Unveiling the Mystery of Internet Packet Forwarding: A Survey of Network Path Validation

KAI BU, Zhejiang University
AVERY LAIRD, Simon Fraser University
YUTIAN YANG and LINFENG CHENG, Zhejiang University
JIAQING LUO, University of Electronic Science and Technology of China
YINGJIU LI, University of Oregon
KUI REN, Zhejiang University, China

Validating the network paths taken by packets is critical in constructing a secure Internet architecture. Any feasible solution must both enforce packet forwarding along end-host specified paths and verify whether packets have taken those paths. However, the current Internet supports neither enforcement nor verification. Likely due to the radical changes to the Internet architecture and a long-standing confusion between routing and forwarding, only limited solutions for path validation exist in the literature. This survey article aims to reinvigorate research on the essential topic of path validation by crystallizing not only how path validation works but also where seemingly qualified solutions fall short. The analyses explore future research directions in path validation aimed at improving security, privacy, and efficiency.

CCS Concepts: • **Networks** → **Network architectures**; **Network protocols**; **Network security**; • **Security and privacy** → Cryptography; Network security;

Additional Key Words and Phrases: Future Internet architecture, packet forwarding, path validation, authentication

ACM Reference format:
Kai Bu, Avery Laird, Yutian Yang, Linfeng Cheng, Jiaqing Luo, Yingjiu Li, and Kui Ren. 2020. Unveiling the Mystery of Internet Packet Forwarding: A Survey of Network Path Validation. *ACM Comput. Surv.* 53, 5, Article 104 (September 2020), 34 pages.
https://doi.org/10.1145/3409796

This work was supported by the National Science Foundation of China (grants 61402404 and 61602093), Natural Science Foundation of Zhejiang Province (grant LY19F020050), National Natural Science Foundation of China (grant 61772236), Zhejiang Key R&D Plan (grant 2109C03133), Alibaba-Zhejiang University Joint Institute of Frontier Technologies; Research Institute of Cyberspace Governance in Zhejiang University, and Leading Innovative and Entrepreneur Team Introduction Program of Zhejiang (grant 2018R01005).
Authors' addresses: K. Bu, Y. Yang, L. Chen, and K. Ren, College of Computer Science and Technology, School of Cyber Science and Technology, Zhejiang University, 38 Zheda Road, Hangzhou 310027, China; emails: {kaibu, ytyang, chenglinfeng, kuiren}@zju.edu.cn; A. Laird, School of Computing Science, Simon Fraser University, Burnaby V5A 1S6, Canada; email: alaird@sfu.ca; J. Luo, School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China; email: csjluo@hotmail.com; Y. Li, Department of Computer and Information Science, University of Oregon, 1477 East 13th Avenue, Eugene, OR 97403-1202; email: yingjiul@uoregon.edu.

0360-0300/2020/09-ART104 $15.00
https://doi.org/10.1145/3409796

1 INTRODUCTION

> "To remember where you come from is part of where you're going."
> — *Anthony Burgess*

Validating network paths is indispensable in constructing a secure Internet architecture [7, 13, 32, 83]. Path validation has been actively propelled by the Path Aware Networking Research Group (PANRG) under the Internet Research Task Force (IRTF) and Internet Engineering Task Force (IETF) [3, 4]. Unlike the current Internet, where routers have full control over packet delivery, path validation empowers end-hosts to enforce the paths they would prefer their packets to follow. End-hosts can also verify whether these paths are truly followed. Clearly, any feasible path validation solution would modify the packet processing logic on routers. This makes path validation difficult to deploy in the current Internet scenario yet necessary for future Internet architectures [13]. Thus, one may wonder: why bother making such a substantive change to the Internet infrastructure? We already enjoy diversified high-quality Internet services. The transparency of underlying packet delivery, however, necessitates path validation. Consider a supposed life-saving call to 911 that is answered a bit late or fails to get through. Should the unfortunate caller attribute this delay to the slow Internet backbone [16] because it failed to process the call more quickly? Or should the caller question whether the call should be directed along a slower path not intended for critical services such as 911? This concern was confirmed a decade ago—Internet service providers (ISPs) sometimes transit each other's traffic for profits [102]. In particular, the ISP you signed up with may direct your traffic to other ISPs with inferior performance. Moreover, in source spoofing attacks [50], the source IP address of a packet is not necessarily the exact IP address of the packet sender. When this situation occurs, received packets may originate from some unknown location rather than the one claimed by the packets' source IP addresses. However, the current Internet provides no means by which users can enforce and verify packet delivery. This deficiency may be exploited to enable security breaches in, for example, financial, medical, and military services built on the Internet [11]. To mitigate this security issue, the Internet community, especially PANRG [3, 4], advocates validating network paths by enforcing the locations that packets should visit and verifying the paths along which packets have transited. With path validation, we expect to unveil the mysteries of packet forwarding that are otherwise hidden from end-hosts and make packet delivery more reliable.

Path validation is not the first approach for tackling the insecurities of the current Internet, but it is the most robust. Instead, the first rally point against an insecure Internet is secure routing [49]. Secure routing aims to find the best (e.g., shortest) paths connecting end-hosts while preventing the path-finding process from being attacked. However, secure routing and path validation are loosely orthogonal yet tightly complementary. Specifically, finding the best path does not necessarily mean that packet delivery must strictly follow that best path. Under the control of attackers, compromised routers may discard or detour packets. To improve forwarding reliability, it is possible to adopt source routing that embeds path directives in packet headers [38]. A path directive specifies a sequence of routers that must be visited, in sequence, by the corresponding packet. To allow checking of the path a packet has transited prior to the destination, Traceroute [74] enables end-hosts to retrieve the path taken by a packet. Given sufficient cryptography augmentation, secure source routing [11] and secure traceroute [65] can enforce and verify network paths, respectively, without being vulnerable to forging and tampering. However, neither solution was designed with the other in mind, and it is inefficient to simply combine them for path validation [50, 62].

This article presents the first comprehensive survey of path validation. Simply put, path validation works by embedding cryptographic states computed over path metadata in packet headers

[62]. Such packet-carried states are initialized by the source and force intermediate routers to forward packets along the source-specified path. Moreover, the states are updated on the intermediate routers, enabling both downstream routers and the final destination to verify path compliance. Clearly, path validation increases packet size and thus imposes bandwidth overhead on a network. This overhead, however, is considered insignificant compared with the plentiful network bandwidth [23]. The benefits in performance, reliability, and security gained from path validation far outweigh the additional bandwidth overhead. Despite the importance of advancing Internet technology, path validation has a surprisingly limited number of solutions. Nevertheless, many Internet enhancement solutions claim reliable end-to-end communication as a goal. Without a deep understanding of each solution, we may easily miss subtle yet essential differences that render a solution unsuitable for path validation, which further motivated us to conduct this survey. When identifying path validation solutions, we adhere to the strict requirement that path validation should jointly achieve path enforcement and path verification [50]. The evolution of path validation design works to improve efficiency while reducing state size. Some sporadic work [97] is dedicated to analyzing security properties using formal security proofs. A variant also validates paths that packets should not transit [52]. Inspired by the disparities in the established efforts for path validation and the attention it should deserve, we also explore future research avenues aimed at enhancing efficiency, security, and privacy.

Conducting such a thorough and systematic survey is challenging, particularly when the effort includes filtering out unqualified solutions and scrutinizing qualified ones. As mentioned earlier, many solutions tout design goals similar to that of path validation, only to fall short. Often, they either refer to secure routing instead of forwarding [49] or partially solve path validation through only one attempt at path enforcement [11] and path verification [65]. Thus, the unqualified solutions far outnumber the qualified ones, making the classification process nebulous and difficult. Even for path validation solutions that jointly enforce and verify network paths, we must scrutinize the subtle differences therein. We find that later solutions gain efficiency only at the cost of security. This, however, is not apparent without a systematic understanding of path validation solutions.

We strive to crystallize the approaches that work for path validation, those that do not, and those that are worth pursuing. Table 1 classifies all of the solutions reviewed in this article. We highlight our major contributions as follows in the hope that they may benefit researchers by providing useful directions for validating network paths and therefore securing the Internet:

- Establish the motivation and requirements for path validation. This knowledge helps readers better understand the significance of path validation.
- Clarify the differences between path validation solutions and various other Internet enhancements, such as secure routing, (secure) source routing, and (secure) traceroute. Only after filtering the unqualified solutions can we concentrate on how path validation works and evolves and highlight its limitations.
- Scrutinize the principles and limitations of path validation solutions. We find that efficiency gains typically come at the expense of security. This subtle caveat can be uncovered only through a systematic understanding of path validation solutions.
- Explore future research avenues and suggest feasible directions. Path validation has many breakthroughs in efficiency, security, and privacy to expect.

The rest of the article is organized as follows. Section 2 provides the motivation for studying the path validation problem and specifies the requirement for a solution. Section 3 covers related works that are concerned with loose control over network traffic but fail to validate their forwarding paths. Section 4 reviews the available path validation solutions. Section 5 focuses on a variant of the

Table 1. Reference Classification

Category	Reference	Index	Principle
Secure routing*	S-BGP [49] SPV [42] Lychev et al. [54]	Section 3.1.1	Cryptographically secure routing messages against prefix hijacking, route spoofing, and eavesdropping
Source routing	i3 [78] DOA [84] NUTSS [37] Dysco [96] NIRA [91] RBF [68] Multipath routing [34, 38, 59, 92, 101]	Section 3.1.2	Embed plain-text path directives in packet headers to direct packet delivery on en-route routers
Traceroute	PPM [74] Efficient PPM [36, 77, 89] DPM [25] Cherrypick [82] SPIE [75] NetSight [40] Hybrid design [28, 55, 56, 90]	Section 3.1.3	Mark router history in packet headers, or log packet history on routers, to retrieve taken paths
Path verification: secure traceroute	Padmanabhan & Simon [65] Wong et al. [86] AudIt [9] RPVM [45] SPP [26]	Section 3.2	Cryptographically secure packet marks or logs against forging and tampering
Path enforcement: secure source routing	Platypus [69] Avramopoulos et al. [11] Ethane [24] SCION [13, 66, 98] ARROW [67] Onion routing [81] Tor [80] HORNET [27]	Section 3.3	Cryptographically secure packet-carried path directives against forging and tampering
Path validation: enforcement + verification	PFRI [23] ICING [62] OPT [50] OSV [21, 22] PPV [87] Alibi routing [52]	Section 4.2 Section 4.3.1 Section 4.3.2 Section 4.3.3 Section 4.3.4 Section 5	Embed cryptographic state in packet headers, state enforces specified paths, routers update state toward verifying path compliance

*We list only typical secure routing solutions here; however, more are reviewed elsewhere [5, 20, 41, 43, 54, 63].

path validation problem called *alibi routing*. Section 6 identifies open issues and future directions. Finally, Section 7 concludes the article.

2 PATH VALIDATION: WHAT AND WHY

In this section, we present the motivation and requirements for validating network paths. The current Internet architecture is focused on the process of finding routes or paths between autonomous systems (ASes) [54]. However, the issues of how to enforce packet forwarding along a specified path and how to verify whether that path was truly taken are less explored. All network entities along a path—the source, the intermediate routers, and the destination—have motivations for enforcing and verifying network paths (Section 2.2). When considering the potential points of failure, which range from external attackers to misbehaving internal network entities, it becomes clear that path enforcement and verification are both essential prerequisites for a robust Internet. The solutions, referred to as path validation [50], are indispensable for a secure Internet architecture.

2.1 Out of Source, Out of Control: Packet Forwarding Under the Internet's Status Quo

In the current Internet architecture, routers determine how a packet should be forwarded toward its destination [29]. The forwarding decision depends on each router's local routing table. Each entry in the routing table associates a reachable destination with a next-hop router along the path to the destination. A routing table entry may also contain fields such as path cost and quality of service that help drive path selection. Routing tables can be configured either manually or dynamically. Manual configuration requires that a network administrator calculate all of the paths for potential packets across the network. For each path, the administrator needs to add a routing table entry to every router along the path. This task is manageable for small and static networks. However, it becomes challenging, tedious, and error prone as networks become larger or more dynamic. Therefore, most routers adopt a dynamic configuration that uses routing protocols to automatically configure routing tables and update them when the network changes. Routing protocols enable routers to propagate the network topology such that any router may determine a local path. Routing protocols for an individual AS are referred to as interior gateway protocols (IGPs), whereas those for communicating among different ASes are referred to as exterior gateway protocols (EGPs) [57]. The mainstream routing protocols currently in use are the open shortest path first (OSPF) protocol [60, 61], the intermediate system to intermediate system (IS-IS) protocol [64] (for IGPs), and the border gateway protocol (BGP) [70] (for EGPs). The software layer that executes the routing protocols in modern router architectures is called the *control plane*. The hardware layer, called the *forwarding plane* or *data plane*, forwards packets at line speeds. Therefore, routers need to compile the routing tables in the control plane into forwarding tables supported in the forwarding plane. The memory that stores forwarding tables should support fast lookup. The types of memory in common use are static random access memory (SRAM), dynamic random access memory (DRAM), and ternary content addressable memory (TCAM). The lookup process usually follows the longest prefix match algorithm. It forwards a packet to the next hop based on a routing table entry that satisfies two conditions: (1) the prefix bits of this entry are also the leading bits of the packet's destination address, and (2) when more than one entry satisfies the first condition, the selected entry is the one with the longest prefix [71]. A routing table may also contain a default rule (i.e., 0.0.0.0/0) to handle packets that match none of the other rules [2].

After a packet leaves its source, the source has no control over how intermediate routers may forward the packet. The destination is equally powerless; neither entity has the ability to enforce path preferences—that is, to specify which path their packets should follow. Moreover, the lack of packet verification on routers makes forwarding processes exploitable for many attacks. For

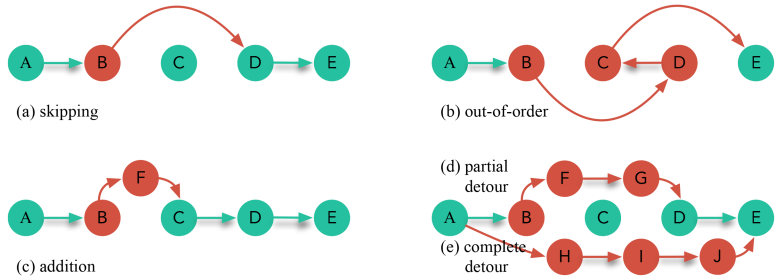

Fig. 1. Example forwarding inconsistencies (i.e., detour attacks) [21, 22, 50, 62, 87] on an example forwarding path, ABCDE. The corresponding descriptions are summarized in Table 2.

