EPIC: Every Packet Is Checked in the Data Plane of a Path-Aware Internet

Markus Legner, Tobias Klenze, Marc Wyss, Christoph Sprenger, and Adrian Perrig
Department of Computer Science, ETH Zurich, Switzerland
{markus.legner, tobias.klenze, marc.wyss, sprenger, adrian.perrig}@inf.ethz.ch

Abstract

An exciting insight of recent networking research has been that path-aware networking architectures are able to fundamentally solve many of the security issues of today's Internet, while increasing overall efficiency and giving control over path selection to end hosts. In this paper, we consider three important issues related to this new networking paradigm: First, network operators still need to be able to impose their own policies to rule out uneconomical paths and to enforce these decisions on the data plane. Second, end hosts should be able to verify that their forwarding decisions are actually followed by the network. Finally, both intermediate routers and recipients should be able to authenticate the source of packets. These properties have been considered by previous work, but there is no existing system that achieves both strong security guarantees *and* high efficiency.

We propose EPIC, a family of data-plane protocols that provide increasingly strong security properties, addressing all three described requirements. The EPIC protocols have significantly lower communication overhead than comparable systems: for realistic path lengths, the overhead is 3–5 times smaller compared to the state-of-the-art systems OPT and ICING. Our prototype implementation is able to saturate a 40 Gbps link even on commodity hardware due to the use of only few highly efficient symmetric cryptographic operations in the forwarding process. Thus, by ensuring that every packet is checked at every hop, we make an important step towards an efficient *and* secure future Internet.

1 Introduction

In the current Internet, end hosts lack control over the paths that their packets take. While they can sometimes select the first hop (using multi-homing) [16], the path beyond it is completely determined by the network. This leads to inefficiencies: The network has to choose paths that balance sometimes conflicting properties such as low latency and high bandwidth. All packets traverse these chosen paths instead of following the routes that best fulfill a particular flow's requirements. The lack of path control also leads to many other problems, such as compliance, when data is not allowed to leave a particular jurisdiction; privacy leaks, when BGP hijacking attacks are used to de-anonymize users [43]; or re-routing attacks being used to obtain fake certificates [10]. Another shortcoming of the current Internet is that there is no way for an end user to verify the *actual* path a packet took on its way to the recipient. While applications such as `traceroute` enable network probing, the obtained information cannot be trusted due to the lack of authentication [2, 4].

Over the past 15 years, different architectures for a new path-aware Internet have been proposed, attempting to give transparency and choices to end hosts [3, 9, 22, 37–39, 46, 47]. Like most modern networking protocols, they are composed of two parts: (i) the low-bandwidth *control plane*, in which neighboring nodes exchange topology and path information, and (ii) the high-bandwidth *data plane*, in which data packets are forwarded across the network along the paths discovered in the control plane. In path-aware networking, paths are exposed to hosts, allowing them to embed a path of their choice in the header of their packets (*packet-carried forwarding state*). The data plane then ensures that packets traverse the network along their source-selected paths. However, to balance control with autonomous systems (ASes), such as Internet service and transit providers, end hosts cannot use arbitrary paths but need to choose from a set of preselected paths created by the ASes. This restriction, which we call *path authorization*, serves both the economic interests of ASes and network availability, for instance by ruling out forwarding loops. It can be enforced either by storing allowed paths on each border router [22, 46], or by cryptographically securing the publicly distributed path information and verifying it during forwarding [3, 9, 36–39]. Stateful solutions scale poorly to the inter-domain context, can suffer from inconsistencies across distributed nodes, and are less efficient than cryptographic solutions [28]. We thus focus on systems that use cryptographic authenticators for each (AS-level) hop in the header of packets. In existing systems, fixing the length of these authenticators poses a dilemma: sufficiently long authenticators cause an unacceptably high communication overhead, but short and efficient authenticators are insecure: An attacker can conduct an online brute-force attack, i.e., send packets with fabricated authenticators between two hosts under his control until a packet is successfully forwarded. So far, there is no solution that is both efficient and secure.

Parallel to the development of next-generation Internet architectures, recognition grew that end hosts and routers need to authenticate the source and contents of packets (*source and packet authentication*), and that the source and destination need to be able to reconstruct and validate a packet's actual path (*path validation*) [11–13, 15, 28, 36, 45]. A prime application of source authentication is defending against denial-of-service (DoS) attacks, in which network links or end hosts are flooded with excessive amounts of traffic. These attacks are often enabled by the attacker's ability to spoof its own address; source authentication at network routers protects both the network and the destination by filtering unauthentic packets early and before they reach any bottleneck links. In addition, more sophisticated DoS-defense mechanisms such as bandwidth-reservation systems fundamentally depend on an efficient source-authentication mechanism [6]. On the other hand, path validation protects the path choices made by the source of packets; if messages need to follow a specific path due to, e.g., compliance reasons, it is crucial that end hosts can check whether their path directive is actually obeyed by on-path routers. Also, in a path-aware Internet, end hosts may be able to choose between several paths of different properties and costs; if using a more expensive path (e.g., through a satellite network), end hosts have a legitimate interest in obtaining proof that this path was actually traversed.

While solutions exist that provide source authentication and path validation, they come with significant communication and computation overhead: ICING [36] and OPT [28, 37] have an overhead of hundreds of bytes per packet for realistic path lengths. A recent proposal, PPV [45], reduces the overhead and reaches practically feasible efficiency, but only verifies individual links on the path probabilistically and only enables source authentication for the destination. However, as described above, this is insufficient for effectively defending against DoS attacks; for this application it is necessary that *every packet* be checked at *every hop*.

In this paper, we propose EPIC, a family of cryptographic data-plane protocols with increasingly strong security properties, including path authorization, source authentication, and path validation. The key insight of our protocols is how they escape the dilemma between low communication overhead and security: On the one hand, we use relatively short per-hop authentication fields to limit communication overhead. On the other hand, we ensure that even strong attackers, with the ability to forge a single one of these fields by sending a large number of packets, cannot cause significant damage. We achieve this in two ways: First, by binding an authenticator to a specific packet, EPIC ensures that a forged authenticator does not allow an attacker to send additional packets and thus prevents volumetric DoS attacks. Second, EPIC uses a longer authentication field for the destination which is unforgeable for even strong attackers, such that the very few packets that were able to deceive intermediate routers are detectable at the destination. As a result of short per-hop authenticators, EPIC has substantially lower communication overhead, which scales better with the path length than state-of-the-art protocols like ICING [36] and OPT [28].

Our main contributions are the following:

- We propose EPIC, a series of protocols that use unique authenticators for each packet to resolve the security–efficiency dilemma in the data plane of path-aware Internet architectures.
- We propose a new attacker model that combines a localized Dolev–Yao [17] adversary with a cryptographic oracle. This allows us to express EPIC's resilience against even powerful attackers. EPIC achieves all desirable security goals in this stronger attacker model.
- We show that EPIC has a communication overhead that is 3–5 times smaller compared to the state-of-the-art solutions OPT and ICING for realistic path lengths.
- We implement EPIC with Intel's Data Plane Development Kit (DPDK) [18] and show that our router implementation running on commodity hardware can saturate a 40 Gbps link while using only four processing cores.

2 Problem Definition

We target the problem of securing the inter-domain data-plane of path-aware Internet architectures. The Internet is a network of networks, which are commonly called autonomous systems (ASes). Each AS has centralized control within its own network, which simplifies managing and securing the communication, e.g., through software-defined networking [8]. By contrast, networking between ASes requires coordination between separate entities without central control. Our work focuses on securing this inter-AS communication. We exclude the equally important, but orthogonal problem of securing intra-AS networking. Thus, in line with previous work [35], we abstract from the internal networks of ASes and consider all security properties at the level of ASes (or the end hosts that connect to them); in particular, throughout the remainder of this paper, "hop" stands for "AS-level hop".

We also only focus on securing the data plane. We assume that the control plane is secure and constructs paths according to the ASes' policies and the participants of our data-plane protocols obtain the required symmetric keys and path information via secure control-plane channels. While securing key distribution and other control-plane functionality is itself a challenging task, it is orthogonal to the challenges for the data plane: as we argue below, in the control plane, asymmetric cryptography can be used to provide strong security guarantees, whereas in the data plane only symmetric cryptography is sufficiently efficient. In practice, Internet architectures implement a public-key infrastructure to secure control-plane operations, such as the Resource Public Key Infrastructure (RPKI) of today's Internet [30].

In our security analysis (§5), we analyze our protocols with respect to a localized variant of the Dolev–Yao attacker that fully controls some ASes. EPIC protects the interests of both end hosts and ASes against such attackers; specifically the properties that we present in this section.

2.1 Security Requirements for End Hosts

Based on the motivation provided in the introduction, we consider two fundamental security properties for end hosts: *path validation* and *packet authentication*.

While path control—provided by the underlying path-aware Internet architecture—allows sources to select a forwarding path, it is by itself insufficient to protect the security and privacy interests of end hosts as it does not provide any guarantees that the directives are actually obeyed. We aim to additionally achieve *path validation*, i.e., enabling the destination of a packet to verify that the actually traversed path of the packet matches the path intended by the sender and allowing the source to also verify this proof.

Packet authentication provides proof of a packet's origin and content to the destination, preventing source-spoofing or packet modification that are possible in today's Internet.

The authentication and path-validation properties for end hosts presented in this work require source and destination hosts to trust both of their ASes due to the key-distribution mechanisms—a trust assumption also found in similar schemes [28, 37]. Due to this additional trust assumption, network-level authentication does not replace the security offered by higher-layer protocols such as TLS. At the same time, higher-level authentication is not a replacement for network-layer authentication: network-layer schemes can be used for packet filtering that sets in *prior* to stateful TCP and TLS handshakes, and is thus highly efficient.

2.2 Security Requirements for ASes

For ASes, we consider two important security properties: *path authorization* and *source authentication*.

Each AS is driven by its own economic interests, which gives rise to path policies that collectively define a set of authorized paths. The control plane is responsible for authentically and efficiently discovering and distributing these paths (see §3.1) and ensuring that they do not contain loops and fulfill the policies of ASes. However, a secure control plane cannot substitute a secure data plane: the data plane needs to provide *path authorization*, i.e., enforce the decisions that ASes make in the control plane for data traffic. Path authorization ensures that malicious end hosts cannot create packets that will be forwarded along unauthorized paths.

