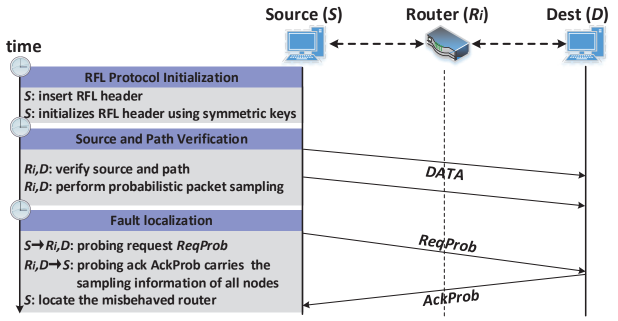

Source (S)
Router (Ri)
Dest (D)
time
RFL Protocol Initialization
S: insert RFL header
S: initializes RFL header using symmetric keys
Source and Path Verification
Ri,D: verify source and path
Ri,D: perform probabilistic packet sampling
DATA
Fault localization
S→Ri,D: probing request ReqProb
Ri,D→S: probing ack AckProb carries the sampling information of all nodes
S: locate the misbehaved router
ReqProb
AckProb

Fig. 2. The workflow of RFL protocol for the source (S), intermediate routers (R_i) and the destination (D), where *ReqProb* and *AckProb* respectively denote the request and acknowledgement messages for fault localization probing.

Algorithm 1 ReqProb and AckProb Initialization.

```
function ReqProb INITIALIZATION BY S ( )
Require: $\Psi$, $SessionID$, $epoch$, $S$, $D$
Compute:
    $M_{cst}^{req} = \Psi||SessionID||epoch||S||D$
    $M_D^{req} = PRF_{K_{S,D}}(M_{cst}^{req}||TTL_D||R_n)$
    for $i$ from $n$ to 1 do
        $M_i^{req} = PRF_{K_{S,D}}(M_{cst}^{req}||TTL_i||R_{i-1}||M_{i+1}^{req}||\cdots||M_n^{req}||M_D^{req})$
    end for
    $M_{Path}^{req} = \langle M_1^{req}, \cdots, M_n^{req}, M_D^{req}\rangle$
    $ReqProb = \{\Psi, SessionID, epoch, M_{Path}^{req}\}$
    S delivers REQProb to intermediate routers towards D.
end function
function AckProb INITIALIZATION BY D ( )
Require: $\Psi$, $SessionID$, $epoch$, $\mathcal{B}_d^e$, $K_S$, $K_D^{-1}$
Compute:
    $Enc\mathcal{B}_d^e = Enc_{K_{S,D}}(\mathcal{B}_d^e)$
    $Sign\mathcal{B}_d^e = Sign_{K_D^{-1}}(H(K_S||\Psi||SessionID||epoch||TTL_d^{ack}||Enc\mathcal{B}_d^e))$
    $AckProb = \{\Psi, SessionID, epoch, Enc\mathcal{B}_d^e, Sign\mathcal{B}_d^e\}$
Forwarding:
    D delivers ReqProb to intermediate routers towards S.
end function
```

Unreliable transmission of request and ACK packet. When S sends request packet ReqProb, the misbehaved router could drop or redirect this packet to disturb fault localization. Thereby, any entities cannot reply packet sampling back to S, incurring the failure of localization. Also, the router could modify the data of ACK packet AckProb. Even though S can capture the modified data by checking signatures, it could not identify and localize the fault.

Resistance to frame attacks. To evade localization, a misbehaved router could launch TTL attack by significantly decreasing TTL value of any packets so as to frame other entities. Moreover, when performing source and path verification, the misbehaved entity could modify the pre-inserted marks in RFL header, resulting in the failure of packet verification, or drop packets to disturb the fault localization.

IV. Source and Path Verification Details

In this section, we detailedly describe the design details of source and path verification. Concretely, S uses the obtained symmetric keys to precompute the markings for each entity on Ψ and inserts these markings into RFL header, a new packet header between IP header and TCP header. During the packet delivery towards D, each entity verifies the packet by recalculating the marking.

A. RFL Protocol Initialization

Before packet departure, S inserts RFL header between IP header and TCP header with the following structure:

$$RFL\ header = \{SessionID, epoch, PacketID, M_{Path}\} \tag{1}$$

$SessionID$=$H(K_S||K_D||T_{start})$ is the session identifier that is the hash over end-hosts' public keys (K_S and K_D) and session's start time T_{start}. *PacketID* is the unique packet identifier, calculated based on Eq. 2:

$$PacketID = H(SessionID||epoch||IP_{cst}), \tag{2}$$

where IP_{cst} represents constant portion of IP packet (excluding variable fields, such as TTL and checksum) during forwarding. The M_{Path} (in Eq. 1) denotes the precomputed marking sequence for later verification at each entity, as Eq. 3 shows.

$$M_{Path} = \langle M_1, M_2, \cdots, M_n, M_D\rangle \tag{3}$$

M_i is precalculated for each entity using PRF keyed with the shared symmetric keys, as Eq. 4 shows, where M_{cst}^{in} is the splice of constant input : $SessionID||epoch||PacketID||S||D$. Adding source and destination address (denoted by S and D) as the input could also help to defense against both source spoofing and traffic redirection, caused by modifying source and destination address of IP packet.

In Eq. 4, TTL_i and TTL_D respectively donates the purposed TTL value when the packet arrives at R_i and D. The marking is precomputed for the entity on the reverse Ψ. More especially, all downstream entities' markings are added as the input, which prevents frame attack caused by malicious modification of the pre-inserted markings of downstream entities.

$$\begin{aligned} M_D &= PRF_{K_{S,D}}(M_{cst}^{in}||TTL_D||R_n) \\ M_i &= PRF_{K_{S,R_i}}(M_{cst}^{in}||TTL_i||R_{i\text{-}1}||M_{i+1}||\cdots||M_n||M_D) \end{aligned} \tag{4}$$

The fields *SessionID*, *epoch* and *PacketID* in RFL header respectively occupies 128 bits, 16 bits and 128 bits. Each marking occupies 32 bits, whose rationality analysis is shown in Section VII.