Table 2. Possible Forwarding Inconsistencies (i.e., Detour Attacks) from the Expected Path ABCDE in Figure 1 [87].

Forwarding Inconsistency	Description
(a) Skipping	Packet forwarding skips one or more intermediate routers.
	Example: skip C and follow ABDE.
(b) Out-of-order	Packet forwarding visits all routers on the expected path but in a different order.
	Example: follow ABDCE.
(c) Addition	Packet forwarding first detours and then returns to the expected path.
	Packet forwarding thus visits one or more routers that otherwise are not expected.
	Example: follow ABFCDE.
(d) Partial detour	Packet forwarding deviates from some but not all of the expected routers.
	Example: follow ABFGDE.
(e) Complete detour	Packet forwarding deviates entirely from the expected path.
	Example: follow AHIJE.

example, because routers do not authenticate packet sources, IP spoofing-based distributed denial of service (DDoS) attacks forge the source addresses of attacking packets, making them difficult to trace [74]. Compromised routers can also arbitrarily tamper with packet payloads without detection by downstream routers. Figure 1 shows an example of forwarding a packet with an expected path of ABCDE. How the forwarding of this packet might be inconsistent with the expected path is summarized in Table 2 [87].

Although source authenticity and packet integrity can be verified based on end-to-end cryptographic authentication, network paths taken by packets cannot easily be verified or enforced in the current Internet scheme. Verifying paths using either per-hop, per-packet receipts to the source (e.g., traceroute [75]) or per-hop, per-packet logs for post hoc queries [26] incurs impractically heavy communication or storage overhead. Enforcing paths is even harder: in the current Internet, packets carry only an indication of where they should go (i.e., the destination addresses in packet headers) but do not include any packet-specific preferences for intermediate routers. One might suggest that before sending packets, the source or destination should inform each intermediate router of its preferred path, with packet identifiers and preferred forwarding decisions. Then, upon receiving a packet, each router could locally match it against the recorded packet identifiers and, when a match is found, follow the associated forwarding decision. Again, however, this

solution involves excessive communication and storage overhead to network channels and equipment. In addition to efficiency concerns, the larger challenge occurs when compromised routers misbehave [50, 62].

2.2 My Packet, My Way: Why You Can't Count on the Current Internet

Before discussing the requirements and challenges for enforcing and verifying network paths, we first explore their use cases. Certain demands must be met for all three types of entities along a network path—that is, the source/sender, intermediate/en-route routers, and the destination/receiver.

Source's demands. The source specifies path preferences mainly to guarantee a certain quality of service. The expected service quality is usually specified in a service level agreement (SLA) negotiated with the service provider. For example, ISPs provide customers with multiple choices of network bandwidth, such as 100 Mbps or 1 Gbps. Similarly, mobile carriers offer different wireless bandwidths to customers through different cellular technologies, such as 3G and 4G. However, a misbehaving ISP/carrier may forward traffic via an inferior path rather than the premium one promised in the SLA [45, 50]. Moreover, different ISPs may sometimes transit traffic between each other [102]. In either case, a customer may experience degraded service quality due to path noncompliance.

The source may also specify path preferences for additional security and privacy. It is common for certain regions to deploy censorship systems to inspect and filter Internet traffic. When the source in a region sends packets to the destination in another region, the packets may transit a series of intermediate regions. When multiple feasible paths transit different sequences of intermediate regions, the source likely wishes to choose one without censorship. This situation is more challenging, because we need to be able to prove not only that a packet takes the chosen path but also that it does not take an unwanted path involving censorship [52]. A proof of taking one path is not necessarily equivalent to a proof of not taking any other path. For example, misbehaving routers could simply process packets without adding the required proofs to them [62].

Destination's demands. The receiver specifies path preferences for incoming packets mainly to enable service chaining [62]. Specifically, the receiver may require that incoming packets pass through services such as accounting, inspection, and load balancing. Different service chaining operations may direct packets through different paths. This requires adapting the Internet to multipath routing [48], which provides two end-hosts with multiple path choices rather than a single best path (e.g., the shortest path [76]).

Intermediate router's demands. It would seem that intermediate routers have less incentive to accommodate path preferences of end-hosts. However, the ability of intermediate routers to satisfy path preferences has a direct impact on the end-host's perceived service quality. An unsatisfactory service quality will discourage end-hosts from continuing their subscriptions and therefore reduce profits for the service provider. When intermediate routers cooperate with end-hosts, they should collaboratively detect deviated packets as early as possible [50]. Tolerating deviated packets may waste network resources, impair service quality, expose the privacy of the source, and jeopardize the security of the destination.

2.3 Path Validation = Path Enforcement + Path Verification

The goal of path validation is to enforce and verify network paths. Kim et al. [50] defined the goal of path validation as follows:

> The source, intermediate routers, and the destination should be able to validate that the packet indeed traversed the path known to (or selected by) the source.

> Successful path validation ensures that the packet traversed each honest router on the path in the correct order.

To achieve this goal, a path validation solution should feature two functionalities. First, it should enable the source and intermediate routers to enforce mutually agreed upon path preferences. Second, it should allow both destination and intermediate routers to verify path compliance. The design principle is to provide two packet-path bindings. One binding is between a packet and the path it should take. This enables path enforcement because the binding specifies the forwarding location for the packet. The other binding is between the packet and the path it actually takes. This enables path verification because the binding verifies whether the taken path exactly matches the specified path. If the verification passes, path validation succeeds.

Path validation solutions rely on packet-carried states. The fundamental premise behind packet-carried states is that bandwidth is plentiful [62]. However, it is still important to limit the state size to minimize bandwidth consumption. Specifically, in path validation, the state is initialized by the source and included in expanded packet headers. The state should include sufficient information regarding specified paths such that intermediate routers can verify whether they pertain to a correct path and determine where to correctly forward a packet. An intermediate router also needs to verify whether a packet transited upstream routers in the correct sequence. To this end, each intermediate router should update the state to provide proof to all downstream routers and the packet's destination. The source, intermediate routers, and destination should all use mutually exchanged shared secrets (e.g., keys) to update and verify the state.

Various design challenges arise from the requirement that path validation should function properly in an adversarial, decentralized, and high-speed environment [62]. Path validation may be employed by all three types of network entities (i.e., source, intermediate routers, and destination) to improve service quality, security, and privacy. It is certainly wise to presume the existence of external attackers or even internal misbehaving entities. Such adversaries aim to enable packets that have not taken the specified path to still pass path verification. Path verification should be decentralized so that intermediate routers and the destination can locally verify path compliance. By not transiting packets and their paths' metadata to a centralized server, decentralized path verification avoids the corresponding transit delay. Furthermore, it avoids a single point of failure that a centralized server represents. However, limiting the delay induced by processing the packet state beyond high-speed forwarding is still a major challenge. For Internet companies, a few milliseconds of delay can result in profit losses of millions of dollars [16].

Clearly, routers need to be upgraded to support path validation that supports packet-carried state and addresses the preceding challenges. This upgrade involves both hardware (e.g., for cryptographic computations) and software (e.g., for processing state), which is why path validation solutions still have not been deployed and remain transcendental for future secure Internet architectures [13]. One question we should ask is whether we truly need to change router architectures to enable path validation. If the answer is yes, then to what extent do we need to change the current Internet? Path validation researchers, of course, are not the first to concern themselves with the problem of uncontrollable forwarding. Thus, the following questions arise. How have other peer researchers tackled this problem? Is it possible to achieve path validation without modifying Internet protocols? Before discussing path validation solutions in Section 4, we first review various related solutions in Section 3 and analyze their shortcomings in validating network paths.

3 THE ROAD TO PATH VALIDATION

In this section, we exclude seemingly qualified solutions and examine their shortcomings regarding path validation. Although most of these solutions claim to achieve goals similar to that of path

validation, they cannot achieve both path enforcement and path verification, which is the key to path validation. The unqualified solutions far outnumber the qualified ones (to be reviewed in Section 4), making the classification process nebulous and challenging. This, however, further inspired us to conduct this thorough and systematic survey of the methods that support path validation, those that do not, and which are worth pursuing:

- We start with secure routing, which provides a secure process for finding feasible paths between end-hosts (Section 3.1.1). Secure routing is easily confused with path validation, because its solutions usually claim secure end-to-end communication as the goal. However, routing plays only a partial role in the communication process: it is the forwarding process that determines whether packets are delivered in accordance with routing policies. Forwarding decisions are congruent with routing policies only when no faulty or malicious routers exist.
- To gain more control over the forwarding process, non-security-based solutions such as source routing (Section 3.1.2) and traceroute (Section 3.1.3) augment packet delivery with operations for enforcing and verifying paths. Specifically, source routing adds a path directive to packet headers to inform intermediate routers where to forward packets, whereas traceroute requires routers to log packet histories or mark packet headers with visit histories. Such historical information enables end-hosts to reconstruct the paths taken by packets and evaluate compliance with forwarding requirements. However, these solutions focus more on effectiveness than on security. Neither path directives nor packet/router histories are cryptographically secured, which renders them vulnerable to forging and tampering.
- Cryptographic enhancements have been introduced to enable secure traceroute (Section 3.2) and secure source routing (Section 3.3), respectively. Secure traceroute achieves path verification, whereas secure source routing achieves path enforcement. However, these operations are not designed to operate concurrently. In other words, it is infeasible or inefficient to simply integrate them to enable path validation.

3.1 What's (Definitely) Not Path Validation? Secure Routing, Source Routing, and Traceroute

We start with non-security-based solutions that (1) secure routing instead of forwarding (Section 3.1.1), (2) improve forwarding flexibility and robustness instead of security (Section 3.1.2), and (3) support forwarding diagnoses via tamperable traceroute (Section 3.1.3).

3.1.1 Secure Routing. Secure routing protocols aim to protect the path-finding process against various attacks, such as prefix hijacking, route spoofing, and eavesdropping [20, 41, 42, 49, 54, 73]. In other words, the ultimate goal of secure routing is to guarantee that the computed route is the best and most authentic choice generated by the routing protocol in use [54]. This may elicit an illusion that secure routing offers secure end-to-end communication. However, routing refers to only the process of finding routes; whether the packets are actually forwarded along the selected routes is beyond the scope of routing. For more specifics, we refer interested readers to comprehensive surveys on secure routing [5, 20, 41, 43, 54, 63]. Secure routing protocols alone are not sufficient. Using a compromised router, an attacker can still tamper with or arbitrarily forward packets. Therefore, securing the forwarding process (e.g., by path validation) is crucial to ensure secure communication [85].

3.1.2 Source Routing. Source routing embeds information regarding source-selected paths (i.e., path directives) in packet headers to direct routing decisions on intermediate routers [34, 37, 38, 59, 68, 78, 84, 91, 92, 96, 101]. This idea was originally adopted by the IP but was not widely used

[8]. More recently, source routing has attracted increasing attention from researchers as control over packet delivery is considered increasingly necessary [8, 38, 85, 91]. Source routing solutions focus mainly on robustness against link failures and flexibility for policy enforcement. However, the current solutions do not consider security as a design goal. In other words, source routing assumes that routers are benign and that they faithfully follow path directives to forward packets. Consequently, path directives appear in plaintext. Once routers are compromised, they may violate source routing (e.g., by tampering path directives) without detection. Path validation, however, considers compromised routers as a possible threat to forwarding and takes precautions against them.

3.1.3 Traceroute. The key idea behind traceroute is to enable routers to track traffic history, either by directly marking packet headers [74] or by logging packet metadata locally [75]. Then, the destination/receiver can use the in-packet router marks or acquire packet metadata by sequentially querying previous-hop routers to determine the paths taken by received packets. Packet logging and packet marking are jointly adopted in some hybrid designs [28, 55, 56, 90] to reap the benefits of both worlds. Traceroute was originally proposed to detect IP spoofing and related DDoS attacks [75, 95]. Follow-up work [17, 36, 40, 77, 89] strove to improve efficiency, particularly to reduce the number of marked packets. At the other extreme, some solutions require each router to mark every packet [25, 82]. Unfortunately, we cannot simply employ traceroute to assist in path validation. First, traceroute does not regulate which path a packet should take. Therefore, it does embed path directives in packet headers, which fails the requirement of path enforcement. Second, neither the in-packet marks nor the on-router logs are cryptographically secured, making them vulnerable to forging and tampering—particularly if some routers are compromised. Path validation, however, effectively combats the impact of compromised routers on packet forwarding.

3.2 What's One Step Closer? Path Verification

Path verification enables the destination to retrieve paths taken by packets and verify their authenticity [26, 65]. The essential idea behind path verification is *secure traceroute*. In contrast to conventional traceroute solutions (Section 3.1.3), secure traceroute uses cryptography to secure the metadata used to retrieve packet trajectories. If the actual paths taken by packets deviate from the expected paths, path verification may also help detect the faulty or compromised routers on which path deviation occurs. A typical secure traceroute operation requires that the source probe the en-route routers hop by hop, prompting each router to respond with the address of its next hop [65, 86]. Another line of research achieves coarser-grained path verification by requiring routers to provide feedback consisting of aggregate metrics over a set of packets [9]. To attempt to create unforgeable packet-level history, per packet cryptographically secured history can be maintained on each router [26, 39, 100]. Path verification, however, does not mean that all of the solutions for detecting faulty or compromised routers (e.g., [12, 35, 45]) are capable of path verification. Because path verification solutions do not enforce paths, they fail to achieve path validation.

3.3 What's Almost There? Path Enforcement

We end this section with path enforcement solutions that further pave the road to path validation. Path enforcement is built on top of *secure source routing*, which differs from secure routing (Section 3.1.1) in that the former protects the forwarding process, whereas the latter protects the routing process. Moreover, it differs from source routing (Section 3.1.2) in that it focuses more on security and makes in-packet path directives harder to forge or tamper with. Specifically, path enforcement seeks to prevent forwarding policies specified by in-packet path directives from being circumvented [24, 27, 67, 69, 80, 81]. Path enforcement requires cryptography to secure the path

directives carried in packet headers [11]; however, it does not require en-route routers to embed verifiable proofs in packet headers. Upon receiving a packet, downstream routers and the destination have no means to verify the actual path taken by the packet. This lack of path verification, however, renders path enforcement alone insufficient for path validation.