In many DoS attacks on the current Internet, the attacker spoofs the origin of attack traffic. *Source authentication* ensures that routers can validate the origin of each packet, thus ruling out source-spoofing attacks. While some protocols (e.g., IPSec) enable source authentication, they typically only filter traffic at the destination. Dropping malicious traffic early is not only more efficient than destination filtering, it also protects against DoS attacks that target the networking infrastructure itself [26, 42], rather than an end host: source authentication by routers ensures that traffic is filtered before any bottleneck links are reached. Furthermore, sophisticated DoS-defense schemes such as bandwidth-reservation systems [6] rely on source authentication to prevent adversaries from using up reserved bandwidth of honest sources.

2.3 Efficiency Requirements

The need to keep ever-growing forwarding tables on routers of the current Internet requires expensive and energy-intensive hardware and fundamentally limits its scalability. It is therefore essential that a future Internet minimizes router state.

The data plane must also have low communication and computation overhead and minimize additional latency during setup and processing. A simple calculation underscores this: Consider 400 Gbps links, which are currently being deployed in the Internet, and 500 B packets. To saturate the link, a router needs to process one packet every 10 ns. Even taking into account pipelining and parallelism, this shows that packet processing in the data plane must proceed within hundreds of nanoseconds—ruling out any asymmetric cryptography, which requires several microseconds for a single operation [19]. In contrast, block ciphers with hardware acceleration such as AES can be computed within tens of nanoseconds and are suitable to use in the data plane [14, 23].

3 Background and Definitions

To provide the necessary context for constructing our data-plane protocols, we sketch out an abstract path-aware control plane, in particular the path-exploration and -registration mechanisms. This description is based on SCION's control plane [37] but abstracts from many low-level details. We postpone the discussion of how EPIC can be integrated into real architectures to §7. Table 1 on the next page summarizes the notation and acronyms.

3.1 Path Exploration and Registration

While we consider paths at an AS-level granularity, we do include the ingress and egress *interface IDs* of each (AS-level) hop. Each AS is free to assign these identifiers to its external connections without coordination with other ASes. The interface IDs are recorded in the control plane and later used for packet forwarding in the data plane. To discover paths between any pair of ASes, each AS periodically initiates path exploration by sending *beacons* to their neighboring ASes. An AS can decide which paths to authorize by forwarding the authenticated beacons to neighbors and registering them at public *path servers*.

A beacon is initialized with an absolute timestamp TS_{path}. An AS disseminating it adds its own *hop information* (*HI*), which is used in the data plane as a forwarding directive; a cryptographic token σ called the *hop authenticator*, which allows the data-plane routers to verify the correctness of the hop information; and a signature, which protects the beacon's authenticity in the control plane and is removed when beacons are turned into data-plane paths by end hosts. HI consists of an expiration time ts_{exp} relative to the beacon's timestamp; the ingress interface, through which the beacon was received; and the egress interface, through which it is forwarded.

A crucial observation is that ASes can make decisions during path exploration about which paths to authorize based on their own economic interests. To that end, ASes can inspect the complete upstream path and only forward beacons that do not contain loops and are consistent with their path policies to their customers. Path authorization for some AS A, which we achieve with our data-plane protocols, builds on the hop authenticator σ_A: this cryptographic tag is calculated using a symmetric secret key K_A (which is shared only among networking entities within A) and can include the upstream path in addition to A's own hop information.

3.2 Path Construction and Forwarding

To simplify the presentation, we assume that packets are always forwarded in opposite direction of beaconing. To create a forwarding path, an end host H_S queries its local path server (located in the same AS) for beacons from the intended destination AS A_ℓ to his own AS A_1. H_S selects a beacon from those offered by the path server, and verifies its signatures. By removing the signatures from the beacon, the beacon is turned into a path that can be directly embedded into the packet. A data-plane packet thus contains the entire forwarding state. For a path from A_1 to A_ℓ it has the format

$$\textsc{Packet} := (\textsc{Path} \parallel \textsc{ValHd} \parallel P), \tag{1a}$$

$$\textsc{Path} := \big(TS_{\text{path}} \parallel \textsc{Src} \parallel \textsc{Dest} \parallel HI_1 \parallel \cdots \parallel HI_\ell\big), \tag{1b}$$

$$\textsc{ValHd} := \big(ts_{\text{pkt}} \parallel S_1 \parallel V_1 \parallel \cdots \parallel S_\ell \parallel V_\ell \parallel V_{\text{SD}}\big), \tag{1c}$$

where P denotes the packet's payload, $\textsc{Src} := (A_1{:}H_S)$ denotes the source, and $\textsc{Dest} := (A_\ell{:}H_D)$ denotes the destination. $\textsc{ValHd}$ contains fields necessary for verifying the packet: The timestamp ts_{pkt} indicates the time at which the packet is sent relative to TS_{path} and is used to provide freshness. The *segment identifier* S_i is a cryptographic code based on the hop authenticator σ_i used for path authorization. It can be created from σ_i and uniquely identifies the portion of the path in between the beacon initiator A_ℓ and A_i. The *hop validation fields* (HVFs) V_i are cryptographic tags that are filled in by the source host and allow intermediate routers to validate the packet. An additional *destination validation field* V_{SD} allows the destination to validate the path of the packet. The definitions of S_i, V_i, and V_{SD} will be presented in §4.

The term *hop field* refers to a triple consisting of hop information HI_i, segment identifier S_i, and HVF V_i. We define the *packet origin* as the triple of source, path timestamp, and

Table 1: Summary of abbreviations and notation.

A_i	AS corresponding to the ith hop on the path; H_S and H_D are located in A_1 and A_ℓ, respectively
C_i	cryptographic result used for authenticating and updating the ith hop fields
H_S, H_D	source and destination hosts of a packet
HI_i	ith hop information consisting of ts_{exp}, ingress interface, and egress interface
K_i	secret symmetric key of A_i
K_i^S	host key shared between A_i, A_1, and H_S, which can be efficiently calculated by A_i
K_{SD}	key shared between A_1, H_S, A_ℓ, and H_D
ℓ	AS-level path length
l_{val}, l_{seg}	length in bytes of V_i, S_i
l_{PRF}	block size in bytes of $\mathsf{PRF}(\cdot)$ and $\mathsf{MAC}(\cdot)$
$\mathsf{MAC}_K(\cdot)$	message authentication code using key K
P, $p = \lvert P \rvert$	packet payload and payload size
PO	packet origin consisting of $\textsc{Src}$, TS_{path}, and ts_{pkt}
$\mathsf{PRF}_K(\cdot)$	pseudorandom function using key K
$S_i^{(l)}$	segment identifier in protocol level l allowing ASes to chain hops to paths
$\sigma_j^{(l)}$	hop authenticator in level l authorizing the jth hop as calculated by A_j during path exploration
$\textsc{Src}$	$(A_1 \parallel H_S)$; source AS and host address
TS_{path}	path timestamp created during path exploration
ts_{exp}	expiration time of a hop field relative to TS_{path}
ts_{pkt}	packet creation time relative to TS_{path}
$V_{i;j}^{(l)}$, HVF	hop validation field in protocol level l corresponding to the ith hop after processing by A_j; when its value stays constant, we omit j.
V_{SD}	destination validation field
$X[\![i{:}j]\!]$	substring from byte i (incl.) to byte j (excl.) of X
$\parallel$	concatenation of strings

packet timestamp,

$$PO := (\textsc{Src}, TS_{\text{path}}, ts_{\text{pkt}}). \tag{2}$$

As forwarding information is included in the packet header, intermediate routers at the border of an AS can directly use this (after cryptographically validating it) and do not need to keep forwarding tables. In case of a link failure, an end host can be notified and immediately switch to a backup path without needing to wait for the network to reconverge.

3.3 Notation

We denote the application of a pseudorandom function (PRF) and the computation of a message authentication code (MAC) with key K as $\mathsf{PRF}_K(\cdot)$ and $\mathsf{MAC}_K(\cdot)$, respectively. For MACs, we assume that they also provide the properties of a PRF. We write l_{val} and l_{seg} for the lengths in bytes of the hop validation fields and the segment identifiers, respectively. The block size of PRFs and MACs in bytes is denoted by l_{PRF}, where $l_{\text{PRF}} = 16$ for AES. In some protocols, HVFs are updated by intermediate routers; in this case, we write $V_{i;j}$ for the HVF corresponding to A_i after processing by A_j

and use $V_{i;0}$ for their initial values. We use superscripts to distinguish the different EPIC protocols, named L0–L3, e.g., $V_i^{(0)}, \dots V_i^{(3)}$. Concatenation of (binary) strings is denoted by $\|$, and $X[\![i{:}j]\!]$ is the substring from byte i (inclusive) to byte j (exclusive) of X. Table 1 summarizes our notation.

3.4 Global Symmetric-Key Distribution

Some of the protocols that we propose require the source host to create authenticators for each packet that either the destination or intermediate routers verify. While asymmetric cryptography scales well in the number of networking entities, the computation overhead of a per-packet usage is prohibitive as shown in §2.3. On the other hand, the standard use of symmetric cryptography would require routers to store a key for each packet source, which is infeasible on core routers in the Internet. In order to be able to use symmetric cryptography without per-host state on intermediate routers, we leverage the dynamically-recreatable-key (DRKey) / PISKES system [28, 40], which we will summarize in this section.

With DRKey, one party, e.g., a router in an AS A, can derive symmetric keys by simply applying PRFs to deterministic inputs, while the other party has to fetch keys from a key server (over a secure control-plane channel). DRKey defines AS-level keys shared between ASes A and B:

$$K_{A\to B} := \mathsf{PRF}_{K_A}(B)\,. \tag{3}$$

Here, K_A is a secret key of the AS A, which is shared between all its (border) routers and key servers but with no external entities, and B is a unique and public identifier of AS B. The arrow in the derived key indicates the asymmetry between A and B: AS A is able to quickly derive the keys on the fly using symmetric cryptography, while AS B needs to fetch the key $K_{A\to B}$ by an explicit request to A's key server, protected by asymmetric cryptography. DRKeys are valid for time periods on the order of one day, such that these key requests happen relatively infrequently.

Given an AS-level key, host-level keys can be derived by another application of a PRF:

$$K_{A\to B:H_B} := \mathsf{PRF}_{K_{A\to B}}(H_B)\,, \tag{4a}$$
$$K_{A:H_A\to B:H_B} := \mathsf{PRF}_{K_{A\to B}}(H_A \,\|\, H_B)\,. \tag{4b}$$

An end host H_B in AS B can query the key servers of B in order to obtain the keys (4a) or (4b), which can be calculated by B from the AS-level key (3). These keys are shared between all entities in the subscripts, e.g., $K_{A_\ell:H_D\to A_1:H_S}$ is shared among A_ℓ, H_D, A_1, and H_S. Therefore, when authenticating sources using DRKey, no end-host-to-end-host guarantees are obtained: A malicious AS A_1 could claim that a packet originating from H_S came from a different host H_S' in A_1. The destination host H_D in AS A_ℓ can only authenticate the source host under the assumption that A_ℓ is honest. As discussed above, these are common restrictions in order to accommodate the efficiency requirements of high-speed routers. As we discuss in §6.4, using DRKeys introduces little communication overhead and negligible additional latency.