B. Source and Path Verification

After RFL header initialization, each packet departs from S and is delivered through intermediate routers towards D. During the transmission, the packet is verified for its origin and forwarding path at each hop. Concretely, each entity uses Eq. 4 to recalculate the marking M_i'. If the computed M_i' equals to M_i in RFL header, it illustrates that the source and path is correct up to the current entity. Else, i.e., the verification fails, the packet would is dropped at this hop. This verification by each entity for the received packet could prevent the state exhaustion attacks on D.

V. Fault Localization

In this section, we show the details of fault localization on the misbehaved router. For every epoch, S respectively establishes one *bloom filter* [16] for each entity. We define $\mathcal{B}_i^e$ and $\mathcal{B}_D^e$ as the bloom filter with L-bits length for the packet sampling on R_i and D at epoch e.

Before each packet's departure, S uses probabilistic packet sampling function $\mathcal{F}$ (detailed below) to learn which entity would sample this packet. For example, if R_η would sample this packet according to $\mathcal{F}$, S samples this packet in $\mathcal{B}_\eta^e$. $\mathcal{F}$ determines R_i's packet sampling, which only known between S and R_i. Concretely, with the symmetric keys and *PacketID* in RFL header, S computes and gains the 128-bits hash value: $H_{sampling}$=$H(K_{S,R_i}||PacketID)$. We define ω as the number of selected bottom bits of $H_{sampling}$. If ω binaries are all equal to 0, i.e., no 1 appears in this lower ω bits binary, this packet would be sampled on the corresponding entity. In this case, ρ-th bit in the corresponding $\mathcal{B}_i^e$ or $\mathcal{B}_D^e$ would be switched from 0 to 1, where $\rho = PRF(H_{sampling}), 0 \leq \rho < L$. If any collision occurs, the next bit i.e., (ρ+1)-th bit, would be switched until no collision occurs. At the same time, each entity establishes two local bloom filters, one ($\mathcal{B}_{R_i}^e$ or $\mathcal{B}_d^e$) for the current epoch and another ($\mathcal{B}_{R_i}^{e+1}$ or $\mathcal{B}_d^{e+1}$) for the next. Note that the storage overhead is analyzed in Section VII. Then packet sampling is carried out at each entity according to $\mathcal{F}$, where the result is stored in the local bloom filter and could only be known by the corresponding entity and S.

At the end of each epoch, S tries to obtain the packet sampling information of all entities for localizing the fault by sending the request packet called *ReqProb*, which is initialized as algorithm 1 shows. On receiving ReqProb, R_i firstly performs source and path verification via recomputing M_i^{req}. Then R_i starts the timer $\mathcal{T}_i$ and forwards ReqProb to the downstream router towards D. When receiving ReqProb, D initializes the probing ACK packet, called *AckProb*, according to algorithm 1, in which the encrypted $\mathcal{B}_d^e$ and D's signature are all added in AckProb. Then D sends AckProb back through routers on Ψ until it arrives at S. During AckProb transmission, each router R_i firstly checks all signatures in AckProb. If no error occurs, R_i inserts its encrypted $\mathcal{B}_{R_i}^e$ and signature in AckProb.

$$\begin{aligned} AckProb = \{&\Psi, SessionID, epoch, Enc\mathcal{B}_d^e, Sign\mathcal{B}_d^e \\ &Enc\mathcal{B}_{R_n}^e, Sign\mathcal{B}_{R_n}^e, \cdots, Enc\mathcal{B}_{R_i}^e, Sign\mathcal{B}_{R_i}^e\}, \end{aligned} \tag{5}$$

where $Sign\mathcal{B}_{R_i}^e$ uses all encrypted bloom filters of entities from R_i to D as the calculation input, as shown in Eq. 6, to prevent the previous encrypted bloom filters from malicious modification and avoid frame attack. If $\mathcal{T}_i$ on R_i expires or the forwarding path is asymmetric[1], R_i would create AckProb$_i$ to send its sampling information back to S. On receiving AckProb$_i$, R_j (0< j <i) would also check the signatures and add its $\mathcal{B}_{R_j}^e$ and $Sign\mathcal{B}_{R_j}^e$ to *AckProb$_i$*.

$$\begin{aligned} Sign\mathcal{B}_{R_i}^e = \ &Sign_{K_i^{-1}}(H(K_S||\Psi||SessionID||epoch \\ &||TTL_i^{ack}||Enc\mathcal{B}_d^e||Enc\mathcal{B}_{R_n}^e||\cdots||Enc\mathcal{B}_{R_i}^e)) \end{aligned} \tag{6}$$

***Positive-ratio-based* fault localization.** As algorithm 2 shows, when receiving *AckPorb*, S firstly checks the signatures $Sign\mathcal{B}_{R_i}^e$ (1≤ i ≤n) and $Sign\mathcal{B}_D^e$ to verify the validity of *AckPorb*. If any error occurs, S could locate R_1 as a misbehaved router. Concretely, if R_ε (1< ε ≤n) on Ψ modified *AckPorb*, $R_{\varepsilon\text{-}1}, \cdots, R_1$ would all discard *AckPorb* when receiving it. In this case, if *AckPorb* with error signatures arrives at S, R_1 either modify it or ignores the signature verifications in collusion attack. Note that if no *AckProb* is received, S could directly locate R_1 as the fault.