4 FINALLY, PATH VALIDATION = PATH ENFORCEMENT + PATH VERIFICATION

In this section, we review path validation protocols that not only enforce source/destination-chosen paths but also verify whether the specified paths were truly taken by packets. To our surprise, after excluding the inapplicable solutions listed in Section 3, only a limited number of path validation solutions exist. These are built on the fundamental premise of plentiful network bandwidth—that incorporating the required mechanisms in packet headers is more important than optimizing bandwidth consumption [23]. Their primary goal is to jointly guarantee path enforcement and verification by embedding path proofs in packet headers. Moreover, the embedded proofs should be difficult to forge or tamper with. Their size can be rather large, on the order of hundreds of bytes. Nevertheless, path validation is an indispensable building block for future secure Internet architectures. Our survey helps clarify the necessity for path validation, provides researchers with a comprehensive understanding of the current solutions, and explores future research directions.

We summarize our major understandings and findings of existing path validation solutions as follows:

- Since the first framework for path validation was proposed [23] (Section 4.2), subsequent solutions have striven to reduce the data size needed for embedding validation information and accelerate the computation over such data [21, 22, 50, 62, 87] (Sections 4.3.1–4.3.4). However, these efficiency enhancements have come at the expense of security. For example, for services that are hardly affected by a single or a limited number of attacking packets, path validation can be probabilistically enforced over individual packets while collectively guaranteeing per-flow forwarding correctness [87].
- Although the design framework for path validation is set, the computational approaches form a promising design space. For example, fast algebraic computation can sometimes be equally effective and secure yet more efficient than cryptographic computation [21, 22] (Section 4.3.3).
- Formal security proofs for path validation protocols have not matched the pace of protocol evolution. To date, only the protocols presented by Kim et al. [50] have been validated by formal methods [97]. Because path validation is indispensable for future secure Internet architectures, formal security proofs are critical before deploying any solutions (Section 6.2).
- Current path validation protocols still suffer from various limitations in various aspects, such as security, privacy, efficiency, and deployability. In this section, we focus mainly on their principles, and in Section 6, we discuss their limitations that will drive future research directions.

4.1 Taxonomy of Design Space

Before delving into the designs of existing path validation solutions, we present a high-level overview of the common framework they use to achieve validation and the design space for exploring efficiency. We review their concrete design principles in Section 4.2 and Section 4.3 followed by an analytical comparison in Section 4.4. We also discuss the interplay between security and efficiency across the design space in Section 4.5.

4.1.1 Framework. Path validation solutions add path proofs to packet headers to enable verification of whether a packet has traversed intermediate routers in the correct order. Such proofs

should be updated by routers in hop-by-hop fashion and should be difficult to forge. The common framework for path validation consists of the following steps:

- *Path dissemination*: Because the ultimate goal of path validation is to enforce packet forwarding along a specified path, the path must first be known to en-route routers. The desired path can either be communicated directly to en-route routers or exist as piggybacked packet headers.
- *Key exchange*: An en-route router adds its cryptographic proof to the packet header for other downstream routers to verify. This requires key exchange and sharing among routers.
- *Proof initialization*: The source initializes in-packet proofs before sending them out. The proof is usually a keyed message authentication code (MAC) of packetwise metadata, such as its payload hash and session identifier (ID).
- *Proof verification*: Upon receiving a packet, the router first verifies the in-packet proof. If the verification fails, the router discards the packet.
- *Proof update*: When the in-packet proof verification succeeds, the router continues by updating the proof with its own. It then forwards the packet to the next hop, which repeats the processes of proof verification and proof update. The iteration ceases after the packet arrives at its destination, which verifies the proof and accepts the packet if verification succeeds.

4.1.2 Design Space. The design space of path validation solutions spans different levels of trust and efficiency. The lower the level of trust shared among en-route routers is, the more proof is required for path validation, which guarantees reduced efficiency. In particular, the level of trust and the validation granularity can be classified as follows:

- *Pairwise distrust versus trusted source/destination*: When routers (including the source and destination) do not trust each other, each router requires proof from all other routers. In other words, a router must be able to provide proof to all other routers. This requires the greatest amount of computation at routers and thus leads to the lowest efficiency. When the pairwise distrust assumption is relaxed, the trusted party can shoulder a certain amount of computational overhead from other routers to improve the overall efficiency.
- *Per-packet validation versus per-flow validation*: Per-packet validation requires a packet to be validated by all en-route routers. Thus, every packet accepted by the destination should have traversed the path in the correct order. This finest level of granularity ensures that the packet reached its destination with the highest trust level of forwarding correctness. However, this approach also requires the most computation. Recent work improves the efficiency by using each packet to probabilistically validate a path segment and using co-flow packets to jointly validate and enforce the entire path.

4.2 Prelude and Rationale: PFRI

Postmodern forwarding and routing infrastructure (PFRI) [23] was the first work to advocate that enforcement and verification be jointly performed on network paths. It centers on a datagram delivery service that pertains to the larger clean-slate postmodern Internet architecture (PIA) [14]. As shown in Figure 2, PFRI models a network as a collection of *realms*. A realm consists of nodes (i.e., end-hosts and forwarding elements) and channels (i.e., links between nodes). A sequence of channels from the source to destination constitutes a PFRI path. Successful forwarding along a PFRI path requires a certain level of centralization [62] to maintain, for example, mappings from end-host identifiers and connection channels to realms. For path enforcement, the motivation field in the packet header carries a network path as a forwarding directive. For example, in Figure 2, node α in realm R_1 has a packet destined for end-channel d. Node α first queries its co-realm centralized

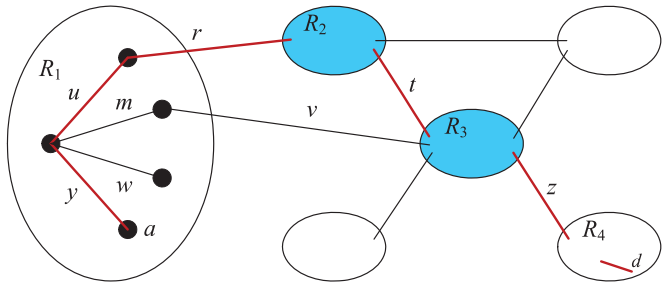

Fig. 2. PFRI instance created by rendering Figure 3 in Calvert et al. [23]. End-host α in realm R_1 sends a packet to end-channel d in realm R_4; realms R_2 and R_3 provide transit services. The initial partial path is $\alpha yurtzd$. The partial paths connecting rt, tz, and zd are locally determined in realms R_2, R_3, and R_4, respectively.

service for the border channel (i.e., z) of the realm (i.e., R_4) containing d. Then, α queries a higher-level centralized service for the transit realms (i.e., R_2 and R_3) and the corresponding inter-realm path (i.e., rtz). Using local knowledge of intra-realm topology, α selects an interior path (i.e., yu) leading to the border channel r that connects to the first transit realm R_2. Thus far, the initial partial path is constructed as $\alpha yurtzd$; it is partial in that the other partial paths connecting rt, tz, and zd will be locally determined in realms R_2, R_3, and R_4, respectively. Along with the forwarding directive, the motivation field should also carry credentials to convince the en-route nodes to forward the packet [23]. To verify that the path truly is forwarded along the path specified in the motivation field, each en-route node should provide hard-to-forge evidence of packet forwarding in the accountability field in the packet header [23].

PFRI, however, does not elaborate on how to configure the fields introduced to the packet headers to enable path validation [23]. For example, how should the source compute credentials for the en-route nodes? How should the en-route nodes compute signatures for the destination? How do all of the nodes on the path share keys? All of these questions introduce practical concerns and design challenges, especially when mingled with the problem of how to control packet header expansion [62]. This is probably why only ICING identifies PFRI [62] as a predecessor. Most of the related works refer to ICING as the first path validation protocol.

4.3 Concrete Design: ICING, OPT, OSV, and PPV

Led by ICING [62], a series of other path validation protocols—OPT [50], OSV [21, 22], and PPV [87]—have also been proposed. All of these protocols provide concrete design strategies with unforgeability as a security goal and efficiency as a performance goal. In particular, a packet header architecture different from the conventional one must first be instantiated. This architecture introduces specific header fields to enable path validation. Then, the corresponding path validation protocol provides the configuration and update details for these fields, as well as how to use them to compute a path validation. In this section, we review the key ideas of the existing path validation protocols [21, 22, 50, 62, 87] and compare them in terms of various metrics in the sections. We refer interested readers to our technical report [19] for synopses of these designs and to the original works [21, 22, 50, 62, 87] for more details.

4.3.1 ICING. Pertaining to a future Internet architecture called *NEBULA* [7], ICING [62] is acknowledged by most studies [21, 22, 50, 87] as the first concrete path validation design. It reinforces the requirements for designing path validation protocols—that is, a path validation protocol should function properly in an adversarial, decentralized, and high-speed environment (Section 2.3). Two key design choices that make ICING efficient are aggregate MACs [47] and self-certifying names

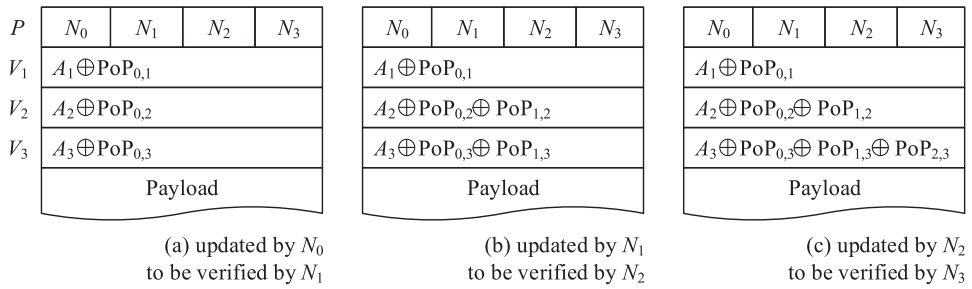

Fig. 3. ICING packet instance created by rendering Figure 2 in Naous et al. [62].

[6, 58]. First, aggregate MACs reduce the state length from quadratic to linear with respect to path length. Second, ICING configures shared keys for each pair of nodes to compute/verify proofs directly using self-certifying node names [6, 58], without requiring a central naming authority or PKI.

Figure 3 illustrates how ICING enforces path validation over a packet along path $P = (N_0, N_1, N_2, N_3)$. Here, N_0 and N_3 are the sender and receiver, respectively, whereas N_1 and N_2 are intermediate routers. The framework of ICING can be outlined as follows:

- *Path dissemination*: Prior to sending packets along path P, the sender (N_0) must first request consent from all of the other nodes. ICING refers to such consent as proof of consent (PoC). N_i ($1 \le i \le 3$) computes the PoC_i using the node list in P and its private key. After obtaining all of the PoCs, N_0 initializes the packet as follows. First, it adds the list of nodes (P) to the packet header.
- *Proof initialization*: To prove to $N_i \in P$ ($1 \le i \le 3$) that all of the nodes along path P have given consent, N_0 should also include the obtained PoCs in the packet header. To prevent an attacker from eavesdropping on the PoCs, N_0 further obfuscates them using the packet content. The obfuscation results are called *authenticators* A_i ($1 \le i \le 3$), and each occupies a verifier field V_i that N_i can use for verification purposes. Further, N_0 should provide N_i with the ability to verify its PoP_i. As illustrated in Figure 3(a), PoP_i is XORed to V_i, which is currently equal to A_i, using the aggregate MAC technique. N_0 then sends the packet to N_1.
- *Proof verification*: N_1 needs only to verify V_1. The verification succeeds if two conditions are satisfied. First, N_1 issues a PoC of path P to N_0. Second, N_0 adds in the verifier V_1 a PoP of the packet. To verify these two conditions, N_1 first computes the PoC and then the corresponding authenticator A_1; then, it computes the PoP using the key shared with N_0. After obtaining the two values, N_1 further compares their XOR result with the value of V_1. If they are equal, the packet passes verification and is then updated by N_1 with its PoPs for N_2 and N_3 XORed to V_2 and V_3, respectively (Figure 3(b)). N_2 and N_3 similarly verify and update the packet. Only when the packet has visited every node on path P will it pass verification at every node.

4.3.2 OPT. Origin and path trace (OPT) [50], integrated in the SCION Internet architecture [13, 98], achieves higher efficiency than does ICING, because it requires a higher level of trust among the on-path nodes. Most of the computational overhead can be offloaded to the trusted nodes to improve efficiency. Again, consider $P = (N_0, N_1, \ldots, N_i, \ldots, N_{n-1})$ as the path to be enforced. Figure 4 illustrates how OPT augments the packet header for path validation [50].

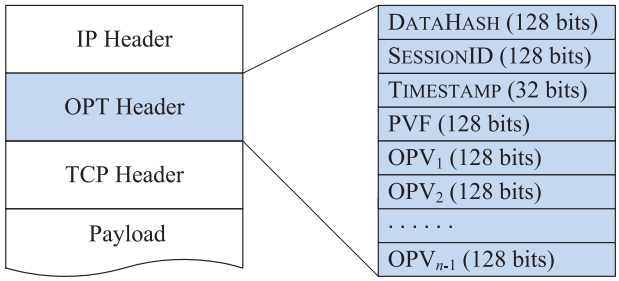

Fig. 4. OPT packet instance created by rendering Figure 4 in Kim et al. [50]. Different from ICING, OPT does not include the list of on-path nodes in a packet header. Instead, the source forwards the path information to on-path nodes when they exchange keys. The source N_0 precomputes PVF and OPV_i for $1 \le i \le n-1$, and N_i validates the packet by verifying whether OPV_i can be derived from PVF_i. If validation succeeds, N_i needs to update only the PVF with its proof of integrated processing.

- *Proof initialization*: Because OPT assumes that all nodes N_i $(1 \le i \le n-1)$ trust the source N_0, each N_i generates a shared symmetric key k_i with N_0. Note that routers do not store these keys locally; instead, they regenerate them on the fly when validating packets. The computational operations to generate keys are faster than the operations required to retrieve them from cache or memory on routers [50]. N_0 initializes a path validation field (PVF) as PVF_0 in the packet header. When each downstream N_i updates PVF_{i-1} with a k_i-keyed MAC, the source N_0 can be aware of all correct PVF_i's if the packet follows path P. Then, for N_i to verify whether its received packet has visited N_0 through N_{i-1} in the correct order, N_0 can compute a shared secret with N_i using the correct PVF_{i-1} and k_i. OPT refers to secrets as origin and path validation (OPV); it allocates one OPV_i field for each N_i in the packet header.
- *Proof verification*: Upon receiving a packet, N_i employs the same method as N_0 to compute OPV'_i using the PVF_{i-1} carried in the packet and k_i. Path validation succeeds if $\text{OPV}_i = \text{OPV}'_i$, costing N_i to have a complexity of only $O(1)$ per MAC computation.