Other AS-level key-establishment systems could be used for exchanging AS-level symmetric keys. For example, Passport establishes symmetric keys $K_{A\leftrightarrow B}$ between any pair of ASes by means of a Diffie–Hellman key exchange on top of BGP announcements [31]. These keys can be used in place of $K_{A\to B}$ in Eq. (4) but require also to input the AS identifier in order to distinguish $K_{A:H\leftrightarrow B}$ from $K_{A\leftrightarrow B:H}$. Furthermore, as they cannot be recreated on the fly at border routers, a router would need to cache a symmetric key to every other AS.

Irrespective of the system used to exchange AS-level keys, the communication between end hosts and key servers relies on secure control-plane channels in order to prevent malicious entities impersonating key servers or discovering keys. As we explained above, this is an orthogonal problem to securing the data plane, and thus outside the scope of this work.

4 EPIC Protocols

In this section, we develop three protocol levels 1–3 of EPIC with increasingly strong security properties. We present the protocols in a step-by-step development, thus explaining for each security property the mechanism and prerequisites to achieve it. As a starting point, we begin by describing a simple protocol (referred to as "EPIC L0") that represents the approach taken in the current SCION data plane (with minor simplifications) [37]. Its primary security property is path authorization, which protects ASes from malicious sources using paths that violate their routing policies.

4.1 Level 0: Path Authorization

EPIC L0 achieves path authorization using static MACs that are calculated during path exploration and directly serve as HVFs for forwarding. During the path-exploration process, an AS A calculates the hop authenticator $\sigma^{(0)}$ as a MAC over the beacon's timestamp, the hop information, and the previous hop authenticator ($\sigma^{(0)\prime}$), truncated to l_{val} bytes:

$$\sigma_A^{(0)} := \mathsf{MAC}_{K_A}\left(TS_{\text{path}} \,\|\, HI_A \,\|\, \sigma^{(0)\prime}\right)[\![0{:}l_{\text{val}}]\!]. \tag{5}$$

For the AS initiating the beacon, there is no previous hop authenticator, so $\sigma^{(0)\prime}$ is not included.

This hop authenticator directly serves as the HVF in the data plane, $V_i^{(0)} := \sigma_i^{(0)}$; segment identifiers and additional header fields ts_{pkt} and V_{SD} as defined in Eq. (1) are therefore unused in EPIC L0. The procedure to create and forward packets is the following:

Source H_S obtains a path, including all hop authenticators, from the path server in its AS. It constructs the packet according to Eq. (1) by copying the path timestamp and the hop information and hop authenticator for each hop.

Transit At every AS A_i, the border router first checks that the packet was received through the correct interface according to HI_i and that the hop field is not expired.

Then the router recalculates $V_i^{(0)} = \sigma_i^{(0)}$ according to Eq. (5) and checks that it coincides with the HVF in the packet header. If the packet passes both checks, the router forwards it to the next hop specified in HI_i, otherwise it drops the packet. The only state required on AS border routers is the AS' secret key K_A and intra-AS forwarding information.

The construction presented here ensures that end hosts and ASes can only send packets on paths that are authorized by *all* on-path ASes. Chaining hops by including the hop authenticator of the previous hop in the MAC calculation defined in Eq. (5) guarantees that *complete paths* are authorized and hosts cannot combine individual hops arbitrarily.

4.2 Level 1: Improved Path Authorization

EPIC L0 suffers from a dilemma between secure hop fields and acceptable communication overhead: Assuming short hop authenticators with $l_{\text{val}} = 3$ (the default length of hop authenticators in SCION [37]), these fields are susceptible to online brute-force attacks. An attacker has to send at most $2^{24} \approx 1.6 \cdot 10^7$ probe packets to find a correct MAC of an unauthorized hop, which takes under 10 seconds on a gigabit link. Afterwards, the attacker can use the unauthorized hop field to send arbitrary traffic until the eventual expiration of the path. MACs can be made longer and thus harder to forge, but only at the expense of increased communication overhead, see §6.3. The fundamental problem is that the static HVFs can be directly reused to send additional packets.

With EPIC L1 we resolve this dilemma by replacing these static hop authenticators by *per-packet* HVFs that cannot be reused for additional packets. During path exploration, an AS A calculates its hop authenticator σ_A as follows:

$$\sigma_A^{(1)} := \mathsf{MAC}_{K_A}\left(TS_{\text{path}} \parallel HI_A \parallel S^{(1)'}\right). \tag{6}$$

Here, $S^{(1)'}$ is the segment identifier of the previous hop during the path exploration, which is obtained by simply truncating the hop authenticator:

$$S^{(1)} := \sigma^{(1)} [\![0{:}l_{\text{seg}}]\!]. \tag{7}$$

The hop authenticator is then subsequently used by the source host to calculate the per-packet HVFs:

$$V_i^{(1)} := \mathsf{MAC}_{\sigma_i}(ts_{\text{pkt}} \parallel \textsc{Src}) [\![0{:}l_{\text{val}}]\!]. \tag{8}$$

As the hop authenticators are not part of the packet header to limit communication overhead, the additional segment identifiers are required for chaining hops as they allow ASes to derive the hop authenticators on the fly. The aim of EPIC L2 is improving path authorization, the field V_{SD} is thus not used. An attacker trying to forge an unauthorized path needs to find at least one HVF that fulfills Eq. (8) without access to σ_i by sending a large number of probing packets. However, in contrast to EPIC L0, this HVF cannot be used to send additional packets, which carry different packet timestamps.

Even though each HVF is only valid for a specific packet origin, an attacker could launch a DoS attack by replaying packets for which he knows the HVFs or simply reusing the packet timestamp. From L1 onwards, we employ a *replay-suppression system* in border routers or inside an AS' network to prevent this [29]. This system tracks and uniquely identifies packets based on the packet origin PO, i.e., source, path timestamp, and the packet timestamp, see Eq. (2). In order for the packet origin to serve as a unique packet identifier, the packet timestamp must be sufficiently long, see §7 for a more detailed discussion. The replay-suppression system uses Bloom filters to identify duplicates but discards old packets in order to make this process viable in high-bandwidth networking applications, see §6.4. Note that packets are processed by the replay-suppression system *after* being authenticated in order to prevent an attacker from poisoning the system with unauthentic packets.

The procedure to create and forward packets is slightly more complicated than for EPIC L0:

Source H_S obtains the desired path including all hop authenticators from its path server. H_S calculates the packet timestamp ts_{pkt} and adds it to the header. The host then calculates the segment identifiers according to Eq. (7) and HVFs according to Eq. (8) and constructs all hop fields consisting of HI_i, $S_i^{(1)}$, and $V_i^{(1)}$.

Transit An AS checks the interfaces and expiration in the same way as in EPIC L0. It recalculates first the hop authenticator as in Eq. (6) using the previous hop's segment identifier (in construction direction) and then its own segment identifier according to Eq. (7) and the HVF as in Eq. (8). If interfaces, segment identifier, and HVF are all correct, and the timestamp is current, the AS forwards the packet, otherwise it drops it.

4.3 Level 2: Authentication

We now extend the previous protocol by a mechanism to allow intermediate routers to authenticate the source of a packet and the destination to additionally authenticate its payload. The hop authenticators, segment identifiers, and the additional header field ts_{pkt} are unchanged. We define the *host keys*

$$K_i^{\text{S}} := K_{A_i \to A_1:H_{\text{S}}} \tag{9a}$$

for every on-path AS A_i and an additional key

$$K_{\text{SD}} := K_{A_\ell:H_{\text{D}} \to A_1:H_{\text{S}}} \tag{9b}$$

shared between source and destination. These keys are based on the derivation defined in Eq. (4) and can be used to provide path authorization and source authentication in a single HVF:

$$V_i^{(2)} := \mathsf{MAC}_{K_i^{\text{S}}}(ts_{\text{pkt}} \parallel \textsc{Src} \parallel \sigma_i) [\![0{:}l_{\text{val}}]\!]. \tag{10}$$

The destination host H_{D} can authenticate the source of the packet and verify that neither the path (as defined in Eq. (1b)) nor the payload was modified through the additional destination validation field

$$V_{\mathrm{SD}}^{(2)} := \mathsf{MAC}_{K_{\mathrm{SD}}}(ts_{\mathrm{pkt}} \parallel \textsc{Path} \parallel P)\,. \tag{11}$$

The procedure to create and forward packets is as follows:

Source In addition to EPIC L1, the source H_{S} fetches all necessary host keys from the local key server and subsequently calculates the HVFs according to Eq. (10) as well as V_{SD} according to Eq. (11).

Transit In addition to the checks in EPIC L1, every AS needs to recalculate the host key K_i^{S} according to Eqs. (3), (4a), and (9a) and then check if the HVF in the packet header satisfies Eq. (10). As all keys can be locally calculated, no key fetching or per-host state is necessary.

Destination H_{D} obtains the key K_{SD} from its local key server and validates $V_{\mathrm{SD}}^{(2)}$ according to Eq. (11).