After S verifies all signatures, $\mathcal{B}_{R_i}^e$ and $\mathcal{B}_d^e$ could be obtained via decrypting $Enc\mathcal{B}_{R_i}^e$ and $Enc\mathcal{B}_d^e$. We define $\mathcal{C}_i$ ($\mathcal{C}_d$) as the count of packet sampling difference between precomputed sampling $\mathcal{B}_i^e$ ($\mathcal{B}_D^e$) and actual sampling $\mathcal{B}_{R_i}^e$ ($\mathcal{B}_d^e$), which actually is the number of binary 1 in $\mathcal{B}_i^e \bigoplus \mathcal{B}_{R_i}^e$ ($\mathcal{B}_D^e \bigoplus \mathcal{B}_d^e$). In RFL protocol, *positive ratio* $\mathcal{P}_i$ ($\mathcal{P}_D$) is the ratio of $\mathcal{C}_i$ ($\mathcal{C}_d$) to bloom filter length L. S tries to find $\mathcal{P}_\tau$ (τ=1,$\cdots$, n, D) that meets $\mathcal{P}_\tau \geq \zeta_\tau$ and $\mathcal{P}_{\tau\text{-}1} < \zeta_{\tau\text{-}1}$, where ζ is the threshold of positive ratio of entity (detailed in Section VIII). In this case, $\langle R_{\tau\text{-}1}, R_\tau \rangle$

[1]Actually, many forwarding paths in today's Internet are asymmetric[17].

Algorithm 2 *Positive-ratio-based* Fault localization.

```
1: function FAULT LOCALIZATION ( )
2: Require: $AckPorb$, $K_i$, $\mathcal{B}_i^e$ (1≤ $i$ ≤$n$), $K_D$, $\mathcal{B}_D^e$, $\zeta$
3:     for 1≤ $i$ ≤$n$ do
4:         if !($CheckSig_{K_i}(Sign\mathcal{B}_{R_i}^e)$ & $CheckSig_{K_D}(Sign\mathcal{B}_{R_D}^e)$) then
5:             $R_1$ is located as the misbehaved router.
6:         end if
7:         Compute: $\mathcal{B}_{R_i}^e = Dec_{K_{S,R_i}}(Enc\mathcal{B}_{R_i}^e)$, $\mathcal{B}_{i,R_i}^e = \mathcal{B}_i^e \bigoplus \mathcal{B}_{R_i}^e$
8:     end for
9:     Compute: $\mathcal{B}_d^e = Dec_{K_{S,D}}(Enc\mathcal{B}_d^e)$, $\mathcal{B}_{D,d}^e = \mathcal{B}_D^e \bigoplus \mathcal{B}_d^e$
10:    for 1≤ $i$ ≤$n$ do
11:        $\mathcal{C}_i$:Count binary 1 in $\mathcal{B}_i^e$, $\mathcal{C}_i'$:Count binary 1 in $\mathcal{B}_{i,R_i}^e$
12:        Compute: $\mathcal{P}_i = \mathcal{C}_i/\mathcal{C}_i'$
13:    end for
14:    $\mathcal{C}_D$:Count binary 1 in $\mathcal{B}_D^e$, $\mathcal{C}_D'$:Count binary 1 in $\mathcal{B}_{D,d}^e$
15:    Compute: $\mathcal{P}_D = \mathcal{C}_D/\mathcal{C}_D'$, $\mathcal{P}_0 = 0$, $\mathcal{P}_{D\text{-}1} = \mathcal{P}_n$
16:    for $\tau$ from 1 to n and D do
17:        if $\mathcal{P}_\tau \geq \zeta$ & $\mathcal{P}_{\tau\text{-}1} < \zeta$ then
18:            $\langle R_{\tau\text{-}1}, R_\tau \rangle$ is located as the misbehaved router.
19:        end if
20:    end for
21: end function
```

is located as the misbehaved router, because one of $R_{\tau\text{-}1}$ and R_τ modified packet origin and path, disturbing actual packet sampling in $\mathcal{B}_{R_\tau}^e$. Note that if $\mathcal{P}_D \geq \zeta_D$ and $\mathcal{P}_n < \zeta_n$, R_n is then located by S. More generally, when S receives *AckProb$_i$* instead of *AckProb*, actual sampling of R_{i+1},$\cdots$, R_n, D would be regarded as empty set, i.e., $\mathcal{B}_{R_{i+1}}^e$=$\cdots$=$\mathcal{B}_{R_n}^e$=$\mathcal{B}_d^e$= $\varnothing$. Then the fault could also localized according to algorithm 2.

VI. Security Analysis

This section discusses RFL protocol's security against data-plane attacks of source authenticity and path consistency by misbehaved router(s). In our adversary model, the misbehaved router can modify source and forwarding path of received packets, disturb secret key establishment and try to evade fault localization. We show the RFL protocol is secure against a single misbehaved router (say, R_τ) as well as multiple colluding entities.

Security against source spoofing. Modifying source address of IP packet by R_τ will introduce the discrepancy of the downstream routers' markings between preinserted (by S) and recomputed (by downstream entities) values. For example, if R_τ corrupts source address, $R_{\tau+1}$ would drop the received packets, because the recalculated marking $M'_{\tau+1}$ (according to Eq. 4) is not equal to the inserted value $M_{\tau+1}$ in RFL header as the source address is also one input of marking calculation.

Security against path inconsistency. Corrupting the packets forwarding path will cause the non-correspondence between preinserted markings and downstream routers on actual path. For example, if R_τ delivers the packets to $R\varphi$ instead of $R_{\tau+1}$, $R\varphi$ would discard this packet, because the recalculated marking M'_φ does not equal $M_{\tau+1}$ in RFL header. As for collusion attack, we will discuss it shortly.