Extended-OPT is a variant of OPT that further suggests how to maintain the $O(1)$ MAC computational complexity when nodes do not trust the source.

- *Attack by untrusted source*: A distrusted source may collude with router N_i to bypass all of the routers in between. Consider the situation, for example, in which the source N_0 colludes with router N_2 to bypass N_1. Without this bypass misbehavior being detected by N_i $(3 \le i \le n-1)$, N_0 and N_2 need only to make PVF look as it had been updated by the bypassed N_1. This is easy for N_0 because the expected PVF for N_1 is $\text{PVF}_1 = MAC_{k_1}(\text{PVF}_0)$ and both inputs of k_1 and PVF_0 are known to N_0.
- *Countermeasure by Extended-OPT*: The countermeasure against a distrusted source by OPT introduces another PVF^D for destination N_{n-1} to verify [50]. To support PVF^D verification, each router N_i should create a shared key k_i^D with the destination. N_i then uses k_i^D to update PVF^D. Because k_i^D is unknown to the source N_0, N_0 cannot easily forge PVF^D. Under the Extended-OPT, each on-path node performs two MAC computations over two PVFs.

Retroactive-OPT, another variant of OPT, eliminates the key setup process. Routers still update packet-carried proofs. The keys used for proof update should be revealed on demand by the source or destination. Retroactive OPT requires the destination to buffer received packets until it either determines that they will not be validated or they have been validated. Different from

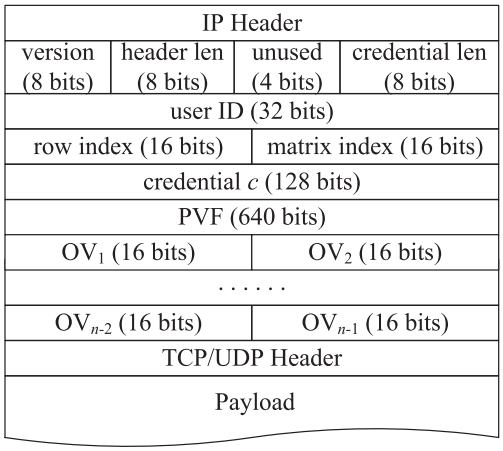

Fig. 5. OSV packet instance created by rendering Figure 4 in Cai and Wolf [22]. The path validation framework is similar to that of OPT. However, OSV adopts orthogonal sequences to achieve faster computation of the PVF and OV fields.

OPT, Extended-OPT, and other solutions, Retroactive-OPT does not support real-time validation of packets immediately after their arrival at routers. Thus, we omit discussing Retroactive-OPT in what follows.

4.3.3 OSV. Orthogonal sequence verification (OSV) [21, 22] follows the same design principle as OPT [50] but leverages orthogonal sequences instead of cryptographic computations to enable faster path validation. As shown in Figure 5, OSV also relies on a PVF and an original validation (OV) field per on-path node. Each node N_i needs to update the PVF in a way known only to itself and the source. The source can easily precompute the expected PVF_i after a series of updates by N_0 through N_i. N_{i+1} can thus use PVF_i as proof of whether its received packet has visited N_0 through N_i in the correct sequence. To accomplish this, the source precomputes OV_{i+1} using a shared function with N_{i+1} and takes PVF_i as an input. Upon receiving the packet, N_{i+1} first computes an OV'_{i+1} using the PVF (i.e., PVF_i) carried in the packet header. If OV'_{i+1} is equal to OV_{i+1} carried in the packet header, path validation succeeds.

The major difference between OSV and OPT is that OSV adopts a considerably faster way to compute PVF and OVs based on orthogonal sequences. Specifically, the source first generates an $m \times m$ Hadamard matrix H that satisfies $HH^T = mI_m$ [10]. All of the distinct rows (or columns) of H contain elements with values of either 1 or -1 and that are mutually orthogonal in that the result of the inner product of any two rows (or columns) is $h_i \cdot h_j = 0$. This motivates OSV to use vectors of H as the credentials for on-path nodes. Consider, for example, when the source N_0 chooses h_0 as its credential and forwards h_1 to N_1 as N_1's credential. Then, N_0 can simply add h_0 to the packet header. Upon verification, N_1 performs an inner-product computation over the carried h_0 and its local h_1. If the result is 0, N_1 can choose to believe that the packet comes from the correct source.

4.3.4 PPV. Motivated by the observation that not all Internet services require strong per-packet path validation [99], PPV [87] instead proposes probabilistic path validation. PPV no longer requires that each packet be marked by all of the routers it visits. Instead, PPV enables routers to mark packets at a certain probability. Although any individual packet cannot be used to validate the forwarding behaviors of all routers, packets in the same flow (e.g., packets of headers with the same five-tuple) can collectively be used to validate whether all routers on the packets' forwarding

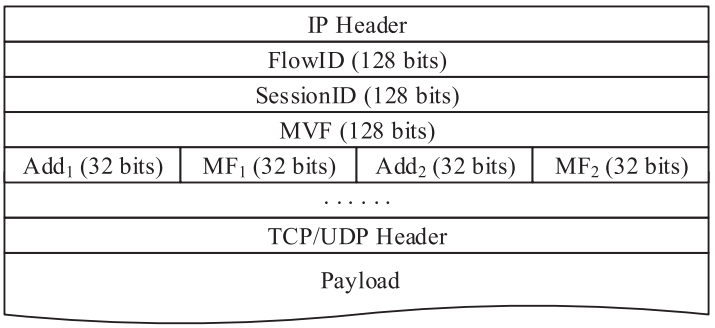

Fig. 6. PPV packet instance created by rendering Figure 3 in Wu et al. [87]. In contrast to validation solutions that require each router to mark every packet with a proof (e.g., ICING, OPT, and OSV), PPV requires routers to probabilistically mark packets. Each packet is marked by at most two routers along the forwarding path. PPV thus does not enforce strong per-packet path validation. Instead, it collectively uses router proofs in packets in the same flow (e.g., packets of headers with the same five-tuple) to achieve per-flow path validation.

paths conform to the forwarding policy. Figure 6 illustrates a PPV packet instance. Following the path validation design principle, a PPV header is located between the IP and TCP headers. Because PPV enforces per-flow path validation, it introduces a FlowID field in the packet header to classify packets. The marking verification field (MVF) field functions as the PVF in an OPT header. Similarly, the marking fields—MF_1 and MF_2—work as the OPV fields in an OPT header. Given these two MF fields, PPV requires only that each packet be marked by two adjacent routers. Their addresses are marked in the two address fields: Add_1 and Add_2. When a router decides to mark a packet, it fills the Add_1 field with its address. Then, its next-hop router should also mark the packet. Upon receiving the marked packet, the destination simply follows the MVF construction and computes an MVF' value. If the recomputed MVF' is equal to the MVF carried in the received packet, the destination accepts the received packet; otherwise, the packet may deviate from its original content or have been modified, and it is dropped.

4.4 Analytical Comparison

Table 3 compares the surveyed concrete designs for path validation in terms of various design and performance properties.

4.4.1 Decisive Factors for Efficiency. As discussed in Section 4.3, the key driver of path validation evolution is efficiency. This is because path validation imposes additional packet processing logic on end-hosts and routers. The more efficient the path validation protocol is, the less intrusive it is to the Internet architecture and the more likely it is to be deployable. At a high level, we observe that two factors determine how path validation can gain efficiency. One is the granularity of the validation process, and the other is the level of trust.

- The granularity of validation determines the amount of traffic that must be validated. As shown in Table 3, most path validation protocols—ICING [62], OPT [50], and OSV [21, 22]—provide per-packet validation in a deterministic way. This full-force validation approach forces every packet undergo additional validation operations on every router it visits, which in turn imposes packet transmission delay and introduces the highest extent of processing overhead to routers. Conversely, a lower validation granularity enables higher efficiency. To this end, PPV [87] does not enforce strict per-packet path validation. Instead, it requires

Table 3. Comparison of Path Validation Solutions Given an n-Hop Forwarding Path

Property	ICING [62]	OPT [50]	Extended-OPT [50]	OSV [21, 22]	PPV [87]
Validation enforcement[1]	Deterministic	Deterministic	Deterministic	Deterministic	Probabilistic
Validation granularity	Per packet	Per packet	Per packet	Per packet	Per flow
Trusted source	Unnecessary	Assumed	Unnecessary	Assumed	Unnecessary
Key exchange	Pairwise	Src vs others	Src&dest vs others	Src vs others	Dest vs others
Overall key count	$O(n^2)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$
Source key count	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(1)$
Router key count	$O(n)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$
Destination key count	$O(n)$	$O(1)$	$O(n)$	$O(1)$	$O(n)$
Source computation	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(1)$
Router computation	$O(n)$	$O(1)$	$O(1)$	$O(1)$	$O(1)^2$
Destination computation	$O(n)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$
Average computation	$O(n)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$
Proof length	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(1)$
Verifier length[3]	42 bytes	16 bytes	16 bytes	2 bytes	4 bytes
Source storage	$O(n)$ keys	$O(n)$ keys	$O(n)$ keys	$O(n)$ keys	$O(1)$ keys
	$O(n)$ path	$O(n)$ path	$O(n)$ path	$O(n)$ path	$O(n)$ path
Router storage	$O(n)$ keys	$O(1)$ keys	$O(1)$ keys	$O(1)$ keys	$O(1)$ keys
Destination storage	$O(n)$ keys	$O(1)$ keys	$O(n)$ keys	$O(1)$ keys	$O(n)$ keys

Notes:

1. Deterministic validation requires that each intermediate router add its proof to packets.
   Probabilistic validation requires that each router add its proof to a packet at a certain probability.
2. Only routers that mark packets suffer the $O(1)$ computation overhead.
3. ICING: V field (Figure 3); (Extended-) OPT: OPV field (Figure 4); OSV: OV field (Figure 5); PPV: MF field (Figure 6).

only two adjacent routers to validate a packet in a probabilistic way. All other routers on the forwarding path simply follow the conventional matching and forwarding logics to process the packet. Obviously, PPV requires much less overhead; however, PPV is unable to detect misforwardings from routers that are not selected to validate a packet.

- The trust level determines the number of entities to trust. When fewer trusted entities exist, more overhead is introduced as protection against the impact of untrusted entities on packet forwarding. Among all existing solutions, ICING has the lowest trust level. Specifically, it considers every entity—source, router, and destination—as potentially malicious. Therefore, it requires each entity to prove itself to all other entities. ICING thus enforces pairwise shared keys [93] among all entities. Each entity maintains $O(n)$ keys, and all together the network must maintain $O(n^2)$ keys. Solutions other than ICING assume either a trusted source or a trusted destination. All other entities maintain keys shared only with the trusted server/destination. In other words, the trusted source/destination maintains $O(n)$ keys, whereas each of the other entities must maintain only $O(1)$ keys. Using the shared keys, the trusted source can then precompute the expected path proofs for each hop. This serves to offload the computational overhead from intermediate routers to the source. Solutions assuming a trusted source [21, 50, 92] can thus achieve higher efficiency than ICING. Similarly, for solutions that assume a trusted destination (e.g., Extended-OPT and PPV), the destination also needs to maintain $O(n)$ keys, each of which corresponds to one of the other $n-1$ entities on the n-hop path.

4.4.2 Computation Overhead. In the preceding schemes, the amount of computational overhead varies on the source, intermediate routers, and the destination. The source computing mainly involves initializing path proofs. The intermediate routers perform extra computing operations for two purposes. One is to validate the path proofs in received packets from the previous hop. The other is to update path proofs for the next hop to validate. Finally, the destination validates path proofs to verify whether the received packets were correctly forwarded.

- *ICING* [62]: Because ICING assumes no trusted entities on the path, each entity except the destination needs to prove itself to all others, inducing a complexity of $O(n)$. Meanwhile, each entity except the source needs to verify the proofs from all of its upstream nodes, yielding another $O(n)$ computation. This causes the computation overhead for the source, all intermediate routers, and the destination to have a complexity of $O(n)$.
- *OPT* [50], *Extended-OPT* [50], *and OSV* [21, 22]: Both OPT and OSV assume a trusted source that precomputes path proofs for other entities. Specifically, for each entity, the source computes an aggregate proof over all proofs of the entity's upstream nodes. Given an n-hop path, the source needs to precompute $O(n)$ such aggregate proofs. The precomputation of proofs follows an iterative process. Each hop requires computation of an OPV field and a PVF, which can then be used to derive those of the next hop. Therefore, each hop requires two computations, and the n-hop path imposes a precomputation complexity of $O(n)$ on the source. The intermediate routers and the destination will then compare the computation result over the PVF (Figure 4 and Figure 5) with the OPV field (for OPT) or the OV field (for OSV) carried in the packet header. If the comparison matches, validation succeeds. Intermediate routers will further update the PVF using their shared keys with the source. All such verification and update of path proofs require a computational complexity of $O(1)$ by the intermediate routers and the destination. In contrast to OPT and OSV, Extended-OPT does not assume a trusted source. Instead, it assumes a trusted destination. Extended-OPT requires one more PVFs to be maintained in the packet header than does OPT, but it requires the same computational complexity.
- *PPV* [87]: Because PPV requires only two adjacent routers to add path proofs on packet headers, it has the smallest computational overhead compared with ICING, OPT, Extended-OPT, and OSV. According to its design (Section 4.3.4), all entities require a computation overhead of $O(1)$. Note that only routers that mark packets suffer this overhead.

4.4.3 Header Space Overhead. The space overhead for packet headers mainly stems from the path proofs. All of the deterministic validation protocols (i.e., ICING, OPT, Extended-OPT, and OSV) generate one proof field per router/destination for validation, inducing a space complexity of $O(n)$ for an n-hop path. Specifically, the proof field per hop consists of a 42-byte V field for ICING (Figure 3), a 16-byte OPV field for OPT and Extended-OPT (Figure 4), and a 2-byte OV field for OSV (Figure 5). In addition, ICING directly includes the path information in the packet headers, causing an additional space overhead of $O(n)$. Again, different from these deterministic solutions, PPV adds proof fields for only two routers in packet headers, and thus PPV incurs a space overhead of $O(1)$.