4.4 Level 3: End-Host Path Validation

EPIC L3 further extends the security properties of EPIC L2 by enabling the source and destination of a packet to perform path validation. To that end, on-path ASes *overwrite* their HVFs with proofs to the source and destination that they have processed the packet. Upon receiving the packet, the destination can directly validate the path based on the destination validation field and enables path validation for the source by replying with a confirmation message. We define

$$C_i := \mathsf{MAC}_{K_i^{\mathrm{S}}}(ts_{\mathrm{pkt}} \parallel \textsc{Src} \parallel \sigma_i)\,, \tag{12}$$

which is equal to Eq. (10) without truncation. This cryptographic result has a length of l_{PRF} bytes, which is generally longer than the HVFs that we propose in this work. This allows us to split the result into multiple separate pieces, which are uncorrelated as we assume the MAC to be a PRF; in particular, under the assumption $l_{\mathrm{PRF}} \geq 2 \cdot l_{\mathrm{val}}$, we can define

$$C_i^{[1]} := C_i[\![0{:}l_{\mathrm{val}}]\!], \quad C_i^{[2]} := C_i[\![l_{\mathrm{val}}{:}2l_{\mathrm{val}}]\!]. \tag{13}$$

The source then performs the same setup as for EPIC L2, setting each HVF to $V_{i;0}^{(3)} := C_i^{[1]}$ (which equals $V_i^{(2)}$). The router in A_i calculates the C_i defined in Eq. (12) and checks that the HVF is correct. Finally, it updates the HVF with $V_{i;i}^{(3)} := C_i^{[2]}$. Without requiring any additional cryptographic computation, the router thus leaves a confirmation for H_{S} that it successfully validated and forwarded the packet (assuming that A_1 is honest), since only A_i, H_{S}, and A_1 can compute $C_i^{[2]}$. We allow H_{D} to also validate this confirmation (under the further assumption that H_{S} and A_ℓ are honest) by including the correct final values $V_{i;\ell}^{(3)}$ in the destination validation field:

$$V_{\mathrm{SD}}^{(3)} := \mathsf{MAC}_{K_{\mathrm{SD}}}\left(ts_{\mathrm{pkt}} \parallel \textsc{Path} \parallel V_{1;\ell}^{(3)} \parallel \cdots \parallel V_{\ell;\ell}^{(3)} \parallel P\right). \tag{14}$$

Algorithm 1 Initialization and path validation at H_{S} in EPIC L3. The second procedure is executed upon receiving a reply packet that contains the path validation proof for the source. Packet contents such as header fields are denoted by $\boxed{\textsc{Field}}$ and $\leftarrow$ is an initialization or assignment. For readability, some superscripts omitted.

```
1: procedure Initialization by $H_{\mathrm{S}}$
Require: $\textsc{Src}, \textsc{Dest}, TS_{\mathrm{path}}, K_{\mathrm{SD}}, P, \forall i \in \{1,\ldots,\ell\}: HI_i, \sigma_i, K_i^{\mathrm{S}}$
2:     construct $\boxed{\textsc{Path}}$ according to Eq. (1b)
3:     $\boxed{ts_{\mathrm{pkt}}} \leftarrow$ (current time) $- TS_{\mathrm{path}}$
4:     for all $i \in \{1,\ldots,\ell\}$ do
5:         $\boxed{S_i} \leftarrow \sigma_i[\![0{:}l_{\mathrm{seg}}]\!]$            ▷ segment identifier (Eq. (7))
6:         $C_i \leftarrow \mathsf{MAC}_{K_i^{\mathrm{S}}}\left(\boxed{ts_{\mathrm{pkt}}} \parallel \boxed{\textsc{Src}} \parallel \sigma_i\right)$
7:         $C_i^{[1]} \leftarrow C_i[\![0{:}l_{\mathrm{val}}]\!]; C_i^{[2]} \leftarrow C_i[\![l_{\mathrm{val}}{:}2l_{\mathrm{val}}]\!]$
8:         $\boxed{V_i} \leftarrow C_i^{[1]}$            ▷ initial value of HVF
9:         $V_{i;\ell} \leftarrow C_i^{[2]}$            ▷ final value of HVF
10:    $\boxed{V_{\mathrm{SD}}} \leftarrow \mathsf{MAC}_{K_{\mathrm{SD}}}\left(\boxed{ts_{\mathrm{pkt}}} \parallel \boxed{\textsc{Path}} \parallel V_{1;\ell} \parallel \ldots \parallel V_{\ell;\ell} \parallel P\right)$
11:    send $\boxed{\textsc{Packet}}$ according to Eq. (1)
12:    store $V_{i;\ell}$ for all $i$ under key $(TS_{\mathrm{path}} \parallel ts_{\mathrm{pkt}})$ for validation

13: procedure Validation at $H_{\mathrm{S}}$
Require: $K_{\mathrm{SD}}$
14:    receive EPIC L2 packet with payload $\boxed{TS_{\mathrm{path}}}$, $\boxed{ts_{\mathrm{pkt}}}$, and $\boxed{V_1} \ldots \boxed{V_\ell}$
15:    if EPIC L2 verification failed then
16:        return "validation failed"
17:    if $\left(\boxed{TS_{\mathrm{path}}} \parallel \boxed{ts_{\mathrm{pkt}}}\right)$ is not a valid key in store then
18:        return "validation failed"
19:    retrieve $V_{i;\ell}$ for all $i$ under key $\left(\boxed{TS_{\mathrm{path}}} \parallel \boxed{ts_{\mathrm{pkt}}}\right)$
20:    for all $i \in \{1,\ldots,\ell\}$ do
21:        if $\boxed{V_i} \neq V_{i;\ell}$ then
22:            return "validation failed"
23:    return "validation succeeded"
```

Note that, as each HVF is only updated once, we have $V_{i;\ell}^{(3)} = V_{i;i}^{(3)}$. In order to allow H_{S} to validate the path, H_{D} needs to send a confirmation message containing the timestamps of the original message together with the updated values $V_{i;\ell}^{(3)}$. To prevent circular confirmations, such a message must be sent to H_{S} as an EPIC L2 packet (cf. §7). To validate the path, H_{S} must store the expected HVFs upon sending a packet. When it receives a reply by the destination that contains the values $V_{i;\ell}^{(3)}$ that the destination received, it validates them against the stored values. If no correct confirmation is received after some timeout, the source can conclude that the original packet has been lost or redirected.

Both source and destination host are free in their reaction to failed path validation: The destination can choose to ignore it and rely on the source to take appropriate action (soft fail) or reject the corresponding packets (hard fail). The source can switch paths on a short timescale and, in case of frequent failures, switch its Internet provider. Note that fault localization in general is a very complex problem and cannot be achieved through EPIC alone in an adversarial environment [5].

The algorithms for initialization, validation, and update in EPIC L3 are shown in Algorithms 1–3. These algorithms do not include the replay-suppression system, which we assume is an external system in each AS that inspects the packet origin of all authenticated packets and eliminates any duplicates.

Algorithm 2 Packet validation and updates at intermediate routers in EPIC L3. Syntax as in Algorithm 1.

```
 1: procedure FORWARDING BY $A_i$
Require: $K_i$
 2:     if $\boxed{HI_i}$ is expired or packet received through wrong interface then
 3:         drop packet
 4:     if (current time) $- \boxed{TS_\text{path}} - \boxed{ts_\text{pkt}} \notin [-\varepsilon, L+\varepsilon]$ then
 5:         drop packet          ▷ invalid timestamp (lifetime $L$, clock skew $\varepsilon$)
 6:     $(A_1 : H_\text{S}) \leftarrow \boxed{\text{SRC}}$
 7:     $K_{A_i \to A_1} \leftarrow \mathsf{PRF}_{K_i}(A_1)$          ▷ derive AS-level DRKey (Eq. (3))
 8:     $K_i^\text{S} \leftarrow \mathsf{PRF}_{K_{A_i \to A_1}}(H_\text{S})$          ▷ derive host-level DRKey (Eq. (9))
 9:     $\sigma_i \leftarrow \mathsf{MAC}_{K_i}\left(\boxed{TS_\text{path}} \parallel \boxed{HI_i} \parallel \boxed{S_i\text{'}}\right)$          ▷ hop authenticator (Eq. (6))
10:     if $\boxed{S_i} \neq \sigma_i[\![0{:}l_\text{seg}]\!]$ then          ▷ check segment identifier (Eq. (7))
11:         drop packet
12:     $C_i \leftarrow \mathsf{MAC}_{K_i^\text{S}}\left(\boxed{ts_\text{pkt}} \parallel \boxed{\text{SRC}} \parallel \sigma_i\right)$
13:     $C_i^{[1]} \leftarrow C_i[\![0{:}l_\text{val}]\!]$; $C_i^{[2]} \leftarrow C_i[\![l_\text{val}{:}2l_\text{val}]\!]$
14:     if $\boxed{V_i} \neq C_i^{[1]}$ then          ▷ authenticate packet
15:         drop packet
16:     $\boxed{V_i} \leftarrow C_i^{[2]}$          ▷ update HVF
17:     forward packet according to $\boxed{HI_i}$
```

Algorithm 3 Packet and path validation at H_D in EPIC L3. Syntax as in Algorithm 1.

```
 1: procedure VALIDATION AND REPLY AT $H_\text{D}$
Require: $K_\text{SD}$
 2:     $V_\text{SD}' \leftarrow \mathsf{MAC}_{K_\text{SD}}\left(\boxed{ts_\text{pkt}} \parallel \boxed{\text{PATH}} \parallel \boxed{V_1} \parallel \ldots \parallel \boxed{V_\ell} \parallel P\right)$
 3:     if $\boxed{V_\text{SD}} \neq V_\text{SD}'$ then
 4:         return "validation failed"
 5:     if (current time) - $\boxed{TS_\text{path}}$ - $\boxed{ts_\text{pkt}} \notin [-\varepsilon, L+\varepsilon]$ then
 6:         return "validation failed"          ▷ timestamp expired or in the future
 7:     send EPIC L2 packet to $H_\text{S}$ with payload          ▷ values of original packet
 8:         $(TS_\text{path} \parallel ts_\text{pkt} \parallel V_1 \parallel \ldots \parallel V_\ell)$
 9:     return "validation succeeded"
```

Algorithms 2 and 3 both enforce the validity of the absolute timestamp $TS_\text{path} + ts_\text{pkt}$: the packet must neither exceed a fixed lifetime L nor must this timestamp lie in the future. These checks take into account a maximum clock skew of ε.

5 Security Analysis

In this section we define the security properties in turn and compare our protocols with ICING [36], OPT [28], and PPV [45]. An overview is shown in Table 2.

5.1 Basic and Strong Attacker Models

Basic-Attacker Model A Dolev–Yao adversary can typically observe, drop, inject, replay, or alter packets anywhere in the network [17]. However, if an attacker can re-route packets arbitrarily, it becomes impossible to ensure that packets follow authorized paths. We therefore consider a *localized* variant of a Dolev–Yao attacker who compromised one or multiple ASes, including their routers, end hosts, and cryptographic keys. This attacker can only send and receive packets at the compromised (and colluding) AS locations. Such a model is common in path-validation schemes [28, 36] and represents our *basic attacker*.

As is standard in Dolev–Yao models, our model assumes cryptography to be perfect. Consequently, the cryptographic primitives that the protocol is built on must be secure. In particular, this requires that cryptographic keys and authentication fields be sufficiently long to prevent an attacker from brute-forcing authentication fields. If short keys or fields are used, the model's assumptions are violated and the security guarantees no longer hold. For instance, in the case of EPIC L0, if a short hop authenticator was used, an attacker could forge a hop field and create an unauthorized path that could be used to send an arbitrary number of malicious packets that violate *path authorization*. Consequently, EPIC L0 must use long authentication fields to be secure.