Security against corrupting fault localization. There are three methods for R_τ to corrupt fault localization. Firstly, R_τ could disturb source and path verification and packet sampling by means of modifying the downstream routers' markings in RFL header. For example, R_τ modifies $M_{\tau+2}$ in RFL header for the purpose that $R_{\tau+2}$ drops this package and frames $\langle R_{\tau+1}, R_{\tau+2} \rangle$ of misbehavior. In RFL protocol, each router R_i uses the downstream entities' markings (M_{i+1},$\cdots$, M_D) in RFL header as the input to recompute the marking M'_i, as shown in Eq. 4. Thus, $M_{\tau+1}$ would drop the package and $\langle R_\tau, R_{\tau+1}\rangle$ would be regarded as the fault if R_τ corrupts $M_{\tau+\Delta}$ ($2 \leq \Delta \leq n-\tau$) in RFL header.

Secondly, R_τ could drop, modify and redirect *ReqProb* and *AckProb* packet to prevent S from obtaining sampling information of routers and D. In RFL protocol, if *timer* $\mathcal{T}_{\tau\text{-}1}$ expires, *AckProb*$_{\tau\text{-}1}$ packet would be initialized and sent back to S. According to the *positive-ratio-based* fault localization, $\mathcal{P}_\tau \geq \zeta$ and $\mathcal{P}_{\tau\text{-}1} < \zeta$ could make $\langle R_{\tau\text{-}1}, R_\tau\rangle$ easily located.

Thirdly, R_τ could frame other entities on Ψ by launching TTL attack. For example, R_τ lowers the TTL value of *ReqProb* packet to a smaller value (say, κ) with the purpose that $R_{\tau+\kappa}$ drops this packet and is located as the fault. RFL protocol could address TTL attack, in which TTL value is added to compute the signatures on *AckProb*$_{(i)}$ packet during symmetric keys establishment and fault localization, and to calculate the markings on *ReqProb* (see Eq. 4) at RFL header initialization stage. Therefore, if R_τ launches TTL attack, the packet would be discarded at next hop. In this case, R_τ and one of its neighbor on Ψ would be located as the fault.

Security against collusion attacks. All attacks discussed above can be launched by colluding routers. We can prove by induction that RFL protocol works well to defense against collusion attacks. We give a proof sketch as following:

Proof: We assume there is another router (denoted by R_σ, $\tau < \sigma \leq n$) on Ψ colluding with R_τ. Without loss of generality:

1) In the first case where R_τ is not adjacent to R_σ (i.e., $\sigma > \tau+1$), (i) if R_τ launches the above source spoofing or path inconsistency attacks while the packets are forwarded, the intermediate routers between R_τ and R_σ would also perform the verification for source and path, and then drop the corrupted packets. (ii) If R_τ corrupts fault localization in any case of three methods described above, R_τ and its one neighbor would be located as the fault.

2) In the other case where R_τ and R_σ are adjacent to each other (i.e., $\sigma=\tau+1$), we can regard these two routers as one single "virtual" misbehaved router R_v with upstream router $R_{\tau\text{-}1}$ and downstream router $R_{\tau+2}$. (i) If R_v launches the above source spoofing or path inconsistency attack described above, $R_{\tau+2}$ would find the verification for packet origin and path fails, and then drop the corrupted packet. (ii) If R_v corrupts the fault localization, S could also obtain the packet sampling information of $R_1,\cdots,R_{\tau\text{-}1}$ (i.e., $\mathcal{B}^e_{R_1},\cdots,\mathcal{B}^e_{R_{\tau-1}}$). In this case, at least one router of R_τ and R_σ would be located as the misbehaved router.

VII. Theoretical Analysis

In this section, we provide the theoretical analysis for some parameters, communication overhead and storage overhead.

A. Communication Overhead

In RFL protocol, the RFL header is the additional communication overhead. From Section IV-A, (*38+4n*) bytes are occupied in RFL header, where n is the length of Ψ. According to the research [18], the average end-to-end path length of Internet is 13.11 hops, causing the communication overhead of 90.44 bytes in RFL protocol. It is worth mentioning that about 85% data of Internet is transmitted by large (>1400 bytes) packet [19]. In this case, with packet size P_{size} = 1500 bytes (Maximum Transmission Unit, MTU), RFL communication overhead accounts for 6.03% of the entire IP packet.

Compared with other related mechanisms [7] [8] [9] [20], RFL protocol outperforms in terms of communication overhead and its ratio under average path length and large packet (1500 bytes) of Internet, as shown in Table I.

TABLE I. COMMUNICATION OVERHEAD COMPARISON

	RFL	*OPT*	*ICING*	*OSV*	*Faultprints*
com-overhead (Byte)	**90.44**	277.76	563.62	134.22	160.88
com-overhead ratio (%)	**6.03%**	18.52%	37.57%	8.95%	10.73%

Com-overhead is short for communication overhead.

B. Verification Rationalization

From Section IV-A, we could learn each precomputed marking M_i occupies 32 bits. The verification at each hop is mainly performed by employing *PRF* to recompute the marking M_i. We must accept that although the inputs are different, there is also the probability, donated by φ, to result in the same output of *PRF*, in which the collision occurs. As every bit has the equal collision probability (i.e., 0.5), φ is the probability that collision of all 32 bits occurs at the same time, i.e., $\varphi = \frac{1}{2^{32}}$. This illustrates sending 2^{32} packets or $2^{32} \cdot 1500 > 2^{42}$=4 TB data (almost impossible for the normal end-to-end communication) would only cause this verification collision, averagely. Therefore, RFL protocol could perform rationally verification for source authenticity and path consistency as φ is small enough.