4.4.4 Storage Overhead. Finally, we summarize the storage overhead on the various entities. ICING enforces pairwise shared keys, and therefore each entity—the source, each intermediate router, and the destination—must store $O(n)$ keys. The source also needs to store the $O(n)$ path information such that it can (1) embed it into packet headers to indicate the intended nodes and (2) follow it to generation proofs for all other nodes. All of the other solutions assume either a trusted source or a trusted destination. Each node shares a key with the trusted source/destination,

requiring $O(1)$ keys to be stored. The trusted source/destination, however, must maintain $O(n)$ keys.

4.5 Directions for Improvement

The design space of path validation features a major trade-off between security and efficiency. This does not necessarily mean that more efficient solutions are less secure. Instead, they may gain efficiency by adopting stronger security assumptions. For example, OPT and OSV assume that a trusted source exists that can precompute part of path proofs for other nodes. In contrast, ICING assumes that none of the on-path nodes are trusted; thus, it requires each node to prove itself to all of the other nodes. Next, we summarize our observations regarding the security and efficiency properties of the existing path validation protocols (more discussions of their limitations and associated future directions will be covered in Section 6):

- *ICING*: We observe that the security level of ICING is bounded to one verifier instead of seemingly all verifiers. This is because each node N_i on the specified path verifies only its own verifier V_i. Because verifier V_i is the XORed result of several identically sized messages (i.e., an authenticator A_i and a number i of PoPs from all N_i's upstream nodes), an attacker does not have to forge every message to forge V_i. The attacker only needs to forge a message of the same size as V_i via, say, random guessing. The size of a verifier according to the ICING design is 42 bytes [50]. This limits the probability of a successful random guessing attack to $\frac{1}{2^{8\times42}}$, which is negligible.
- *OPT*: We observe a subtle but key difference between OPT and ICING that contributes to their efficiency gap. As mentioned previously, when the destination distrusts the source, Extended-OPT requires that the destination create one shared key with each of its upstream nodes. Then, for the destination to verify whether all of the upstream nodes truly did process the packet, each upstream node must perform one more MAC computation to generate proof of the destination. Similarly, if each on-path node needs to verify proofs from its upstream nodes without trusting the source, each on-path node must perform one MAC computation for each of its downstream nodes. This renders the number of MAC computations per node by OPT identical to that by ICING—that is, $O(n)$.
- *OSV*: Although computation over orthogonal sequences is faster than cryptographic computation, we observe that the OSV framework resembles OPT in the case of a trusted source. In other words, all of the verifier fields in the packet header can be precomputed by the source. Other on-path nodes do not share secrets. Then, for any pair of nodes N_i and N_j ($j > i$) excluding the source, N_i has no way to provide a nonforgeable proof of packet processing for N_j. This renders it difficult for N_j to ensure whether a verified packet has visited all predecessor nodes in the correct order, especially when it does not trust the source.
- *PPV*: Without careful configuration of the marking probability of each router under PPV, it is possible that a packet will not be marked by any router along its forwarding path. If this happens, an attacking packet may evade detection. PPV addresses this concern by setting the initial TTL in such a way that the marking probability at the last intermediate router (the router immediately preceding to the destination) is 1. Specifically, consider a forwarding path (S, R_1, R_2, R_3, D). For R_3 to guarantee a marking probability of 1, the TTL value of incoming packets should satisfy $1/TTL = 1$. In other words, the packets arriving at router R_3 should have TTL updated to 1. Because the TTL is decremented per hop, the source router S needs to initialize the TTL to 3. After two adjacent nodes have marked a packet, no subsequent router needs to mark that packet. Only the destination will verify the packet. However, this verification granularity opens up a security loophole. Specifically,

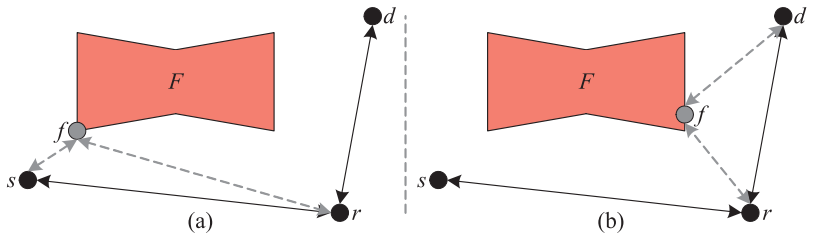

Fig. 7. Alibi routing (from Figure 1 in Levin et al. [52]). For source s to communicate with destination d without packets going through a forbidden region F, a relay node r is selected and required to leave a proof in the packets. This proof should negate the possibility of the packet traversing the forbidden region F. To this end, the relay node should be sufficiently far from the forbidden region such that transiting both the forbidden region and the relay induces a noticeably higher delay than transiting only the relay. Let $R(x, y)$ denote the end-to-end latency between two nodes x and y. For any node f in the forbidden region F, the selected relay r should satisfy both the following inequalities for a packet from s to d and its response from d to s: (a) $R(s, r) + R(r, d) \ll \min_f \{R(s, f) + R(f, r)\} + R(r, d)$; (b) $R(d, r) + R(r, s) \ll \min_f \{R(d, f) + R(f, r)\} + R(r, s)$.

after a packet has been marked by two routers, subsequent routers can arbitrarily forward it as long as the packet eventually arrives at the specified destination.

5 WHAT'S BEYOND? ALIBI ROUTING

The path validation solutions surveyed in Section 4 validate where packets should traverse rather than where they should not. It may seem that these two types of validation are identical—not traversing a node can be achieved by specifying a path without including the node. However, this holds only when all nodes behave in accordance with the specified forwarding process. However, if a certain on-path node misbehaves (Section 6.2), it may direct packets to a forbidden node that should be avoided, and that forbidden node might perform unwanted packet processing. Subsequently, the forbidden node forwards these packets toward the destination or back to the misbehaving on-path node. If the latter is the case and the forbidden node leaves no mark on packet headers, path validation solutions will be unable to detect such a deviation.

Therefore, it is necessary to design a path validation solution that generates proof of packets impossibly traversing through some node or region. To the best of our knowledge, alibi routing [52], as illustrated in Figure 7, is the only solution capable of doing this. The idea is to introduce a trusted relay node r that is sufficiently distant from the forbidden region F. If a packet traverses both r and some node f in F, it will exhibit noticeably higher latency than a node that traverses only r.

Alibi routing can guarantee proof that a forbidden region has been avoided if the key used for the relay to generate the proof is kept secret. Thus, we note that alibi routing fits better in scenarios where nodes in the forbidden region seek to manipulate the packets (e.g., drop or alter them) rather than simply overhear. In such scenarios, after the destination receives a packet signed by the relay, it can be assured that the packet has not been affected by any node in the forbidden region. In some other scenarios, where nodes in the forbidden region intend to infer certain privacy concerns over the passing packets, alibi routing becomes vulnerable to attacks by compromised on-path nodes that send copies of packets to the forbidden region, which is also a concern in the original alibi routing design [52]. Even when packets are encrypted, traffic ascription and communication patterns may be inferred [15, 53, 79]. However, this situation is hard to prevent by most path validation solutions. We will discuss this concern in Section 6.4, together with other underinvestigated aspects of path validation in Section 6.

6 WHAT'S LEFT TO UNFOLD? OPEN ISSUES AND FUTURE DIRECTIONS

In this section, we discuss the limitations of the current path validation protocols and explore feasible countermeasures. The limitations lie primarily in three aspects: security, privacy, and efficiency:

- *Security*: Because path validation relies on cryptography, it is intrinsically resistant to various security attacks, such as forging and replay (Section 6.1). However, most path validation protocols (e.g., ICING [62], OSV [21, 22], and alibi routing [52]) demonstrate their security properties by analysis. To date, only OPT [50] has been subjected to a formal security proof [97] (Section 6.2). We also discuss two other security concerns in the literature. One is how to attest that an on-path node has correctly processed packets as expected (Section 6.3). The other is hidden-node attacks, in which some off-path nodes may process packets without marking them (Section 6.4).
- *Privacy*: We find that there is a dilemma in path validation solutions regarding path privacy preservation (Section 6.5). For a node to verify whether a packet has correctly visited upstream hops, it should know the identities of those upstream nodes. However, for a node to prove its packet processing history to downstream nodes, it must also know the identities of those downstream nodes. Therefore, each on-path node is potentially aware of all other on-path nodes, making it hard to ensure path privacy. Furthermore, the path privacy leaks reveal end-host identities and thus prohibit anonymous communication.
- *Efficiency*: Although path validation solutions have evolved due to more computational power and reduced state size, they still cannot efficiently support flexible forwarding such as multipath routing (Section 6.6). In multipath routing [59, 88], a packet is allowed to take one of multiple paths; it may also switch among the allowable paths during forwarding. Consequently, given the likely large number of possible paths, it becomes prohibitively inefficient to apply current path validation protocols to multipath routing. Specifically, dedicating an independent state for encoding each path and concatenating the state for all possible paths in a packet header would significantly boost packet length, which would affect the efficiency of both packet processing and packet forwarding.

6.1 Attack Space

As mentioned in Section 2, the design goal of any path validation protocol is to protect forwarding against the inconsistencies illustrated in Figure 1 and Table 2. The illustrated inconsistencies include skipping, out-of-order, addition, and detour nodes. All of these forwarding inconsistencies can be essentially regarded as "detour" attacks because they detour packets away from the specified paths. Robustness against detour attacks is the key aspect of path validation solutions. Whether a packet takes the specified path must be verifiable via a packet-carried proof. An immediate security property of path validation should guarantee that the path validation proof is hard to forge. In other words, a counterfeit proof should fail path validation such that packets following unspecified paths can be detected and discarded. The attacker may also exploit more opportunistic attempts. For example, they may simply replay a valid proof or launch an attack when it is less likely to be detected. They may even collude to increase the attack success rate. All such attacks are typically intended to fool path validation using packets carrying invalid secrets (e.g., forged or expired) rather than to subvert path validation through, for example, simply dropping packets on a compromised router and leaving downstream routers with no packets to verify. We now review how such attacks affect forwarding and how robust various path validation solutions are against them.

Fig. 8. Counterfeit attacks [21, 22, 50, 62, 87] inject packets with forged proofs intended to pass path validation.

Detour. A detour attack causes the forwarding path of a packet to differ from its specified path [21, 22, 50, 62, 87]. The detour attack either directs a packet along on-path routers in an incorrect order or to off-path routers. Following the classification in Wu et al. [87], we showcased various detour instances in Figure 1 and explained them in Table 2. For ease of presentation, we hereby use a simple toy example with the source-selected path of ABCDE, where A and E are the source and the destination, respectively, and B, C, and D are intermediate routers. Figure 1 illustrates various detour attacks on the example forwarding path. We assume node B as a misbehaving/compromised node. Among all of the detour attacks, skipping (Figure 1(a)), addition (Figure 1(b)), partial detour (Figure 1(d)), and complete detour (Figure 1(e)) can easily be detected by the existing path validation protocols [21, 22, 50, 62, 87]. The major evidence they use is that an on-path router receives a detoured packet from an unspecified previous hop. For example, in the skipping attack (Figure 1(a)), router D receives the detoured packet from router B instead of the specified previous-hop C. An out-of-order attack is trickier (Figure 1(c)) because it directs a packet along the same set of routers as in the specified path but in a different sequence. As shown in Figure 1(b), the forwarding sequence is ABDCE instead of the specified ABCDE. In this case, E can easily detect the attack because it receives the detoured packet from C instead of the specified previous-hop D. A successful out-of-order attack requires collusion among routers. If nodes B, C, and D all collude to violate the forwarding order yet maintain the proof update order and still let D direct the detoured packet to destination E, then E will be unable to detect the out-of-order attack. However, collusion attacks require much greater capabilities from the attacker and are more difficult to deal with. We discuss collusion attacks in more detail in Section 6.1.

Counterfeit. A counterfeit attack aims to forge valid secrets and inject them into packets [21, 22, 50, 62, 87]. Specifically, the injected packets may come from the attacker instead of from the source. The attacker needs to forge a valid proof such that the injected packets can be verified as originating from the source and thus forwarded along the specified path (Figure 8(a)). However, because the attacker (i.e., a compromised router) does not have the keys used by other nodes for proof generation, it depends on random guesses to forge secrets. The probability of a successful guess decreases exponentially with proof length and can be negligible when the proof length is reasonably large. Counterintuitively, the attacker does need to forge all of the on-path nodes' proofs to pass path validation. As discussed in Section 4.5, the weakest link of path validation is the destination. In other words, if the attacker is the hop previous to the destination (e.g., router D in Figure 8(b)) and can successfully forge the proof corresponding to the destination, the attack will succeed regardless of whether the proofs of upstream nodes can be forged. Preventing this type of attack requires the length of an individual proof field to be sufficiently large. For example, the proof field by OPT [50] is 128 bits, which limits the probability of random guesses to under $\frac{1}{2^{128}}$. All existing path validation protocols [21, 22, 50, 52, 62, 87] are robust against counterfeit attacks.

Replay. Given the difficulty of forging proofs, the attacker may simply record valid packets and replay them later. This partly resembles the counterfeit attack in Figure 8 in that the compromised routers still inject packets. The difference is that the injected packets hold valid proofs. Replay attacks may jeopardize both the source and the destination. First, replaying valid packets leads to a larger amount of traffic than the actual traffic attributable to the source. If the packets request

billable services from intermediate nodes or the destination, the source will be charged more than expected. Second, the destination must process the replayed valid packets, causing it to suffer from a potentially large waste of resources. A high-rate replay attack can thus be exploited as a DoS attack. All path validation protocols should include countermeasures against replay attacks as a key component. However, defense against replay attacks is a challenge faced by all Internet architectures [51] and should be orthogonal to path validation.

OPT [50] investigated how to augment path validation with countermeasures against replay attacks. One intuitive idea is to let intermediate routers and the destination cache unique metadata from each received packet, such as its carrying proof or counter. Upon receiving a packet, a router or destination first compares the packet's metadata with the cached data. Any match found during the comparison reveals a replay attack. This idea is effective but impractical due to the high storage overhead required. A feasible enhancement is to divide the communication between the source and the destination into sessions [50]. Within each session, all packets have the same timestamp. Received packets with obsolete timestamps (e.g., from days ago) indicate replay attacks and should be discarded. In this way, routers and the destination only need to cache packets received within some number of recent sessions.