Strong-Attacker Model In contrast, our protocols EPIC L1–3 are designed to decrease communication overhead by using short HVFs and segment identifiers. A malicious sender could therefore send large amounts of probing packets—and, with a small chance, guess the correct values for these fields in individual packets.

To reflect the attacker's ability to brute-force the HVFs and segment identifiers in the model, we propose a *strong-attacker* model, which weakens the assumption of perfect cryptography of the basic attacker. In particular, this model allows a malicious sender to obtain valid HVFs and segment identifiers of the validation header by querying an *oracle*.

We define for EPIC levels $l \in \{1,2,3\}$ the oracle $\mathcal{O}^{(l)}$ as the function that for given *PO* and *HI* fields produces valid HVFs V_i and segment identifiers S_i:

$$\mathcal{O}^{(l)}(PO, HI_1, \ldots, HI_\ell) = (V_1^{(l)}, \ldots, V_\ell^{(l)}, S_1^{(l)}, \ldots, S_\ell^{(l)}). \quad (15)$$

The attacker can thus query the oracle and learn the V_i and S_i (but not σ_i or V_SD). As this allows him to trivially construct packets that violate the security properties for ASes, we restrict the security guarantees to packets whose origin *PO* was not part of an oracle query. Security under this model then means that, while the attacker may be able to forge individual packets (obtained from the oracle in the model), this does not help him to craft *different* packets that violate the guarantees.

Additionally, we need to argue in our security analysis that forging individual packets (as modeled by an oracle invocation) does not represent a serious risk for the security of the system: in the next subsection, we will show that the *likelihood* of success of such an attack is low in many practical cases and, even if it succeeds, its *impact* is severely limited. Consequently, the attacker's benefit from brute-forcing a packet is small compared to the computational costs involved.

Protocols that are secure under the basic-attacker model are not necessarily less secure than those under the strong-attacker model, but their implementations must ensure that authenticators are long enough to rule out any practical brute-force attacks. The length of the authenticators is crucial for the communication overhead, which we discuss later.

5.2 Low Risk of Forging Individual Packets

The strong-attacker model explicitly acknowledges the ability of an attacker to brute-force individual HVFs and segment identifiers in EPIC L1–3 through its oracle. However, in practice, the risk of such an attack is limited in four ways:

Table 2: ✓ satisfied, (✓) partially satisfied, and ✗ unsatisfied properties of our protocols EPIC L0–3, ICING, OPT, and PPV. The 2nd and 3rd columns list for whom and under which honesty assumptions the property holds. For protocols evaluated in the strong-attacker model (SA) rather than the basic-attacker model (BA), the 4th column indicates if the property holds only for packets that do not originate from the oracle, or for all packets.

				L0	L1	L2	L3	ICING	OPT	PPV
	who	honesty ass.	packets	(BA)	(SA)	(SA)	(SA)	(BA)	(BA)	(BA)
P1: path authorization	A_i	–	non-oracle	✓	✓	✓	✓	✓	✓	✗
P2: freshness	A_i, H_D	–	all	✗	✓	✓	✓	(✓)	(✓)	✗
P3: packet authentication	H_D	$H_\mathrm{S}, A_1, A_\ell$	all	✗	✗	✓	✓	✓[2]	✓	✓
P4: source authentication	A_i	H_S, A_1	non-oracle	✗	✗	✓	✓	✓[2]	✓[1]	✗
P5: path validation	H_S	A_1	all	✗	✗	✗	✓	✓[2]	✓	✗
P6: path validation	H_D	$H_\mathrm{S}, A_1, A_\ell$	all	✗	✗	✗	✓	✓[2,3]	✓	(✓)[3]

[1] A_i has to additionally assume the honesty of H_D. [2] A_1 and A_ℓ do not need to be honest. [3] A_1 and H_S do not need to be honest.

First, forging even a single packet (i.e., at least one HVF) is expensive as it cannot be performed locally but only by sending packets. Second, a forged packet will be forwarded at most once. The HVFs are bound to the packet origin (source and timestamp). If the attacker brute-forces a HVF and creates an unauthorized (but valid) path, she can violate path authorization or source authentication at routers, but only for a specific *PO*. Any packets with an outdated timestamp in their *PO* will be dropped immediately by routers, meaning that the attack can only happen in a short time frame. Furthermore, the replay-suppression system prevents more than one packet with the same *PO* from being forwarded. Third, in many practical attacks more than a single HVF needs to be brute-forced and the attack becomes exponentially harder in the number of fields to be forged. The probability of forging n HVFs and segment identifiers for any packet is given by $2^{-8n(l_\mathrm{val}+l_\mathrm{seg})}$. Fourth, the security guarantees for end hosts are not affected, since they are based on the validation field V_SD, which is cryptographically strong.

Attacks that only allow a tiny number of packets to be falsely validated by ASes do not pose a grave threat to them. Their concerns regarding path authorization are primarily driven by economic interests, and it suffices if path-policy enforcement works for the vast majority of packets. On the other hand, the main application of source authentication at routers is DoS defense by filtering out unauthentic packets before they reach a bottleneck and enforcing bandwidth reservations through source attribution. For these in-network security applications a small number of forged packets that fool routers (but not the destination) have minimal consequences.

5.3 Path Authorization

The following property protects the path policies of ASes:

P1 *Path authorization*: Packets traverse the network only along paths authorized by all honest on-path ASes.

This enforces the control-plane choices in the data plane and prevents path-splicing attacks: in these, a malicious source would combine hop fields from multiple authorized paths to create an unauthorized path. An *on-path attacker* can exchange the authorized path that the source picked by a different authorized path. Nevertheless, each portion of the path that the packet traverses along *honest* ASes is still authorized.

EPIC L0 and OPT EPIC L0 satisfies path authorization due to its chained hop authenticators: each authenticator contains in its MAC recursively all previous authenticators. Thus, the MAC binds the entire portion of the path from the authenticating AS to the end. Since the property is only achieved in the basic-attacker model, hop authenticators have to be long enough to prevent brute-force attacks. Otherwise, attackers could forge a path and not only use it to send a single packet, but use it for arbitrarily many packets until a hop field expires (based on ts_exp) or the ASes' long-term keys K_i are rotated. OPT also only satisfies P1 in the basic-attacker model since its mechanism is based on SCION / EPIC L0.

EPIC L1–3 EPIC L1 and onward achieve property P1 in the strong-attacker model. These protocols create a HVF for each hop, which is a MAC containing both the hop authenticator σ and the packet origin fields (SRC, TS_path, and ts_pkt).[1] The former ensures path authorization, similar to EPIC L0. The latter ensures that this property holds even under the strong attacker: an attacker who obtains a HVF for a specific *PO* from the oracle cannot use it to create a HVF that is valid for a different *PO*, as the HVF is *bound* to its *PO*.

Both the segment identifier and the HVF directly appear in the packet and are truncated for efficiency reasons. In contrast, the hop authenticator σ itself does *not* appear in the packet and thus does not need truncation as it can be recomputed on demand. The combination of long hop authenticators and short HVFs and segment identifiers minimizes risk; on one hand, a successful brute-forcing attack on a 16 B hop authenticator is practically infeasible; on the other hand, such an attack on a HVF or segment identifier, which is possible by sending a large number of probing packets, has limited impact, as we have discussed in §5.2.

[1] TS_path is indirectly contained in the HVF through the hop authenticator.

ICING and PPV ICING achieves path authorization in the basic-attacker model through its *proofs of consent* (PoCs), which are used to calculate authenticators. PPV does not consider path authorization.

5.4 Freshness

In order to prevent DoS attacks by repeated packet resending, we require that each packet's origin (*PO*) is unique.

P2 *Freshness*: Packets are uniquely identifiable and cannot be replayed.

EPIC L1–3 achieves freshness using a replay-suppression system where *PO*, i.e., the combination of source, path timestamp, and packet timestamp, serves as a unique packet identifier. With such a system in place, the attacker can send at most one unauthorized packet per forged HVF, which is an enormous cost for a very limited return value.

EPIC L0 lacks unique packet identifiers required for replay suppression; ICING and OPT have limited support for replay suppression but do not discuss this in their work. PPV does not use sequence numbers or timestamps and instead uses a "PacketID" based on source, destination, and the hash of the payload. This is insufficient to uniquely identify packets or to enable an efficient replay-suppression system.

5.5 Packet and Source Authentication

Packet and source authentication are desirable properties at the network layer. We formulate authentication as non-injective agreement properties [32]. Together with property P2 and enforcement of the timestamp's validity, they yield strong recent-injective-agreement properties [32].

P3 *Packet authentication for* H_D: The destination H_D agrees with the source H_S on the packet origin, path, and payload unless H_S, its AS, or H_D's AS are corrupted.

P4 *Source authentication for routers*: On-path ASes agree with the source on the packet origin unless the source or its AS are corrupted.

EPIC EPIC L0–1 do not provide any authentication. EPIC L2–3 achieve P3 in the strong-attacker model by computing the destination validation field V_{SD} as the MAC under K_{SD} of the packet timestamp ts_{pkt}, the path (including TS_{path}), and the payload, see Eqs. (11) and (14). Since we assume that V_{SD} is unforgeable (it is not included in $\mathcal{O}^{(l)}$'s output), any source, path, payload, or timestamp modifications by an attacker can be detected by the destination.

EPIC L2–3 achieve P4 since their HVFs are computed as the MAC under the host key K_i^S of the packet timestamp, the source, and the hop authenticator (which is calculated based on the path timestamp). The reasoning is similar to the one for property P3 above, with the difference that individual HVFs are forgeable by sufficiently strong attackers (included in $\mathcal{O}^{(l)}$'s output). The modification of part of the packet origin, i.e., the timestamps or the source, requires forging all honest ASes' HVFs on the path from the attacker to the destination. As a consequence, these routers may falsely authenticate the source of a packet, but, due to freshness (P2), this is limited to *individual packets*, see also §5.2.

OPT OPT authenticates the source and payload, but it achieves property P4 only in the basic-attacker model and only under the additional assumption that H_D is honest. This is due to the use of DRKeys of the form $K_{A_i \to A_1:H_S,A_\ell:H_D}$, which are not only shared between H_S and the intermediate AS A_i, but also with H_D. This weakens the source-authentication property compared to EPIC as all HVFs could also have been created by A_ℓ or H_D. For example, if source authentication is used for bandwidth attribution, a malicious destination could slander the source by fabricating packets or sharing this key.