C. Bloom Filter Size

In RFL, end-hosts and routers all store at least two *bloom filters*, which is an important factor in determining their storage overhead. Although RFL tries to decrease the storage requirements, blindly reducing *bloom filter* size is not advisable. On one hand, *bloom filter* should be sufficient to store the packet sampling information of one *epoch*, where usage rate ξ^e_i of $\mathcal{B}^e_i$ does not exceed usage rate threshold ξ^e_{th}. This is also related to the link bandwidth (detailed in Section VII-E). On the other hand, false positive rate increases with *bloom filter* size decreases. For one *epoch e*, at most $L\cdot\xi^e_{th}$ packets would be sampled, causing C(L, $L\cdot\xi^e_{th}$) sampling results, where C($\cdot$) donates the number of combinations of L and $L\cdot\xi^e_{th}$. So the false positive rate is $\mathbb{F}=C(L,\ L\cdot\xi^e_{th})^{-1}$.

In this paper, we set *bloom filter* size L=1 Kb and usage rate threshold ξ^e_{th}= 0.8. In this case, $\mathbb{F} \ll 0.001\%$, which is reasonable in RFL protocol.

D. Storage Overhead

Router storage overhead. Using DRKey protocol, each intermediate router or the destination don't have to store the symmetric key for per-path or per-source. As for each session, R_i establishes two *bloom filters* for current and next epoch. Therefore, the storage overhead of source for per-session is *2L* bits. From CAIDA [19], 12.91K application sessions, on average, are observed in a router per-second. So each router's storage overhead is $\frac{12.91K\cdot 2L}{8\cdot 1024^2}$ = 3.23 MB.

End-host storage overhead. For one session, S stores the symmetric keys shared with entities on Ψ. With 128 bits length of each secret key, 16(n+1) bytes is occupied to store the keys. For each router and D, two *bloom filters* are established, resulting in extra storage overhead of 2(n+1)L bytes in S. So the source storage overhead for one session are (16+2L)(n+1) bytes. On the destination host, only two *bloom filters* should be stored for the packet sampling and localization, introducing its storage overhead of $\frac{2L}{8}$=0.25L bytes. With the average path

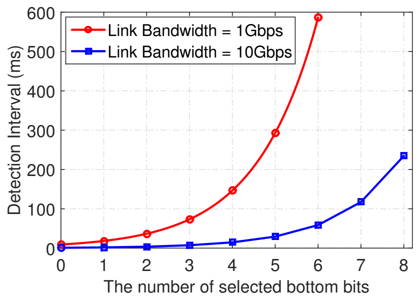


Fig. 3. Detection interval T_{MI} *vs.* the number of selected bottom bits ω.

length of Internet, the storage overhead of S and D are 146.28 bytes and 3.28 bytes, respectively.

E. Detection Interval

When S switches *epoch* value, *ReqProb* packet is forwarded to all the routers and D on *Path* for requiring packet sampling information. We respectively define T_{MI} and W_B as the detection interval and link bandwidth. As the existence of other sessions, there are at most $\frac{W_B}{P_{size}}$ packets and $\frac{W_B}{2^\omega \cdot P_{size}}$ packets of current session and epoch delivered and sampled by router or D per-second.

$$\frac{W_B}{2^\omega \cdot P_{size}} \cdot T_{MI} = L \cdot \xi_{th}^e \Rightarrow T_{MI} = \frac{L \cdot \xi_{th}^e \cdot 2^\omega \cdot P_{size}}{W_B} \quad (7)$$

Eq. 7 shows the *bloom filter* is used up to usage rate threshold ξ_{th}^e after time interval T_{MI}. When L=1 Kb, ξ_{th}^e=80%, P_{size}=1500 bytes and W_B =1 Gbps, we can learn T_{MI} increases as $\mathcal{N}$ increases at the link bandwidth of both 1 Gbps and 10 Gbps, as Fig. 3 shows. According to [9] [21], with the average value of 225 *ms*, T_{MI} between 100 *ms* and 350 *ms* conforms to the realistic network. In this case, $\omega = 5$, $T_{MI} = 292.97$ *ms* when W_B =1 Gbps and $\omega = 8$, $T_{MI} = 234.38$ *ms* when W_B =10 Gbps meet the requirements of detection interval in realistic network, respectively.

VIII. Performance Evaluation

A. Router Throughput and Goodput

In order to evaluate the packet forwarding efficiency at routing entities, we implements a prototype of the router, called RFL router, using Click Modular Router [22], which runs on Ubuntu Linux 12.04 with Intel(R) Core(TM) i5-4590, CPU @ 3.30 GHz, 16GB memory and NIC of 1000 Mbps[2]. With the help of iperf [23], we achieve the communications between two computers (Intel(R) Core(TM) i5-4200U, CPU @ 1.6GHz/2.3 GHz, 12.0 GB memory) through RFL router.

RFL router performs the following operations when delivering packets: source and path validation, probabilistic packet sampling and storing sampling information in bloom filter. In this evaluation, we use SHA-3 algorithm to compute the hash values of long strings, such as the calculation of PacketID (see Eq. 2). For computing the PRF value, we use HMAC based on SHA-1. We respectively truncate the value of SHA-3 and HMAC to meet our requirements, such as from 256 bits to 128 bits for computing PacketID and from 160 bits to 32 bits for computing M_i.