Byzantine. A Byzantine attack (also known as a coward attack [50]) on path validation occurs when misbehaving routers violate path validation protocols only when they are less likely to be detected. For example, path validation is invoked only when forwarding anomalies such as long delays or packet losses occur or when the sender or destination receives a new service requirement that causes packets to traverse a path different from the default path. After the forwarding anomalies are fixed or if the new service is not applied to subsequent traffic, path validation may be switched off separately from packet forwarding, which makes it difficult to detect such misbehaviors by a sophisticated attacker (e.g., a compromised router) during the period that path validation is not in effect. In comparison with the preceding attacks, Byzantine attacks raise more challenges but attract less attention.

Neither ICING [62] nor OSV [21, 22] investigate countermeasures. We find this reasonable if path validation is a prerequisite for packet forwarding. In other words, if every data packet carries path secrets, and a packet is accepted only if the secrets pass path validation, then until all of the on-path nodes have generated and exchanged the necessary keys for verifying and updating path secrets, no data packet will be transmitted. Note that we use the term *data packet* to differentiate these packets from packets used to exchange keys during the setup phase.

An extended version of OPT mitigates Byzantine attacks by removing the need for the explicit key setup process [50]. In this way, routers no longer have a clear signal regarding when path validation takes effect. Thus, malicious node cannot determine when to behave correctly according to the path validation protocol and when to behave otherwise without being detected. This forces routers to perform consistently during packet processing and forwarding. The extended OPT assumes that the source and destination trust each other and have shared keys. Consequently, they do not need to exchange keys with intermediate routers before sending packets. Because the source has no shared keys with the intermediate routers, it simply initializes PVF_0 in the packet header without OSVs. Upon receiving the packet, each intermediate router derives its shared key with the destination on the fly and updates PVF with the derived key. The destination must cache the received packets for a certain period of time, within which the destination can trigger path validation at any time. Upon a path validation request, each intermediate router can reveal the key it used for updating PVF to the destination.

Although it mitigates Byzantine attacks, the extended OPT still requires routers to leave proofs for future verification. Because packets must be cached on the destination until verification, a delay

Table 4. Robustness of Existing Path Validation Protocols [21, 22, 50, 62, 87] Against Common Attacks in the Literature

Attack	ICING [62]	(Extended-) OPT [50]	OSV [21, 22]	PPV [87]	Alibi Routing [52]
Skipping	✓	✓	✓	✓	✓*
Out-of-order	✗ if collusion ✓ otherwise	✗ if collusion ✓ otherwise	✗ if collusion ✓ otherwise	✗ if collusion ✓ otherwise	✗ if collusion ✓* otherwise
Addition	✓	✓	✓	✓	✓*
Partial Detour	✓	✓	✓	✓	✓*
Complete Detour	✓	✓	✓	✓	✓*
Counterfeit	✓	✓	✓	✓	✓*
Replay	✗ (orthogonal)	✓**	✗ (orthogonal)	✗ (orthogonal)	✗ (orthogonal)
Byzantine	✓	✓	✓	✓	✓*
Collusion	✗	✗	✗	✗	✗

Notes:

* Alibi routing [52] can detect these kinds of attacks only when the lengths of the intended and actual paths differ significantly.

** Replay attack resistance is also orthogonal to OPT [50], which aims at path validation. OPT, however, suggests timestamping packets to mitigate replay attacks.

is imposed. Furthermore, requiring routers to leave proofs—with or without an explicit key setup process—may alert routers of future verification. This renders the problem as approached by OPT [50] vulnerable to a weak version of the Byzantine attack. A more challenging version, in which path validation is either on or off, would be an interesting research direction.

We suggest that following the original OPT but allowing no data packets to be sent until a successful key setup has completed might be a faster choice because it allows the destination to verify a received packet instantly. Specifically, the suggested design is a hybrid design that reaps the benefits of both ICING/OSV and OPT. First, following ICING and OSV, it does not allow transmission of data packets during the key setup phase. Second, after the key setup completes, it follows OPT to achieve faster path validation than ICING and OSV. The hybrid design requires no further design techniques. It simply twists the framework of existing path validation protocols. We can understand it as ICING/OSV that preserves the key setup phase yet replaces the data packet transmission phase with OPT. We can also understand it as OPT, in which the key setup phase is replaced with ICING/OSV but the data packet transmission phase remains untouched. Because data packet transmission is allowed only when path validation is on, the attacker cannot successfully launch Byzantine attacks.

Collusion. Two or more compromised/malicious nodes on a path may collude to launch any of the preceding attacks [21, 22, 50, 62, 87]. Again, consider the source-specified path ABCDE. B and D could collude to skip C (Figure 1(a)). Instead of verifying the packet, D might directly update the packet with its proof and forward it to E, but E could easily detect this detour attack due to the lack of C's proof in the packet header using existing the path validation protocols. One may wonder what happens if B, C, and D collude. For example, they could skip C but still have C provide its proof. This trickier version of s detour attack cannot be detected by E. The intention behind such a collusion is intended to make packets bypass C's processing but present as if they had been processed by C. All path validation protocols consider this situation as unaddressable. We will discuss this case in Section 6.3 and suggest a countermeasure.

Table 4 summarizes the robustness of existing path validation protocols against the preceding attacks. All of the existing protocols can successfully fulfill the security goals for which they were

designed. Specifically, when a packet does not follow the specified path and its proof is not updated according to the specified sequence, the detoured packet will not be accepted by the existing path validation protocols. Furthermore, the existing path validation protocols ensure that their proofs are sufficiently robust against forging attacks. Attackers are highly unlikely to be able to forge a valid proof as if a packet had truly taken a specified path. However, their robustness against more sophisticated attacks, such as replay and collusion attacks, are not part of the security goals of path validation; thus, these attacks necessitate augmenting orthogonal countermeasures.

6.2 Formal Security Proof

Although the security analysis for most path validation protocols—ICING [62], OSV [21, 22], and PPV [87]—depend heavily on arguments about how the path validation protocol defends against common attacks, only OPT [50] has been subjected to formal security proofs [97]. Because path validation relates to network protocols, its proof should be applicable to arbitrary network topologies; however, this approach is more challenging than proving cryptographic protocols [97]. Specifically, the proofs in Zhang et al. [97] focus on the version of OPT with an untrusted source—that is, where each router shares keys with both the source and the destination (Section 4.3.2). The properties to prove include (1) the secrecy and authenticity of the key setup phase and (2) the authenticity of origin/source and the path of the packet during the forwarding phase. OPT was first formally expressed using LS^2 [31], which reasons about programs executing concurrently with programs controlled by adversaries. Adversaries are interface confined [33] to reasonably limit their capabilities. Moreover, to reason about protocols using cryptographic functions, LS^2 is extended with the relevant definitions of data structures and axioms formalized by protocol composition logic (PCL) [30]. Finally, LS^2-encoded OPT runs in Coq [1], a security proof assistance system that OPT uses to emulate an adversarial environment. We refer interested readers to Zhang et al. [97] for a detailed proof.

Instead, what we would emphasize here is that the proof of OPT focuses on only the chained MAC technique. Proofs over techniques such as aggregated MACs by ICING [62] and orthogonal sequences by OSV [21, 22] remain untouched and would also be critical for deploying path validation.

6.3 Packet Processing Attestation

All path validation protocols acknowledge that they are incapable of verifying whether required packet processing has taken effect [21, 22, 50, 62]. This concern is reasonable because a router adding its proof to a packet does not guarantee that it faithfully processed the packet as intended. Consider, for example, a router connecting to a middlebox for deep packet inspection. If compromised, the router may not direct packets to the middlebox [18]. Even if the router behaves correctly, the middlebox may not inspect received packets as expected. In both cases, as long as the router adds its proof to packets and sends them to the correct next hop, the downstream routers and the destination will be unable to detect these uninspected packets using path validation protocols. Attestation of packet processing is orthogonal to yet imperative for strengthening path validation.

We suggest a probing-based method to indirectly attest to packet processing. The idea is to inject probe packets along with production packets. Probe packets are constructed in such a way that their expected processing result is known to the destination. Consider a firewall service as an example. When the source or destination acquires a firewall service that filters unwanted traffic, it needs to verify that it is working correctly and faithfully. To this end, the source can send a probe packet that the firewall should filter. If the destination receives the probe packet, it can detect any inaccurate or malicious behavior by the firewall. One may wonder about the difference between packet processing attestation and conventional service tests. In other words, can the effect of

probing packets not be achieved by testing the service quality prior to using the service by the source or destination? The answer is no. To logically explain this answer, recall the difference between path validation and secure routing. Secure routing ensures that all of the forwarding paths constructed by routing protocols conform to routing polices. However, secure routing cannot ensure whether packets are forwarded via paths they should follow, which is why path validation is introduced to verify the forwarding behavior. Similarly, service test results do not necessarily represent the actual service behavior. Consider the firewall example again. As the source and destination communicate for service, the source randomly or periodically injects probe packets. Whenever the results of probe packets vary from the expected results, the destination suspects the forwarding devices of inappropriate packet processing.

A sophisticated attacker who understands packet-processing attestation may process probe packet normally while still manipulating production packets. Therefore, probe packets should be indistinguishable from production packets. Additionally, probe packets should carry random indicators synchronized between the source and destination. Such indicators help the destination identify probe packets and discard them to avoid performance and security impacts.

Packet processing attestation is, however, orthogonal to path validation and cannot enforce per-packet attestation. Path validation forces packets to be forwarded along the specified paths, but it is not designed to verify whether packet processing other than forwarding has occurred. In contrast, packet processing attestation enables the source and destination to inject probe packets and verifies the validity of their processing. Probe packets must not be production packets. The essential reason for introducing probe packets is because of their predetermined processing results. This makes packet processing attestation over probe packets only an indicator of packet processing behaviors. Specifically, if probe packets are found to be incorrectly processed, the source and destination can suspect that misprocessing may have also occurred to production packets. The more misprocessed probe packets that are detected, the more suspicious the processing results of production packets become. Attestation over production packets per se is, however, not supported.

6.4 Hidden-Node Attack

If nodes on the specified path are compromised, they might detour packets to some off-path nodes. Such off-path nodes may perform unwanted packet inspection without leaving traces and then direct them back to the specified path. Consider, for example, the specified path $N_0N_1N_2N_3$. A compromised N_1 may detour packets to an off-path node (N_x), which then directs the packets back to N_1. In another case, if N_2 is also compromised and colludes with N_1, the off-path node N_x could direct packets back to N_2. This protocol breach is referred to as a hidden-node attack [62], in which the hidden nodes correspond to the off-path nodes. To avoid being noticed by end-hosts, hidden nodes do not interfere with communication by, for example, altering or dropping traffic. Ensuring authenticity against packet alteration can easily be achieved with cryptographic schemes (e.g., MACs and signatures). However, because hidden nodes do not leave marks on packets, the path validation protocols surveyed in Section 4 cannot detect hidden-node attacks.

Depending on whether hidden nodes are involved in the packet forwarding process, hidden-node attack can be further classified into two types—detour attacks and copy attacks:

- As mentioned previously, detour attacks involve hidden nodes in the packet forwarding process. A packet redirected to off-path hidden nodes will eventually return to the specified path, and if the detoured packet still traverses the routers on the specified path in the specified sequence, it will successfully satisfy any path validation in Section 4. However, given that hidden nodes are off the specified path, one may wonder whether we could resort to alibi routing [52] focusing on forwarding through a forbidden region in Section 5. We observe

that alibi routing can detect detour attacks if the hidden node is sufficiently far away from the specified path such that it causes a noticeable transmission delay. End-hosts can use such this delay to easily detect hidden-node attacks. In contrast, if the hidden node is only few hops away from compromised on-path nodes, detouring packets through them would cause little delay. A slight delay is difficult to differentiate from normal transmission-time fluctuations. In this case, alibi routing cannot be adopted to detect hidden-node attacks.

- In copy attacks, rather than detouring packets through hidden nodes, compromised on-path nodes simply copy packets and forward the copies to hidden nodes while correctly forwarding the original packets to the next hop on-path node [62]. In other words, copy attacks do not affect the packet forwarding process. All of the existing path validation solutions in Section 4 and alibi routing in Section 5 thus are unable to detect or prevent copy attacks. However, a best-effort strategy is to make packet copies less useful to hidden nodes [52]. We can protect the confidentiality of packet payloads by encrypting them, and can protect the anonymity of packet sources using, for example, Tor [80].

Most path validation protocols consider hidden-node attacks difficult to prevent and detect [50, 52, 62]; therefore, their common advice is that the sender and the receiver should choose nodes with sufficient trust to form the specified path. In this way, the on-path nodes are less likely to be compromised and to detour packets or send copies to hidden nodes.

6.5 Host Anonymity and Path Privacy

Path validation cannot preserve anonymity and path privacy. End-host anonymity aims to protect the identities of the source and the destination during their communication. However, end-host anonymity implies the requirement of path privacy as well. If the path information is leaked, it is easier to track down end-hosts connected by the path. Therefore, anonymous communication protocols vary packet formats per hop to prevent path privacy leakage by packet correlation. The typical protocols for anonymous communication are built upon onion routing [81] or its evolved version Tor [80]. In such protocols, the source specifies a series of relay nodes that constitute an overlay network. After negotiating keys with each relay node, the source encrypts its message m to the destination using nested encryption. Consider, for example, that source S and destination D communicate via three relay nodes R_1, R_2, and R_3. Let k_1, k_2, and k_3 represent the keys used by S to communicate with R_1, R_2, and R_3, respectively. Then, the encrypted message sent from S to R_1 is constructed as follows: $\text{Enc}_{k_1}(R_2, \text{Enc}_{k_2}(R_3, \text{Enc}_{k_3}(message, D)))$.

Upon receiving the preceding encrypted message, R_1 decrypts it but obtains only the next hop specification R_2 and the encrypted message targeted at R_2—$\text{Enc}_{k_2}(R_3, \text{Enc}_{k_3}(message, D))$. Similarly, R_2 forwards the encrypted message $\text{Enc}_{k_3}(message, D)$ to R_3, which finally forwards the message to D. (Note that the message should be encrypted with a key shared between S and D to protect confidentiality.) Each relay node can infer from the received encrypted message is only its previous hop and its next hop. Such inferred information is the minimum required by any forwarding protocol, and it is the maximum information revealed by onion routing. In other words, other than the first relay node, the other relay nodes (e.g., R_2 and R_3) cannot know the source. This helps guarantees source anonymity, and destination anonymity can be protected in the same way. However, path validation protocols require that path information be revealed to on-path nodes through a packet-carried state (e.g., ICING [62]) or a preloaded configuration (e.g., OPT [50]). When this is not the case, a node cannot ensure packet-processing proof of the previous hops it needs to verify.