ICING and PPV ICING achieves both authentication properties P3–P4 through its *proofs of provenance* (PoPs). PPV achieves property P3 through its "PacketID", which is calculated using a secret key shared between H_S and H_D. There is no mechanism in PPV for authentication to routers (P4).

Honesty Assumptions In all schemes discussed here except for ICING (which is not based on DRKey), an end host's use of a host key shared with its AS requires the host's trust in its AS. While this may appear like a strong assumption, a malicious source or destination AS would need to launch an active attack, which hosts can detect by comparing authenticators out of band. Hosts have contracts with their ASes and could have a legal remedy when misbehavior occurs. This is in stark contrast to today's Internet, where hijacks can be performed by an off-path adversary with no relationship to the affected hosts, and no common jurisdiction to settle disputes.

The alternatives to using DRKey in the data plane are using asymmetric cryptography or using symmetric cryptography with pairwise end-to-end keys, which both violate our efficiency requirements (see Sections 2.3 and 3.4).

5.6 Path Validation

Path-validation properties ensure that the *actual* path corresponds to the sender's *intended* path. This is primarily interesting to the end points, for instance if there are compliance rules that mandate certain paths. It can be considered the dual property to path authorization: while path authorization protects the routing decisions of ASes from malicious end hosts, path validation protects the path choices of end hosts from on-path ASes.

P5 *Path validation for* H_S: Upon receiving a reply from H_D, the source H_S can verify that the original packet traversed all honest ASes on the path intended by H_S.

P6 *Path validation for* H_D: H_D can verify that the packet traversed all honest ASes on the path from H_S to H_D intended by H_S.

Both P5 and P6 are achieved by EPIC L3 in the strong-attacker model through the destination validation field V_{SD} (for which the attacker's ability to forge HVFs is irrelevant).

ICING and OPT also satisfy path-validation properties P5 and P6. They additionally ensure that ASes are traversed in

Table 3: Size (in bytes) and number of occurrences (#) of various header fields in a path of length ℓ.

field	content	#	size
TS_{path}	path timestamp	1	4
SRC	source AS and host	1	8
V_i	hop validation field	ℓ	3
S_i	segment identifier	ℓ	2
ts_{pkt}	packet timestamp offset	1	8
V_{SD}	destination validation field	1	16

the correct order. PPV does not allow the source to validate the path (P5) and only probabilistically validates individual links at the destination (P6).

Honesty Assumptions For EPIC L3 and OPT, property P5 requires that the source assumes the honesty of its own AS, since they share the host key. Likewise, for property P6, the destination must assume the honesty of its own AS and also of the source and its AS, since all validation fields are computed by H_S. This assumption is not needed for ICING, which does not rely on DRKey and uses separate keys for the destination. PPV also uses a key which is not shared with the source to achieve property P6 and therefore does not need to assume the source to be honest.

6 Implementation and Evaluation

In this section, we describe our prototype implementation and evaluate its performance. In addition, we analyze the communication overhead of EPIC, OPT, ICING, and PPV as well as of supporting systems. For this analysis, we assume the sizes for various fields in the EPIC header shown in Table 3.

6.1 Implementation and Measurement Setup

To show that EPIC is practically feasible, we implemented and evaluated EPIC L3 prototypes for the source, the routers, and the destination according to the algorithm specification in Algorithms 1–3 using Intel DPDK [18]. As other EPIC levels have a strict subset of processing steps, they would achieve strictly better performance.

In summary, the following evaluation shows that the system can be implemented efficiently even on commodity hardware, it is parallelizable and scales well to core links on the Internet, has significantly lower communication overhead compared to existing systems, requires virtually no state on routers, and limits additional control-plane overhead.

EPIC Packet Structure In our prototype implementation, we follow the packet structure of Eq. (1), using the field sizes specified in Table 3, and extend it with some auxiliary fields (a pointer to the current hop field, the total path length, a version number, and additional flags) and an Ethernet header.

Cryptographic Primitives As we calculate many PRFs and MACs over short inputs and want to avoid the overhead due to subkey generation of CMAC [25], we use the AES-128 block cipher in CBC mode for both PRFs and MACs. As we calculate MACs over variable-length inputs, we prepend the input length and use zero padding such that the CBC-MAC indeed fulfills all properties of a PRF and a MAC [7]. Because EPIC and DRKey heavily rely on MAC and PRF calculations, we use Intel's AES-NI hardware instructions [41], available on all modern Intel CPUs, to reduce the computation time.

HVF Store at the Source The store of HVFs of sent packets at the source is implemented as a hash table as it enables insertion and retrieval of data using the 12-byte key $(TS_{\text{path}} \parallel ts_{\text{pkt}})$ with average complexity $O(1)$ and there exists a ready-to-use hash-table implementation in DPDK.

Measurement Setup The prototypes are evaluated using a Spirent SPT-N4U, which serves as packet generator and bandwidth monitor, and a commodity machine with an 18-core Intel Xeon 2 GHz processor executing the component to be tested, i.e., the source or router. The two machines are connected with a 40 Gbps Ethernet link.

We evaluate the performance of the prototype as a function of the EPIC L3 payload. However, the size of the EPIC header depends on the AS-level path length and therefore contributes dynamically to the Ethernet packet content. To test the prototypes using the same EPIC L3 payload range, independent of the path length, we enable jumbo-frame support (Ethernet frames with more than 1500 B payload) on both machines.

The current average path length in the Internet is less than 4 AS-level hops [24, 33, 44, 45]. However, as we expect that number to increase due to the benefits of being an AS in a path-aware Internet, we consider path lengths of up to 16 AS-level hops in our evaluation (the current average number of router-level hops is 13).

6.2 Performance Evaluation

In this section, we evaluate the performance of our implementation in terms of throughput (total traffic) and goodput (payload traffic). Note that we account for the full header overhead as described above when referencing the goodput.

Source For the evaluation of the source we assume that it has already fetched the necessary hop authenticators and DRKeys, which corresponds to the situation of an existing connection. The throughput achieved by the source (using a single CPU core) is shown in Fig. 1. For packets of $p \geq 500$ B and path lengths of $\ell \leq 8$, the prototype implementation consistently achieves throughput above 2 Gbps. Figures 7 and 8 in Appendix A further illustrate the parallelizability of the implementation, which enables throughputs of tens of Gbps, and the linear increase of the processing time with both payload size and path length.

The processing at the source and destination is similar for ICING, OPT, and PPV; in all protocols either a MAC or hash is calculated over the packet's payload, which dominates the computational effort. In the future, these cryptographic computations could be offloaded to multiple dedicated hardware units in network-interface cards (NICs).

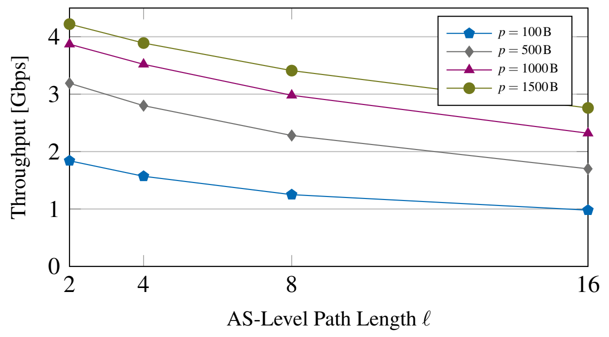

Figure 1: EPIC L3 packet throughput generated by the source on a single core for different payload sizes.

Router Figure 2 shows the forwarding performance of an EPIC L3 router for a path of length $\ell = 8$. In these measurements, we assume no cached hop authenticators or DRKeys, they are always recalculated on the fly. For packets with a payload $p \geq 500$ B, the 40 Gbps link is saturated for all path lengths using only 4 cores; using 16 cores, the link is even saturated for small packets ($p = 100$ B). As the implementation is easily parallelizable (see Fig. 6 in Appendix A), it can be used even on 100 Gbps or 400 Gbps links by adding more processing cores or dedicated hardware. An important observation is that the processing time of the router is 445 – 460 ns independent of both payload size and path length. The forwarding performance in terms of Mpps (million packets per second) is thus also independent of these parameters and amounts to approximately 2 Mpps per processing core. These results are further illustrated by Figs. 4–6 in Appendix A.

The processing on routers is similar for all levels of EPIC, OPT, and PPV, which all have a small constant number of cryptographic operations. In ICING, every router calculates both a hash and a MAC over the payload and in addition performs ℓ symmetric cryptographic operations (one for each router). In the software implementation provided by ICING's authors [36], each router has a processing time of $\sim$50 µs for $\ell = 10$, which is two orders of magnitude slower than EPIC. If keys are not cached, additional Diffie–Hellman computations are necessary, leading to processing times of ≥ 100 ms [19].

Comparison to IP Comparing the performance of EPIC to IP is challenging due to the strong impact of routing-table sizes on software performance and hardware cost for IP. Highly optimized software switch implementations like DPDK vSwitch achieve throughputs of $\sim$11 Mpps on a single core (corresponding to a processing time of approximately 90 ns) [20]. However, these values are only valid for small routing tables when no memory accesses are necessary (as a single DRAM access takes $\sim$70 ns). Our prototype implementation is approximately five times slower at $\sim$2 Mpps, but the throughput is independent of the number of concurrent flows due to packet-carried forwarding state. Furthermore, the processing time could be further reduced through optimizations such as concurrent execution of cryptographic operations.

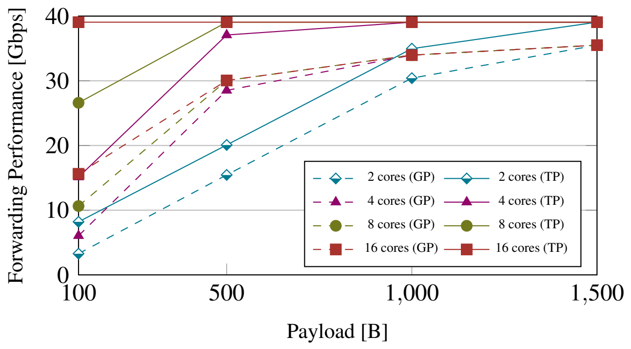

Figure 2: Throughput (TP) and goodput (GP) of a router plotted against the payload for 2, 4, 8, and 16 cores and $\ell = 8$.

Table 4: Communication overhead in bytes in EPIC, ICING, OPT, and PPV due to security-related fields.

	L0	L1	L2–L3	ICING	OPT	PPV
	3ℓ	$5\ell+8$	$5\ell+24$	$42\ell+13$	$19\ell+52$	64
for $\ell = 8$	24	48	64	349	204	64

Hardware implementations, which are particularly relevant in a production deployment, compare even more favorably. IP routers require large amounts of expensive ternary content-addressable memory (TCAM) for longest-prefix matching. In contrast, EPIC requires very little additional hardware for its cryptographic operations. Naous et al. [36] have compared the gate count of FPGA implementations of ICING and IP routers and found comparable values (13.4 million vs. 8.7 million gates) even for very small amounts of TCAM in the IP router; in comparison, hardware implementations of AES are very efficient and only require 13,000 gates [1].