From Eq. 4, we learn RFL routers close to S have higher computation overhead than the router close to D as the longer input when recomputing the markings. In this case, we evaluate the middle router $R_{\lceil\frac{n}{2}\rceil}$ for the average-case analysis. Fig. 4(a) shows the relationship between packet size and forwarding efficiency with the average Internet path length of 13 hops. We calculate goodput as the valid throughput of useful packet data, excluding RFL header. Note that the smallest packet size is 130 bytes, including 90-byte RFL header and 40-byte IP/TCP header. We adjust packet size of 130 bytes to 1500 bytes by configuring the interface Maximum Transmission Unit (MTU) sizes. From Fig. 4(a), we could know both throughput and goodput of RFL router increase with the improvement of packet size. Especially for large packet of 1500 bytes, RFL router can achieve over 90% throughput and about 85% goodput of baseline. From [18], we learn that the path length of end-to-end communication is 15.3 ± 4.2 hops for IPv4 packets. Thus, we evaluate the packet forwarding efficiency with path length of 10 hops to 20 hops in Fig. 4(b). We can learn that RFL router's throughput and goodput all decrease when the forwarding path length increases, because more downstream routers' markings are added as the input of marking recomputation. Concretely, with path length increasement of 1 hop, the throughput and goodput would reduce by 9.26 Mbps. Fortunately, the throughput and goodput respectively exceed 850 Mbps and 800 Mbps in the networks of 13-hop path length, more than 90% and 85% compared to the baseline.

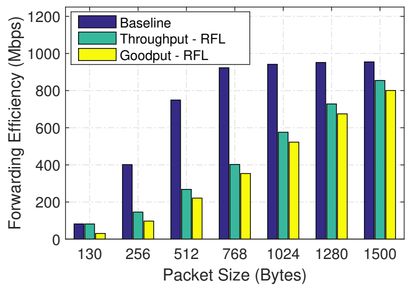
B. Positive Ratio threshold

We evaluate positive ratio threshold ζ_i of R_i through a simulation network with the path length of 20 hops, the longest forwarding path in end-to-end communication according to a CAIDA research [18]. In RFL, the misbehaviors (source spoofing and path inconsistency), and sophisticated attacks (frame and collusion attack) all lead to downstream entities' dropping the packet. We respectively define θ_{na} and θ_{mis} as the probability of natural loss and malicious loss (mis-loss) of the entity (as well as its upstream neighbored link), where $\theta_{mis} > \theta_{na}$.

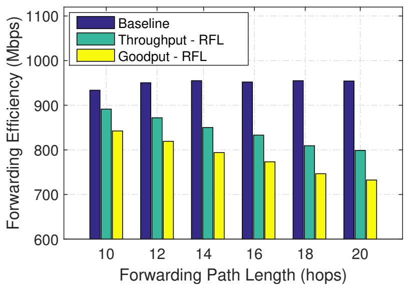
Fig. 4(c) shows the relationship between router location and its positive ratio with the variation of packet loss probability. To obtain more accurate result, we run our simulation over 100 times for each result. The red line depicts the scenarios only with natural packet loss (θ_{na}=0.001), which is also the threshold line that helps identify the misbehaved entity. When the packet loss probability is 0.00, the positive ratio $\mathcal{P}$ is lower than the threshold value, because lower packet loss brings about less packet loss during the transmission. We set one misbehaved router at different location from R_1 to R_{20}. The blue line (θ=0.10) and the green dotted line (θ=0.05) respectively shows the positive ratio $\mathcal{P}$ of the misbehaved router in different location. We learn that the positive $\mathcal{P}_i$ would exceed the threshold ζ_i when the malicious router behaves abnormally with the probability of θ=0.10 or θ=0.05. According to this threshold, S could locate the misbehaved router with different mis-loss probability.

We also evaluate the relationship between positive ratio and natural packet loss probability under different misbehaved packet loss probability. Fig. 4(d) shows the positive ratio of the middle entity (i.e., R_7) in 13-hop forwarding path. With the increasement of natural packet loss probability, the positive ratio threshold becomes larger, because the higher loss probability introduces more packets to be dropped. When the

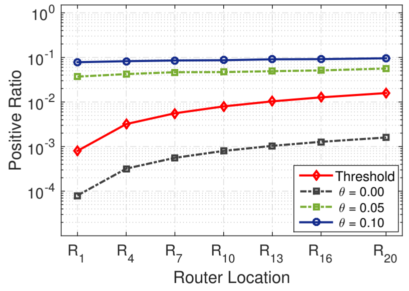
[2]Of course, it is also feasible for the similar packet processing as RFL to be implemented along fast TCAM path or in commercial router [9].



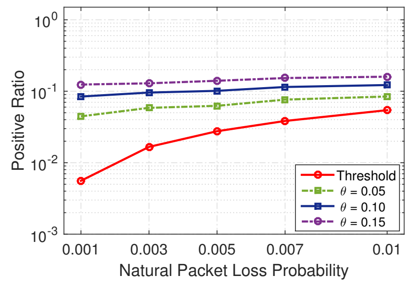
(a) RFL router's forwarding efficiency for different packet sizes under the Internet average path length.



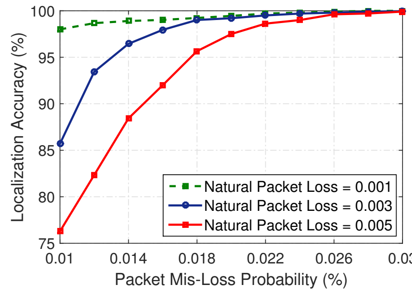
(b) RFL router's forwarding efficiency under different path lengths for the packet size of 1500 bytes.



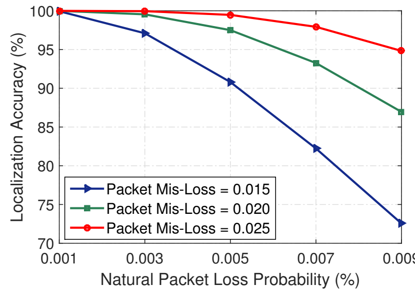
(c) The positive ratio of different router locations for different misbehaved packet loss probabilities.



(d) The relationship between positive ratio and different packet natural loss probabilities.