6.6 Forwarding Flexibility

Current path validation protocols bind a packet with only one fixed path. Therefore, they cannot efficiently be applied to flexible forwarding, which allows packets to switch paths during

forwarding. Flexible forwarding is built on multipath routing protocols. Unlike traditional routing protocols such as BGP, multipath routing computes multiple forwarding paths for each packet. There are two types of forwarding. The first type selects one out of the multiple available paths and forwards the packet along the selected path [88]. This type of forwarding usually provides benefits for traffic engineering, such as congestion control and load balancing. In this case, each packet is still coupled with one path during forwarding, and the existing path validation protocols can be directly applied. Here, we are more interested in the second type, which allows on-path nodes to switch packets to another backup path [59]. This approach aims to improve forwarding robustness through, for example, fast failover. As a typical example of multipath routing, path splicing [59] organizes a packet's multiple paths into a tree structure. The number of possible paths thus may be exponential to the path length, offering substantial forwarding flexibility. However, it is difficult for current path validation protocols to encode this enormous set of paths as packet-carried states. Finding more practical and efficient path encoding techniques is therefore imperative for path validation under flexible forwarding. When pursuing secure validation designs, finding a trade-off between forwarding flexibility and validation efficiency would also be an encouraging research direction.

6.7 End-Host Autonomy Versus Internet Stability

Finally, we investigate how deploying path validation schemes affects the Internet ecosystem. Specifically, when end-hosts are allowed to select the forwarding paths, will this autonomy disrupt the stability the current Internet already has? For example, if a large number of end-hosts were to request the use of a specific forwarding path, that path could become congested, affecting throughput as well as service quality. Note that this concern is not be unique to the incorporation of path validation. The current Internet faces a similar situation. For example, the most commonly used shortest-path routing protocols compute the end-to-end shortest paths for packet forwarding; thus, it is common for many end hosts to flood the same shortest path and cause congestion [44]. In this case, the shortest path forwarding turns out to not be the fastest form of forwarding as intended. Meanwhile, longer paths may be used for faster forwarding [94]. This is why load balancing is widely deployed in intranets [46]. It balances traffic volume across various links even when some traffic is directed off the supposed shortest path. All such methods also apply to path validation in an augmented Internet. For example, when allocating specified paths to end-hosts, the path allocation process should consider the current network status to optimize performance requirements such as load balancing.

To quickly converge to network dynamics, path validation should adopt short sessions. This is because path validation usually enforces sessionwise path allocation. A session usually spans 30 minutes but can be configured to span from seconds to hours. For link-quality sensitive applications, end-hosts can be persuaded to adopt short sessions. Whenever network dynamics lead to different path allocation to suit their performance requirements, they would be able to switch to the newly assigned paths quickly without much impact on performance. Handling network dynamics can be divided into two phases. The first is how to allocate paths based on network dynamics. The second is how to validate packets forwarded along the newly allocated paths. The latter focuses on path validation designs per se—that is, no matter how the forwarding paths are allocated, the existing path validation protocols simply take the allocated paths as an input and enforce packet forwarding along them. In other words, handling network dynamics should not be a technical focus of path validation. Instead, path validation should focus more on constructing routing policies, especially in the first phase—path allocation. However, this aspect is orthogonal to path validation and is beyond the scope of this work. Nevertheless, we consider it an interesting research direction.

7 CONCLUSION

We have presented, to the best of our knowledge, the first comprehensive survey of research on validating network paths. Due to various flexibility, security, and privacy limitations, traditional network architecture has long been insufficient for fostering innovative Internet services. Regardless of whether the future Internet follows a clean slate or an evolutionary design [72], validating network paths is critical for security reasons [13]. Path validation strives for two enhancements over traditional packet delivery. One is path enforcement, which causes routers to forward packets along paths specified by the source, destination, or intermediate routers. The other is path verification, which enables intermediate routers and the destination to verify whether a packet has taken its specified path. In summary, path validation requires that the source specifies in a packet header which routers the packet should traverse and their sequence. Routers then forward the packet according to this path directive. Furthermore, the source and each router should embed a series of proofs in the packet header, one for each downstream node (i.e., downstream routers and the destination). Such proofs allow a node to verify path compliance. This packet-carried state should be difficult to forge or to tamper with. With path validation, network entities gain more control over packet delivery. A variant of path validation could even enable end-hosts to verify whether their packets have undergone a detour from a certain region [52].

Although path validation can ensure higher service quality and security, this discipline has attracted only a small body of work. One major reason is that path validation imposes radical changes on the current Internet architecture, making it difficult to evaluate and deploy path validation solutions. We consider that the rarity of path validation solutions might also be attributable to a long-standing confusion between routing and forwarding. Routing aims to find paths for pairs of end-hosts, whereas forwarding aims to direct packets from one end-host to another. Compromised routers may make forwarding decisions that deviate from the paths generated by the routing protocols. In other words, securing the routing process alone cannot guarantee correct packet forwarding. It is also insufficient to merely enforce packet delivery along a specified path or to simply verify which path a packet has taken. Path enforcement and path verification should be jointly adapted to accomplish path validation. This strict requirement renders many routing and forwarding solutions unqualified for validating network paths. Clearly, to satisfy both path enforcement and verification, the cryptographic state carried in packet headers would be quite large in size. A major trend in path validation design therefore involves shortening the state. However, we find that the efficiency gains come at the cost of reduced security. We conducted a comprehensive analysis of the properties and limitations of existing path validation solutions, which revealed that they are still limited along various aspects, thus forming a wide range of inspiring and exciting research directions to further advance security, privacy, and efficiency.

It is our hope that this survey article benefits both the research and industry community by providing useful references and research directions for validating network paths. Together, we shall strive for a better Internet.

ACKNOWLEDGMENTS

The authors would also like to sincerely thank the editors and reviewers of *ACM Computing Surveys* and AJE editors for their review efforts and helpful feedback.

REFERENCES

[1] Coq. 2017. The Coq Proof Assistant. Retrieved July 28, 2020 from https://coq.inria.fr/.
[2] Wikipedia. 2019. Default/ Route. Retrieved July 28, 2020 from https://en.wikipedia.org/wiki/Default_route.
[3] IRTF. 2019. Path Aware Networking Research Group PANRG. Retrieved July 28, 2020 from https://irtf.org/panrg.

[4] IETF. 2019. Path Aware Networking RG (panrg). Retrieved July 28, 2020 from https://datatracker.ietf.org/rg/panrg/about/.
[5] Bahaa Al-Musawi, Philip Branch, and Grenville Armitage. 2017. BGP anomaly detection techniques: A survey. *IEEE Communications Surveys & Tutorials* 19, 1 (2017), 377–396.
[6] David G. Andersen, Hari Balakrishnan, Nick Feamster, Teemu Koponen, Daekyeong Moon, and Scott Shenker. 2008. Accountable Internet Protocol (AIP). *ACM SIGCOMM Computer Communication Review* 38, 4 (2008), 339–350.
[7] Tom Anderson, Ken Birman, Robert Broberg, Matthew Caesar, Douglas Comer, Chase Cotton, Michael J. Freedman, et al. 2014. A brief overview of the NEBULA future Internet architecture. *ACM SIGCOMM Computer Communication Review* 44, 3 (2014), 81–86.
[8] Katerina Argyraki and David R. Cheriton. 2004. Loose source routing as a mechanism for traffic policies. In *Proceedings of ACM FDNA 2004*. 57–64.
[9] Katerina Argyraki, Petros Maniatis, Olga Irzak, Subramanian Ashish, and Scott Shenker. 2007. Loss and delay accountability for the Internet. In *Proceedings of IEEE ICNP 2007*. 194–205.
[10] Edvard F. Assmus and Jennifer D. Key. 1994. *Designs and Their Codes*. Vol. 103. Cambridge University Press.
[11] Ioannis Avramopoulos, Hisashi Kobayashi, Randy Wang, and Arvind Krishnamurthy. 2004. Highly secure and efficient routing. In *Proceedings of IEEE INFOCOM 2004*. 197–208.
[12] Ioannis C. Avramopoulos and Jennifer Rexford. 2006. Stealth probing: Efficient data-plane security for IP routing. In *Proceedings of USENIX ATC 2006*. 267–272.
[13] David Barrera, Laurent Chuat, Adrian Perrig, Raphael M. Reischuk, and Pawel Szalachowski. 2017. The SCION Internet architecture. *Communications of the ACM* 60, 6 (2017), 56–65.
[14] Bobby Bhattacharjee, Ken Calvert, Jim Griffioen, Neil Spring, and James P. G. Sterbenz. 2006. *Postmodern Internetwork Architecture*. Technical Report. University of Kansas.
[15] George Dean Bissias, Marc Liberatore, David Jensen, and Brian Neil Levine. 2005. Privacy vulnerabilities in encrypted HTTP streams. In *Privacy Enhancing Technologies*. Lecture Notes in Computer Science, Vol. 3856. Springer, 1–11.
[16] Ilker Nadi Bozkurt, Anthony Aguirre, Balakrishnan Chandrasekaran, P. Brighten Godfrey, Gregory Laughlin, Bruce Maggs, and Ankit Singla. 2017. Why is the Internet so slow?! In *Proceedings of PAM 2017*. 173–187.
[17] Andrei Broder and Michael Mitzenmacher. 2004. Network applications of Bloom filters: A survey. *Internet Mathematics* 1, 4 (2004), 485–509.
[18] Kai Bu, Yutian Yang, Zixuan Guo, Yuanyuan Yang, Xing Li, and Shigeng Zhang. 2018. FlowCloak: Defeating middlebox-bypass attacks in software-defined networking. In *Proceedings of IEEE INFOCOM 2018*. 396–404.
[19] Kai Bu, Yutian Yang, Avery Laird, Jiaqing Luo, Yingjiu Li, and Kui Ren. 2018. What's (not) validating network paths: A survey. arXiv:1804.03385 [cs.NI].
[20] Kevin Butler, Toni R. Farley, Patrick McDaniel, and Jennifer Rexford. 2010. A survey of BGP security issues and solutions. *Proceedings of the IEEE* 98, 1 (2010), 100–122.
[21] Hao Cai and Tilman Wolf. 2015. Source authentication and path validation with orthogonal network capabilities. In *Proceedings of IEEE INFOCOM Workshops 2015*. 111–112.
[22] Hao Cai and Tilman Wolf. 2016. Source authentication and path validation in networks using orthogonal sequences. In *Proceedings of IEEE ICCCN 2016*. 1–10.
[23] Kenneth L. Calvert, James Griffioen, and Leonid Poutievski. 2007. Separating routing and forwarding: A clean-slate network layer design. In *Proceedingsof IEEE BROADNETS 2007*. 261–270.
[24] Martin Casado, Michael J. Freedman, Justin Pettit, Jianying Luo, Nick McKeown, and Scott Shenker. 2007. Ethane: Taking control of the enterprise. *ACM SIGCOMM Computer Communication Review* 37, 4 (2007), 1–12.
[25] André Castelucio, Antônio Tadeu A. Gomes, Artur Ziviani, and Ronaldo M. Salles. 2012. Intra-domain IP traceback using OSPF. *Computer Communications* 35, 5 (2012), 554–564.
[26] Ang Chen, Andreas Haeberlen, Wenchao Zhou, and Boon Thau Loo. 2017. One primitive to diagnose them all: Architectural support for Internet diagnostics. In *Proceedings of EuroSys 2017*.
[27] Chen Chen, Daniele E. Asoni, David Barrera, George Danezis, and Adrain Perrig. 2015. HORNET: High-speed onion routing at the network layer. In *Proceedings of ACM CCS 2015*. 1441–1454.
[28] K. H. Choi and H. K. Dai. 2004. A marking scheme using Huffman codes for IP traceback. In *Proceedings of IEEE ISPAN 2004*. 421–428.
[29] Luca Cittadini, Giuseppe Di Battista, and Massimo Rimondini. 2012. On the stability of interdomain routing. *ACM Computing Surveys* 44, 4 (2012), 26.
[30] Anupam Datta, Ante Derek, John C. Mitchell, and Arnab Roy. 2007. Protocol composition logic (PCL). *Electronic Notes in Theoretical Computer Science* 172 (2007), 311–358.
[31] Anupam Datta, Jason Franklin, Deepak Garg, and Dilsun Kaynar. 2009. A logic of secure systems and its application to trusted computing. In *Proceedings of IEEE S&P 2009*. 221–236.