6.3 Communication Overhead

In addition to processing overhead and performance, we also evaluate the communication overhead of EPIC and compare it to other systems. To allow for a meaningful comparison, we evaluate only the overhead owed to security here, since the normal routing headers (e.g., IPv4/v6, SCION) depend on the underlying networking architecture. Thus, we use HD to refer to the size of all security-related header fields (in EPIC, these are ts_{pkt}, V_{SD}, and S_i, V_i for all hops i). We define the goodput ratio as the ratio between goodput and throughput, or, equivalently, as the ratio of payload and total packet size, $GR = \frac{p}{p+HD}$. Table 4 shows the size of the additional header for all considered systems, Fig. 3 depicts the goodput ratio.

We find that the goodput ratio is high for all variants of EPIC. For $\ell = 8$, the additional header is between 24 B for EPIC L0 and 64 B for EPIC L3, which corresponds to a goodput ratios 98 % and 94 %, respectively, for payloads of size $p = 1000$ B. The goodput ratio of OPT is significantly worse with $GR \sim 83$ % for the same values of ℓ and p, and does not scale as well as the overhead of EPIC with the length of the paths. For ICING, we find a five times larger overhead

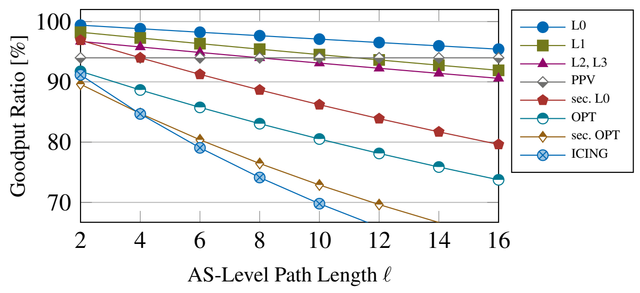

Figure 3: Goodput ratio of different protocols as a function of the AS-level path length ℓ for a 1000 B payload, calculated from Table 4. "sec. L0" and "sec. OPT" correspond to L0 and OPT with authenticators of 16 B that are required to rule out brute-force attacks. For $GR < 2/3$, the total packet size exceeds the maximum size for an Ethernet payload.

than EPIC L2–3 and $GR \sim 74\,\%$ for these parameters. As PPV performs checks at only two routers along the path, its overhead is constant in the path length. Still, EPIC L2–3 have a higher goodput ratio than PPV for path lengths up to $\ell = 8$.

In Table 4, authenticators for path authorization are 3 B for EPIC L0 and OPT (the default for SCION on which they are based). This is despite only achieving property P1 in the basic-attacker model, meaning that brute-force attacks are unmitigated and exploitable for practical attacks. To correct for this, the size of HVFs would need to be increased to a similar length of other brute-force-resistent fields like the destination validation field, i.e., 16 B. Considering these modifications, which are shown by "sec. L0" and "sec. OPT" in Fig. 3, the goodput ratio is even more favorable for EPIC L1–3, which significantly outperform both protocols.

6.4 Other Overhead

State at Routers In EPIC, routers can perform all cryptographic checks and updates with a single AS-specific secret value, there is no per-host or per-flow state required. This is equivalent to OPT and PPV, which both rely on DRKey, but a significant advantage compared to ICING, which requires per-flow state [28, 36]. In terms of routing information, border routers only need to store intra-AS information as packet headers contain the inter-AS forwarding information. This is a huge improvement over the current Internet, shared by all architectures based on packet-carried forwarding state.

Replay Suppression All EPIC levels except L0 depend on a replay-suppression system for freshness (P2), which has additional state and overhead. Since this task can be taken over by dedicated machines, we did not include it in the router measurements above. Prototypes that are entirely implemented in software have been deployed successfully on 10 Gbps links [29]. In turn, EPIC L1–3 provide important properties for replay suppression: (i) the system can use the timestamp to discard packets that are expired, thus limiting the number of packets that need to be tracked in Bloom filters, and (ii) by authenticating all packet contents tracked by the replay-suppression system, attackers are prevented from modifying unauthenticated fields and replaying packets. If the replay-suppression system were not deployed, the packet timestamp could still be used to filter out expired packets, and an attacker could only replay packets in a very short time window due to the check in line 4 of Algorithm 2.

Control-Plane Overhead In EPIC, end hosts have to request paths from the path server and, for EPIC L2–3, host-level symmetric keys from the key server, before they can communicate with a new destination. We assume that the underlying path-aware Internet architecture minimizes latency by locally caching public paths, e.g., at path servers in SCION [37, §7.2]. End hosts also cache paths themselves, such that only the initial packet to a new destination requires a path lookup. This caching strategy can also be applied to EPIC's hop authenticators and the host keys required in EPIC L2–3. Concretely, AS-level keys can be set up between every pair of ASes ahead of time (either using PISKES / DRKey or Passport) such that local key servers can immediately respond to requests by end hosts. In the current Internet, storing 16 B keys for each AS only amounts to ∼1 MB [34]. Given that path and key information is available at local ASes, the additional latency incurred in EPIC is minimal: only the round-trip time between the source host and its own AS, and the destination host and its AS is added to the connection setup. End hosts can cache both paths and host keys, which eliminates additional latency for subsequent packets. Further optimization would be possible by combining DNS, path, and key requests, which would eliminate all additional latency for the initial packet compared to today's Internet.

7 Discussion

Low Communication Overhead of EPIC The benefit of EPIC's lower overhead compared to OPT comes in part from the fact that EPIC does not use separate fields for path authorization on the one hand, and for authentication and path validation on the other hand. The larger contributor to a lower overhead is however the shorter length of HVFs in EPIC of 3 B, compared to the 16 B OPV fields in OPT. While a shorter authenticator translates to easier brute-force attacks (and thus seemingly weaker security), we have shown the practical usefulness of such attacks is severely limited by EPIC, as the attacker can only send a single packet that traverses an unauthorized path, and in EPIC L2-3 that packet will be discarded by the destination, see §5.2. EPIC is the first data-plane protocol designed to limit the consequences of a successful brute-force attack to a single packet; previous protocols rely on long authenticators to prevent harmful attacks.

Deployment on Path-Aware Architectures Our data-plane protocols are generic and applicable to a wide range of path-aware networking protocols. We now describe how EPIC fits into these architectures.

In SCION, the authenticators used in EPIC can be used directly instead of the built-in MACs that protect hop fields. However, a difference to EPIC is that in SCION only a subset of ASes called *cores* (typically, Tier-1 providers) initiate beacons. These beacons have limited reach and do not discover the entire Internet topology for scalability reasons. Thus, end hosts must combine paths from multiple beacons to obtain global end-to-end paths. SCION defines rules for combining multiple segments to rule out loops and uneconomical routes (such as *valley paths* [21] [37, §8.2]) and allows paths to be used in either direction. While our presentation of EPIC abstracts from these aspects, we designed the protocols with path combinations and bidirectionality in mind. For combined paths, path authorization holds for each segment individually while path validation applies to the complete path.

Besides SCION, multiple other path-aware Internet architectures cryptographically protect forwarding directives in packet headers, including NEBULA [3, 36], PoMo [9], and Platypus [38, 39]. PoMo introduces an abstract "motivation" header that can be calculated in the same way as the HVFs of EPIC. NEBULA uses "proofs of consent" for path authorization and, with the ICING extension, achieves source authentication and path validation through its "proofs of provenance". EPIC can be used to replace these proofs to significantly reduce both computation and communication overhead while only slightly weakening security properties. The "bindings" in Platypus already implement a system similar to EPIC L1; they could, however, easily be augmented with source authentication and path validation with EPIC L2–3.

Incremental Deployment The (incremental) deployment of a new Internet architecture is very challenging but is facilitated by the reuse of existing intra-domain infrastructure and protocols. An extensive discussion of (incremental) deployment is provided for the SCION architecture [37, Chapter 10]. In turn, the incremental deployment of EPIC on an existing path-aware architecture—e.g., as a premium product for customers requiring stronger security properties such as the financial and healthcare sectors—is benefited precisely by their path awareness: EPIC only requires support by on-path ASes and can thus be supported on some paths without requiring global coordination. An upgraded end host can then favor these paths, providing benefits to early adopters.

Timestamps and Time Synchronization The path timestamp TS_{path} encodes Unix time with second-level precision; both the expiration time of hop fields (ts_{exp}) and the packet timestamp introduced in EPIC (ts_{pkt}) are relative to TS_{path}. The length of the ts_{exp} field determines a maximum lifetime for hop fields. As a path expires when one of its hop fields expires, the packet timestamp offset ts_{pkt} only needs to cover the period between creation and expiration of a beacon. For instance, in SCION, this period is at most one day [37, §15.1]. An 8 B field then corresponds to a granularity of $\sim$5 fs. This enables end hosts to send $2 \cdot 10^{14}$ packets with unique timestamps per second, which is sufficient for any practical application. We can consequently use the packet origin, i.e., the triple of source, path timestamp, and packet timestamp defined in Eq. (2), to uniquely identify all packets in the network.

The timestamps serve multiple purposes in EPIC: they (i) allow routers to drop packets that are too old or use expired paths, (ii) uniquely identify packets, and (iii) ensure that the replay-suppression system only needs to track recent packets. For the first purpose, a coarse global time synchronization providing a precision of multiple seconds is sufficient. The second purpose does not require time synchronization at all, as packets are uniquely identified based on the packet origin, which also includes the source. The third purpose has been shown to work based on per-AS sequence numbers and therefore only requires relatively precise time synchronization *within* an AS [29]. The higher-order bits of the packet timestamp can serve as sequence numbers in this replay-suppression system.

Key Distribution The use of DRKeys in EPIC L2–3 creates potential issues of circular dependencies: how is it possible to exchange DRKeys when they themselves are required for sending packets? In a steady state, this is unproblematic as ASes can proactively exchange new AS-level keys before the current keys expire using EPIC L2–3 packets. For an initial key exchange, which only happens very infrequently, we propose to support EPIC L1 in addition, such that key requests can be sent over this lower-level protocol. Although EPIC L1 has lower security guarantees and may be susceptible to DoS attacks, these issues are mitigated by the fact that only a single request and response are needed for fetching a key. Even in a persistent, powerful DoS attack, such an exchange would succeed eventually.