(e) The localization accuracy for different packet mis-loss or natural loss probabilities.



(f) The relationship between localization accuracy and natural packet loss probability.

Fig. 4. The RFL performance evaluation in terms of RFL router's forwarding efficiency, and positive ratio and localization accuracy of fault localization.

misbehaved packet loss probability is larger than natural loss, the corresponding ratios exceed the threshold value. Therefore, under different natural packet loss, the fault could also be localized in RFL protocol.

C. Localization Accuracy

Based on the positive ratio threshold above, we evaluate the fault localization accuracy denoted by δ through a simulation scenario with 13-hop forwarding path. We set one router of random location on forwarding path as the misbehaved router, that could launch both source spoofing and path inconsistency attacks. Besides, this misbehaved router could also disturb RFL protocol, which finally introduces the packets dropping.

We firstly evaluate the fault localization accuracy with the variation of packet mis-loss probability of misbehaved router, just as Fig.4(e) shows. From Fig. 4(e), we could learn the fault localization accuracy of RFL becomes higher with the increase of packet mis-loss probability. This is because more packet mis-loss results in higher positive ratio than the threshold. We respectively take the value of natural packet loss probability as 0.001, 0.003 and 0.005, which introduce different positive ratio thresholds (described in Section VIII-B). From Fig. 4(e), we knows less natural packet loss brings about higher localization accuracy, as the lower positive ratio threshold makes it easier to localize the fault.

Then the relationship between localization accuracy and natural packet loss probability is evaluated in Fig. 4(f). With the larger range of natural packet loss, we could learn localization accuracy becomes lower when more natural packet is dropped. Fortunately, under the smaller natural packet loss probability, such as 0.001 or 0.003, RFL achieves the fault localization with the accuracy of over 99.5% when the mis-loss probability is 0.020 or more.

IX. Related Work

Secure routing and forwarding. Routing security has been widely studied to ensure correct packet forwarding on the Internet [14] [24] [25]. S-BGP [6] verified the authenticity of announced routing paths by signing them, which incurs significant computation and communication overhead. In order to reduce the costs, a large amount of variants have been proposed. For example, So-BGP [26] ensured correctness of announced routing paths by leveraging network topologies. IRV [27] validated the correctness of the announced routing paths by establishing an additional IRV server in each AS, which limited its deployementability. All these approaches did not address the security of routing data plane.

Source and path verification. Origin and Path Trace (OPT) protocol [7] [28] allows each router to verify delivered packets so as to verify correctness of packet source and forwarding paths. It reduced storage overhead in routers, which prevents state exhaustion attack. Naous et. al., [20] proposed a Path Verification Mechanism (PVM) to validate whether the packets correctly forwarded their forwarding paths. Cai et. al., [8] performed source authentication and path validation by leveraging a set of orthogonal sequences instead of lightweight cryptographic operations. Unfortunately, these mechanism cannot localize the detected faults. Although Passport [2] and SNAPP [13] did not have such a problem, they were vulnerable to source spoofing or path deviation attacks.

Fault localization. There are a large amount of studies on locating data plane errors [9] [10] [12] [29] [30]. Faultprints [9] was the first secure inter-domain fault localization scheme. It could localize the misbehaved links that drop, delay, modify packets at a high speed. ShortMAC [10] leveraged probabilistic packet authentication to locate the illegal network links, which achieved low detection delay and incurred small overhead.

DynaFL [12] proposed the secure neighborhood-based fault localization (FL) protocol to cope with dynamic traffic patterns and routing path with small router state. TrueNet [29] leveraged trusted computing technology to build a trusted network-layer architecture, and implemented a small TCB to address secure FL with small router state. However, these schemes might failed to localize faults if the generated acknowledgement packets are maliciously dropped by colluding entities. Our proposed protocol well addresses this issue by setting the timer on each entity, where the entity would send its sampling information towards S once the timer expires.

X. Conclusion

In this paper, we propose RFL, a robust and lightweight data-plane fault localization protocol, which achieves high accuracy even over unreliable communication channels. RFL embeds cryptographic markings in packets so as to verify packet source and forwarding paths, and achieve fault localization, which ensures accuracy and robustness of the protocol even in the presence of interferences from adversaries. We prototype RFL and use real experiments based on the prototype to demonstrate the performance of RFL. The results show RFL achieves around 99.5% localization accuracy, and incurs small communication overhead, e.g., more than 90% throughput and 85% goodput compared to the baseline. We hope that the robustness and lightweight properties of RFL can become a fundamental primitive to construct highly secure and efficient data-plane protocols.

Acknowledgments

This work was supported by the National Natural Foundation of China (61472212 and 61572278), EU Marie Curie Actions CR-OWN (FP7-PEOPLE-2013-IRSES-610524), and the R&D Program of Shenzhen (JCYJ20170307153259323).

References

[1] Hongyi Zeng, Peyman Kazemian, George Varghese, and Nick McKeown. Automatic test packet generation. In *Proceedings of the 8th international conference on Emerging networking experiments and technologies*, pages 241–252. ACM, 2012.

[2] Xin Liu, Ang Li, Xiaowei Yang, and David Wetherall. Passport: Secure and adoptable source authentication. In *NSDI*, volume 8, pages 365–378, 2008.

[3] Adrian Perrig, Ran Canetti, Dawn Song, and J Doug Tygar. Efficient and secure source authentication for multicast. In *Network and Distributed System Security Symposium, NDSS*, volume 1, pages 35–46, 2001.

[4] Bryan Parno, Adrian Perrig, and Dave Andersen. Snapp: Stateless network-authenticated path pinning. In *Proceedings of the 2008 ACM symposium on Information, computer and communications security*, pages 168–178. ACM, 2008.