[32] Wenxiu Ding, Zheng Yan, and Robert H. Deng. 2016. A survey on future Internet security architectures. *IEEE Access* 4 (2016), 4374–4393.
[33] Deepak Garg, Jason Franklin, Dilsun Kaynar, and Anupam Datta. 2010. Compositional system security in the presence of interface-confined adversaries. *Electronic Notes in Theoretical Computer Science* 265 (2010), 49–71.
[34] P. Godfrey, Igor Ganichev, Scott Shenker, and Ion Stoica. 2009. Pathlet routing. *ACM SIGCOMM Computer Communication Review* 39, 4 (2009), 111–122.
[35] Sharon Goldberg, David Xiao, Boaz Barak, and Jennifer Rexford. 2007. *Measuring Path Quality in the Presence of Adversaries: The Role of Cryptography in Network Accountability*. Technical Report. Department of Computer Science, Princeton University, Princeton, NJ.
[36] Michael T. Goodrich. 2002. Efficient packet marking for large-scale IP traceback. In *Proceedings of ACM CCS 2002*. 117–126.
[37] Saikat Guha and Paul Francis. 2007. An end-middle-end approach to connection establishment. *ACM SIGCOMM Computer Communication Review* 37 (2007), 193–204.
[38] P. Krishna Gummadi, Harsha V. Madhyastha, Steven D. Gribble, Henry M. Levy, and David Wetherall. 2004. Improving the reliability of Internet paths with one-hop source routing. In *Proceedings of USENIX OSDI 2004*, Vol. 4. 13.
[39] Andreas Haeberlen, Petr Kouznetsov, and Peter Druschel. 2007. PeerReview: Practical accountability for distributed systems. In *Proceedings of ACM SOSP 2007*, Vol. 41. 175–188.
[40] Nikhil Handigol, Brandon Heller, Vimalkumar Jeyakumar, David Mazières, and Nick McKeown. 2014. I know what your packet did last hop: Using packet histories to troubleshoot networks. In *Proceedings of USENIX NSDI 2014*. 71–85.
[41] Matthias Hollick, Cristina Nita-Rotaru, Panagiotis Papadimitratos, Adrian Perrig, and Stefan Schmid. 2017. Toward a taxonomy and attacker model for secure routing protocols. *ACM SIGCOMM Computer Communication Review* 47, 1 (2017), 43–48.
[42] Yih-Chun Hu, Adrian Perrig, and Marvin Sirbu. 2004. SPV: Secure path vector routing for securing BGP. In *Proceedings of ACM SIGCOMM 2004*. 179–192.
[43] Geoff Huston, Mattia Rossi, and Grenville Armitage. 2011. Securing BGP—A literature survey. *IEEE Communications Surveys & Tutorials* 13, 2 (2011), 199–222.
[44] Junchen Jiang, Rajdeep Das, Ganesh Ananthanarayanan, Philip A. Chou, Venkata Padmanabhan, Vyas Sekar, Esbjorn Dominique, et al. 2016. Via: Improving Internet telephony call quality using predictive relay selection. In *Proceedings of ACM SIGCOMM 2016*. 286–299.
[45] Jian Jiang, Wei Li, Junzhou Luo, and Jing Tan. 2013. A network accountability based verification mechanism for detecting inter-domain routing path inconsistency. *Journal of Network and Computer Applications* 36, 6 (2013), 1671–1683.
[46] Naga Katta, Aditi Ghag, Mukesh Hira, Isaac Keslassy, Aran Bergman, Changhoon Kim, and Jennifer Rexford. 2017. Clove: Congestion-aware load balancing at the virtual edge. In *Proceedings of ACM CoNEXT 2017*. 323–335.
[47] Jonathan Katz and Andrew Lindell. 2008. Aggregate message authentication codes. In *Topics in Cryptology—CT-RSA 2008*. Lecture Notes in Computer Science, Vol. 4964. Springer, 155–169.
[48] H. Tahilramani Kaur, Shivkumar Kalyanaraman, Andreas Weiss, Shifalika Kanwar, and Ayesha Gandhi. 2003. BANANAS: An evolutionary framework for explicit and multipath routing in the Internet. *ACM SIGCOMM Computer Communication Review* 33, 4 (2003), 277–288.
[49] Stephen Kent, Charles Lynn, and Karen Seo. 2000. Secure border gateway protocol (S-BGP). *IEEE Journal on Selected Areas in Communications* 18, 4 (2000), 582–592.
[50] Tiffany Hyun-Jin Kim, Cristina Basescu, Limin Jia, Soo Bum Lee, Yih-Chun Hu, and Adrian Perrig. 2014. Lightweight source authentication and path validation. In *Proceedings of ACM SIGCOMM 2014*, Vol. 44. 271–282.
[51] Taeho Lee, Christos Pappas, Adrian Perrig, Virgil Gligor, and Yih-Chun Hu. 2017. The case for in-network replay suppression. In *Proceedings of ASIACCS 2017*. 862–873.
[52] Dave Levin, Youndo Lee, Luke Valenta, Zhihao Li, Victoria Lai, Cristian Lumezanu, Neil Spring, and Bobby Bhattacharjee. 2015. Alibi routing. In *Proceedings of ACM SIGCOMM 2015*, Vol. 45. 611–624.
[53] Marc Liberatore and Brian Neil Levine. 2006. Inferring the source of encrypted HTTP connections. In *Proceedings of ACM CCS 2006*. 255–263.
[54] Robert Lychev, Michael Schapira, and Sharon Goldberg. 2016. Rethinking security for Internet routing. *Communications of the ACM* 59, 10 (2016), 48–57.
[55] S. Malliga and A. Tamilarasi. 2008. A proposal for new marking scheme with its performance evaluation for IP traceback. *WSEAS Transactions on Computer Research* 3, 4 (2008), 259–272.
[56] S. Malliga and A. Tamilarasi. 2010. A hybrid scheme using packet marking and logging for IP traceback. *International Journal of Internet Protocol Technology* 5, 1–2 (2010), 81–91.

[57] David A. Maltz, Geoffrey Xie, Jibin Zhan, Hui Zhang, Gísli Hjálmtỳsson, and Albert Greenberg. 2004. Routing design in operational networks: A look from the inside. *ACM SIGCOMM Computer Communication Review* 34, 4 (2004), 27–40.
[58] David Mazieres, Michael Kaminsky, M. Frans Kaashoek, and Emmett Witchel. 1999. Separating key management from file system security. In *Proceedings of ACM SOSP 1999*. 124–139.
[59] Murtaza Motiwala, Megan Elmore, Nick Feamster, and Santosh Vempala. 2008. Path splicing. In *Proceedings of ACM SIGCOMM 2008*. 27–38.
[60] John Moy. 1998. OSPF Version 2. Retrieved July 28, 2020 from https://tools.ietf.org/html/rfc2328.
[61] John T Moy. 1998. *OSPF: Anatomy of an Internet Routing Protocol*. Addison-Wesley Professional.
[62] Jad Naous, Michael Walfish, Antonio Nicolosi, David Mazières, Michael Miller, and Arun Seehra. 2011. Verifying and enforcing network paths with icing. In *Proceedings of ACM CoNEXT 2011*. 30.
[63] Martin O. Nicholes and Biswanath Mukherjee. 2009. A survey of security techniques for the border gateway protocol (BGP). *IEEE Communications Surveys & Tutorials* 11, 1 (2009), 52–65.
[64] David Oran. 1990. OSI IS-IS Intra-Domain Routing Protocol. Retrieved July 28, 2020 from https://tools.ietf.org/html/rfc1142.
[65] Venkata N. Padmanabhan and Daniel R. Simon. 2003. Secure traceroute to detect faulty or malicious routing. *ACM SIGCOMM Computer Communication Review* 33, 1 (2003), 77–82.
[66] Adrian Perrig, Pawel Szalachowski, Raphael M. Reischuk, and Laurent Chuat. 2017. *Scion: A Secure Internet Architecture*. Springer.
[67] Simon Peter, Umar Javed, Qiao Zhang, Doug Woos, Thomas Anderson, and Arvind Krishnamurthy. 2014. One tunnel is (often) enough. *ACM SIGCOMM Computer Communication Review* 44, 4 (2014), 99–110.
[68] Lucian Popa, Norbert Egi, Sylvia Ratnasamy, and Ion Stoica. 2010. Building extensible networks with rule-based forwarding. In *Proceedings of USENIX OSDI 2010*. 379–392.
[69] Barath Raghavan and Alex C. Snoeren. 2004. A system for authenticated policy-compliant routing. *ACM SIGCOMM Computer Communication Review* 34 (2004), 167–178.
[70] Yakov Rekhter, Tony Li, and Susan Hares. 2005. *A Border Gateway Protocol 4 (BGP-4)*. Technical Report. T. J. Watson Research Center, IBM Corp.
[71] Gábor Rétvári, János Tapolcai, Attila Kőrösi, András Majdán, and Zalán Heszberger. 2013. Compressing IP forwarding tables: Towards entropy bounds and beyond. In *Proceedings of ACM SIGCOMM 2013*. 111–122.
[72] Jennifer Rexford and Constantine Dovrolis. 2010. Future Internet architecture: Clean-slate versus evolutionary research. *Communications of the ACM* 53, 9 (2010), 36–40.
[73] Leonid Reyzin and Natan Reyzin. 2002. Better than BiBa: Short one-time signatures with fast signing and verifying. In *Information Security and Privacy*. Lecture Notes in Computer Science, Vol. 2384. Springer, 144–153.
[74] Stefan Savage, David Wetherall, Anna Karlin, and Tom Anderson. 2000. Practical network support for IP traceback. *ACM SIGCOMM Computer Communication Review* 30 (2000), 295–306.
[75] Alex C. Snoeren, Craig Partridge, Luis A. Sanchez, Christine E. Jones, Fabrice Tchakountio, Stephen T. Kent, and W. Timothy Strayer. 2001. Hash-based IP traceback. In *Proceedings of ACM SIGCOMM 2001*. 3–14.
[76] Christian Sommer. 2014. Shortest-path queries in static networks. *ACM Computing Surveys* 46, 4 (2014), 45.
[77] Dawn Xiaodong Song and Adrian Perrig. 2001. Advanced and authenticated marking schemes for IP traceback. In *Proceedings of IEEE INFOCOM 2001*, Vol. 2. IEEE, Los Alamitos, CA, 878–886.
[78] Ion Stoica, Daniel Adkins, Shelley Zhuang, Scott Shenker, and Sonesh Surana. 2002. Internet indirection infrastructure. In *Proceedings of ACM SIGCOMM 2002*. 73–86.
[79] Qixiang Sun, Daniel R. Simon, Yi-Min Wang, Wilf Russell, Venkata N. Padmanabhan, and Lili Qiu. 2002. Statistical identification of encrypted web browsing traffic. In *Proceedings of IEEE S&P 2002*. 19–30.
[80] Paul Syverson, R. Dingledine, and N. Mathewson. 2004. Tor: the second-generation onion router. In *Proceedings of USENIX Security 2004*.
[81] Paul F. Syverson, David M. Goldschlag, and Michael G. Reed. 1997. Anonymous connections and onion routing. In *Proceedings of IEEE S&P 1997*. 44–54.
[82] Praveen Tammana, Rachit Agarwal, and Myungjin Lee. 2015. Cherrypick: Tracing packet trajectory in software-defined datacenter networks. In *Proceedings of ACM SOSR 2015*. 1–7.
[83] Brian Trammell, Jean-Pierre Smith, and Adrian Perrig. 2018. Adding path awareness to the Internet architecture. *IEEE Internet Computing* 22, 2 (2018), 96–102.
[84] Michael Walfish, Jeremy Stribling, Maxwell N. Krohn, Hari Balakrishnan, Robert Morris, and Scott Shenker. 2004. Middleboxes no longer considered harmful. In *Proceedings of USENIX OSDI 2004*, Vol. 4. 15.
[85] Dan Wendlandt, Ioannis Avramopoulos, David G. Andersen, and Jennifer Rexford. 2006. Don't secure routing protocols, secure data delivery. In *Proceedings of ACM HotNets 2006*. 7–12.

[86] Edmund L. Wong, Praveen Balasubramanian, Lorenzo Alvisi, Mohamed G. Gouda, and Vitaly Shmatikov. 2007. Truth in advertising: Lightweight verification of route integrity. In *Proceedings of ACM PODC 2007*. 147–156.
[87] Bo Wu, Ke Xu, Qi Li, Zhuotao Liu, Yih-Chun Hu, Martin J. Reed, Meng Shen, and Fan Yang. 2018. Enabling efficient source and path verification via probabilistic packet marking. In *Proceedings of IEEE/ACM IWQoS 2018*. 1–10.
[88] Wen Xu and Jennifer Rexford. 2006. MIRO: Multi-path interdomain routing. In *Proceedings of ACM SIGCOMM 2006*. 171–182.
[89] Abraham Yaar, Adrian Perrig, and Dawn Song. 2005. FIT: Fast Internet traceback. In *Proceedings of IEEE INFOCOM 2005*, Vol. 2. 1395–1406.
[90] Ming-Hour Yang and Ming-Chien Yang. 2012. RIHT: A novel hybrid IP traceback scheme. *IEEE Transactions on Information Forensics and Security* 7, 2 (2012), 789–797.
[91] Xiaowei Yang, David Clark, and Arthur W. Berger. 2007. NIRA: A new inter-domain routing architecture. *IEEE/ACM Transactions on Networking* 15, 4 (2007), 775–788.
[92] Xiaowei Yang and David Wetherall. 2006. Source selectable path diversity via routing deflections. *ACM SIGCOMM Computer Communication Review* 36 (2006), 159–170.
[93] Wen Yao, Chao-Hsien Chu, and Zang Li. 2011. Leveraging complex event processing for smart hospitals using RFID. *Journal of Network and Computer Applications* 34, 3 (2011), 799–810.
[94] Minlan Yu, Jennifer Rexford, Michael J. Freedman, and Jia Wang. 2011. Scalable flow-based networking with DIFANE. In *Proceedings of ACM SIGCOMM 2011*. 351–362.
[95] Shui Yu, Wanlei Zhou, Robin Doss, and Weijia Jia. 2011. Traceback of DDoS attacks using entropy variations. *IEEE Transactions on Parallel and Distributed Systems* 22, 3 (2011), 412–425.
[96] Pamela Zave, Ronaldo A. Ferreira, X. Kelvin Zou, Masaharu Morimoto, and Jennifer Rexford. 2017. Dynamic service chaining with Dysco. In *Proceedings of ACM SIGCOMM 2017*. 57–70.
[97] Fuyuan Zhang, Limin Jia, Cristina Basescu, Tiffany Hyun-Jin Kim, Yih-Chun Hu, and Adrian Perrig. 2014. Mechanized network origin and path authenticity proofs. In *Proceedings of ACM CCS 2014*. 346–357.
[98] Xin Zhang, Hsu-Chun Hsiao, Geoffrey Hasker, Haowen Chan, Adrian Perrig, and David G. Andersen. 2011. SCION: Scalability, control, and isolation on next-generation networks. In *Proceedings of IEEE S&P 2011*. 212–227.
[99] Xin Zhang, Zongwei Zhou, Hsu-Chun Hsiao, Tiffany Hyun-Jin Kim, Adrian Perrig, and Patrick Tague. 2012. ShortMAC: Efficient data-plane fault localization. In *Proceedings of NDSS 2012*.
[100] Wenchao Zhou, Qiong Fei, Arjun Narayan, Andreas Haeberlen, Boon Thau Loo, and Micah Sherr. 2011. Secure network provenance. In *Proceedings of ACM SOSP 2011*. 295–310.
[101] Dapeng Zhu, Mark Gritter, and David R. Cheriton. 2003. Feedback based routing. *ACM SIGCOMM Computer Communication Review* 33, 1 (2003), 71–76.
[102] Earl Zmijewski. 2008. You Can't Get There from Here. Retrieved July 28, 2020 from https://blogs.oracle.com/internetintelligence/you-cant-get-there-from-here?.

Received July 2019; revised May 2020; accepted June 2020