[5] Meiyuan Zhao, Sean W Smith, and David M Nicol. Aggregated path authentication for efficient bgp security. In *Proceedings of the 12th ACM conference on Computer and communications security*, pages 128–138. ACM, 2005.

[6] Stephen Kent, Charles Lynn, and Karen Seo. Secure border gateway protocol (s-bgp). *IEEE Journal on Selected areas in Communications*, 18(4):582–592, 2000.

[7] Tiffany Hyun-Jin Kim, Cristina Basescu, Limin Jia, Soo Bum Lee, Yih-Chun Hu, and Adrian Perrig. Lightweight source authentication and path validation. In *Proceedings of ACM SIGCOMM*, 2014.

[8] Hao Cai and Tilman Wolf. Source authentication and path validation with orthogonal network capabilities. In *Computer Communications Workshops (INFOCOM WKSHPS), 2015 IEEE Conference on*, pages 111–112. IEEE, 2015.

[9] Cristina Basescu, Yue-Hsun Lin, Haoming Zhang, and Adrian Perrig. High-speed inter-domain fault localization. In *Security and Privacy (SP), 2016 IEEE Symposium on*, pages 859–877. IEEE, 2016.

[10] Xin Zhang, Zongwei Zhou, Hsu-Chun Hsiao, Tiffany Hyun-Jin Kim, Adrian Perrig, and Patrick Tague. Shortmac: Efficient data-plane fault localization. In *NDSS*, 2012.

[11] Peng Zhang, Hao Li, Chengchen Hu, Liujia Hu, Lei Xiong, Ruilong Wang, and Yuemei Zhang. Mind the gap: Monitoring the control-data plane consistency in software defined networks. In *Proceedings of the 12th International on Conference on emerging Networking EXperiments and Technologies*, pages 19–33. ACM, 2016.

[12] Xin Zhang, Chang Lan, and Adrian Perrig. Secure and scalable fault localization under dynamic traffic patterns. In *Security and Privacy (SP), 2012 IEEE Symposium on*, pages 317–331. IEEE, 2012.

[13] Boaz Barak, Sharon Goldberg, and David Xiao. Protocols and lower bounds for failure localization in the internet. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 341–360. Springer, 2008.

[14] Yih-Chun Hu, Adrian Perrig, and Marvin Sirbu. Spv: Secure path vector routing for securing bgp. In *ACM SIGCOMM Computer Communication Review*, volume 34, pages 179–192. ACM, 2004.

[15] Sandra L Murphy and Madelyn R Badger. Digital signature protection of the ospf routing protocol. In *Network and Distributed System Security, 1996., Proceedings of the Symposium on*, pages 93–102. IEEE, 1996.

[16] Burton H Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.

[17] Wolfgang John, Maurizio Dusi, and Kimberly C Claffy. Estimating routing symmetry on single links by passive flow measurements. In *Proceedings of the 6th International Wireless Communications and Mobile Computing Conference*, pages 473–478. ACM, 2010.

[18] Bradley Huffaker, Marina Fomenkov, Daniel J Plummer, David Moore, and K Claffy. Distance metrics in the internet. In *Proc. of IEEE international telecommunications symposium (ITS)*, 2002.

[19] CAIDA. Passive monitor: equinix-chicago. http://www.caida.org/data/monitors/passive-equinix-chicago.xml.

[20] Jad Naous, Michael Walfish, Antonio Nicolosi, David Mazières, Michael Miller, and Arun Seehra. Verifying and enforcing network paths with icing. In *Proceedings of the Seventh COnference on emerging Networking EXperiments and Technologies*, page 30. ACM, 2011.

[21] CAIDA. round-trip time internet measurements. http://www.caida.org/research/performance/rtt/walrus0202/.

[22] Eddie Kohler, Robert Morris, Benjie Chen, John Jannotti, and M Frans Kaashoek. The click modular router. *ACM Transactions on Computer Systems (TOCS)*, 18(3):263–297, 2000.

[23] Ajay Tirumala, Feng Qin, Jon Dugan, Jim Ferguson, and Kevin Gibbs. Iperf: The tcp/udp bandwidth measurement tool. *htt p://dast. nlanr. net/Projects*, 2005.

[24] Catharina Candolin, Janne Lundberg, and Hannu Kari. Packet level authentication in military networks. In *Proceedings of the 6th Australian Information Warfare & IT Security Conference*, 2005.

[25] Guangwu Hu, Ke Xu, Jianping Wu, Yong Cui, and Fan Shi. A general framework of source address validation and traceback for ipv4/ipv6 transition scenarios. *IEEE Network*, 27(6):66–73, 2013.

[26] James Ng et al. Extensions to bgp to support secure origin bgp (sobgp). Technical report, Internet Draft, Apr, 2004.

[27] Geoffrey Goodell, William Aiello, Timothy Griffin, John Ioannidis, Patrick Drew McDaniel, and Aviel D Rubin. Working around bgp: An incremental approach to improving security and accuracy in interdomain routing. In *NDSS*, volume 23, page 156, 2003.

[28] Fuyuan Zhang, Limin Jia, Cristina Basescu, Tiffany Hyun-Jin Kim, Yih-Chun Hu, and Adrian Perrig. Mechanized network origin and path authenticity proofs. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 346–357. ACM, 2014.

[29] Xin Zhang, Zongwei Zhou, Geoff Hasker, Adrian Perrig, and Virgil Gligor. Network fault localization with small tcb. In *Network Protocols (ICNP), 2011 19th IEEE International Conference on*, pages 143–154. IEEE, 2011.

[30] Xiaoliang Wang, Ke Xu, and Ziwei Li. Smartfix: Indoor locating optimization algorithm for energy-constrained wearable devices. *Wireless Communications and Mobile Computing*, 2017, 2017.