Confirmation Packets in EPIC L3 In EPIC L3, the confirmation message that allows the source to validate the path of its packets is sent as an EPIC L2 packet. This is necessary as each confirmation message would otherwise trigger yet another confirmation and consequently cause an infinite sequence of such confirmations. Using EPIC L2 means that the path of the original packet but not the confirmation message can be validated; all other security properties are retained (see also Table 2). Even in case a malicious on-path AS is able to modify the path of the confirmation message without being detected, this does not deteriorate the security properties of the original packet.

There are a number of possible optimizations for the confirmation message similar to acknowledgments in TCP: Instead of directly sending confirmation message for every received packet, the receiver can batch several confirmation messages and send them in a single packet. Confirmation messages can also be "piggy-backed" on normal data packets sent from the receiver to the source. Finally, instead of sending all HVFs, only a hash of them can be returned to the source and validated against the stored values.

Failure Scenarios As the EPIC protocols depend on several additional systems, a failure of any of these systems could potentially break connectivity. Most failure scenarios are comparable to similar issues in today's Internet: Failures of path or key servers are similar to failures of DNS servers today and can be prevented with similar techniques (e.g., replication, access control). Concerning potential misconfigurations, EPIC may actually increase the networks resilience as some concepts of new Internet architectures such as SCION's isolation domains ensure that the effects of misconfigurations are locally confined [37].

The most notable additional prerequisite of EPIC is time synchronization; it is possible that (i) a host, (ii) some router or server in an AS, or (iii) a complete AS is unsynchronized with the Internet. The first case can be handled by the host's AS replying with a corresponding control message triggering a re-synchronization. Cases (ii) and (iii) can be detected through increased packet-drop rates and can thus trigger a re-synchronization within the AS or with its neighbors. All cases may cause brief outages but can be resolved within a short time period (one second or less in most cases).

Path Validation for Intermediate Routers Path validation is primarily interesting to the end points. Despite this, ICING and OPT allow not only the source and destination to validate the path of a packet, but also enable intermediate routers to validate the portion of the path that has already been traversed. The authors of ICING and OPT provide little motivation to provide path validation for routers, and since we are not aware of any important use cases of this feature we have omitted it from our design criteria of EPIC L1–3. However, for the sake of completeness we describe an extension of EPIC L3 in Appendix B to also satisfy this property.

8 Related Work

Over the past 15 years, much research was conducted on path-aware Internet architectures and routing schemes including Platypus [38, 39], PoMo [9], Pathlet Routing [22], NIRA [46], NEBULA [3], and SCION [37, 47]. Many of these systems recognized the need to find a balance of control between end hosts and ASes. This is why PoMo includes a "motivation" field containing a proof to routers that either the sender or receiver is a paying customer [9], NEBULA requires a "proof of consent" for the complete path of traversed ASes [3, 36], and SCION secures the authorization of its hop fields using MACs [37]. These solutions correspond to EPIC L0 in terms of the path authorization properties achieved. NIRA and Pathlet Routing obtain similar properties by restricting allowed paths (NIRA) and keeping state in routers (NIRA and Pathlet Routing) [22, 46]. Platypus uses a system similar to Level 1 presented in §4.2 where each network capability is secured by a "binding", but it does not address the issue of chaining multiple hops to paths [38, 39].

In addition, since PFRI (integrated into PoMo) discussed a high-level outline for a path-validation system via an "accountability" field in packets [15], multiple path-validation schemes have been proposed. ICING [36] is integrated into the NEBULA architecture and provides path validation using a validation field for each hop [36]. It uses aggregate MACs [27] in order to limit the bandwidth overhead but still requires each router to perform one symmetric cryptographic computation for *each* other router on the path (and, if keys are not cached, an additional *asymmetric* Diffie–Hellman computation), which makes it very expensive. Subsequent proposals try to reduce the complexity through different means: OPT reduces the required cryptographic computations to a constant number by sacrificing some guarantees for intermediate routers, yet it still has a high communication overhead [28, 37]. OSV tries to create a more efficient system by replacing cryptographic primitives by orthogonal sequences based on Hadamard matrices [12, 13]. Finally, PPV reduces both computation and communication overhead by only probabilistically validating a single link for each packet [45].

9 Conclusion

Several path-aware Internet architectures proposed in recent years promise to improve the security and efficiency of the Internet by providing path control to end hosts. However, this shift of control requires mechanisms to protect the routing policies of ASes from malicious end hosts on the one hand, and raises the challenge of verifying that the path directives were followed by ASes on the other hand. Previous systems for both path authorization and path validation faced a dilemma between security and efficiency in terms of communication overhead.

The highly efficient EPIC protocols proposed in this paper resolve this dilemma, and furthermore enable all on-path routers and the destination to authenticate the source of a packet. Thus, by ensuring that the source and path of every packet is checked efficiently at the network layer, EPIC enables a wide range of additional in-network security systems like packet filtering for DoS-defense systems and provides a secure foundation for the data plane of a future Internet.

Acknowledgments

We would like to thank Y.-C. Hu, who provided important early feedback and suggestions; B. Rothenberger for support with the implementation; K. Paterson, who improved our understanding of cryptographic primitives; and S. Hitz and S. Monroy for discussions about the practical aspects of our protocols. We are grateful to the five anonymous reviewers for their very helpful feedback on the manuscript, and our shepherd, B. Reaves, for his guidance during the preparation of the final version of the paper. We gratefully acknowledge financial support from ETH Zurich and from the Zurich Information Security and Privacy Center (ZISC).

References

[1] Alma'aitah, A., and Abid, Z.-E. Transistor level optimization of sub-pipelined AES design in CMOS 65nm. In *Proceedings of the International Conference on Microelectronics (ICM)* (2011), IEEE.

[2] Amini, L., Shaikh, A., and Schulzrinne, H. Issues with inferring Internet topological attributes. *Computer Communications 27*, 6 (2004).

[3] Anderson, T., Birman, K., Broberg, R., Caesar, M., Comer, D., Cotton, C., Freedman, M. J., Haeberlen, A., Ives, Z. G., Krishnamurthy, A., Lehr, W., Loo, B. T., Mazières, D., Nicolosi, A., Smith, J. M., Stoica, I., van Renesse, R., Walfish, M., Weatherspoon, H., and Yoo, C. S. The NEBULA future Internet architecture. In *The Future Internet*. Springer, 2013.

[4] Augustin, B., Cuvellier, X., Orgogozo, B., Viger, F., Friedman, T., Latapy, M., Magnien, C., and Teixeira, R. Avoiding traceroute anomalies with Paris traceroute. In *Proceedings of the ACM SIGCOMM conference on Internet measurement* (2006).

[5] Basescu, C., Lin, Y.-H., Zhang, H., and Perrig, A. High-speed inter-domain fault localization. In *Proceedings of the IEEE Symposium on Security and Privacy* (2016).

[6] Basescu, C., Reischuk, R. M., Szalachowski, P., Perrig, A., Zhang, Y., Hsiao, H.-C., Kubota, A., and Urakawa, J. SIBRA: Scalable Internet bandwidth reservation architecture. In *Proceedings of the Symposium on Network and Distributed System Security (NDSS)* (2016).

[7] Bellare, M., Kilian, J., and Rogaway, P. The security of the cipher block chaining message authentication code. *Journal of Computer and System Sciences 61*, 3 (2000).

[8] Benzekki, K., El Fergougui, A., and Elbelrhiti Elalaoui, A. Software-defined networking (SDN): a survey. *Security and Communication Networks 9*, 18 (2016).

[9] Bhattacharjee, B., Calvert, K., Griffioen, J., Spring, N., and Sterbenz, J. P. G. Postmodern internetwork architecture. *NSF Nets FIND Initiative* (2006).

[10] Birge-Lee, H., Sun, Y., Edmundson, A., Rexford, J., and Mittal, P. Bamboozling certificate authorities with BGP. In *Proceedings of the USENIX Security Symposium* (2018).

[11] Bu, K., Yang, Y., Laird, A., Luo, J., Li, Y., and Ren, K. What's (not) validating network paths: A survey. *arXiv preprint arXiv:1804.03385* (2018).

[12] Cai, H., and Wolf, T. Source authentication and path validation with orthogonal network capabilities. In *Proceedings of the IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)* (2015).

[13] Cai, H., and Wolf, T. Source authentication and path validation in networks using orthogonal sequences. In *Proceedings of the International Conference on Computer Communication and Networks (ICCCN)* (2016).

[14] CALOMEL. AES-NI SSL performance: a study of AES-NI acceleration using LibreSSL, OpenSSL. https://calomel.org/aesni_ssl_performance.html, 2018.

[15] Calvert, K. L., Griffioen, J., and Poutievski, L. Separating routing and forwarding: A clean-slate network layer design. In *Proceedings of the International Conference on Broadband Communications, Networks and Systems (BROADNETS)* (2007).

[16] DARPA. Internet protocol. RFC 791, 1981.

[17] Dolev, D., and Yao, A. On the security of public key protocols. *IEEE Transactions on Information Theory 29*, 2 (1983).

[18] DPDK Project. Data Plane Development Kit. https://dpdk.org.

[19] ECRYPT. eBATS: ECRYPT benchmarking of asymmetric systems. https://bench.cr.yp.to/results-dh.html, 2019.

[20] Emmerich, P., Raumer, D., Wohlfart, F., and Carle, G. Assessing soft-and hardware bottlenecks in PC-based packet forwarding systems. *ICN* (2015).

[21] Gao, L., and Rexford, J. Stable Internet routing without global coordination. *IEEE/ACM Transactions on Networking* (2001).

[22] Godfrey, P. B., Ganichev, I., Shenker, S., and Stoica, I. Pathlet routing. In *Proceedings of ACM SIGCOMM* (2009).

[23] Gueron, S. Intel® advanced encryption standard (AES) new instructions set. *Intel Corporation* (2010).

[24] Huffaker, B., Fomenkov, M., Plummer, D. J., Moore, D., and Claffy, K. Distance metrics in the Internet. In *Proceedings of the IEEE International Telecommunications Symposium (ITS)* (2002).

[25] Iwata, T., Song, J., Lee, J., and Poovendran, R. The AES-CMAC Algorithm. RFC 4493, 2006.

[26] Kang, M. S., Lee, S. B., and Gligor, V. D. The Crossfire attack. In *IEEE Symposium on Security and Privacy* (2013).

[27] Katz, J., and Lindell, A. Y. Aggregate message authentication codes. In *Topics in Cryptology – CT-RSA*. Springer, 2008.

[28] Kim, T. H.-J., Basescu, C., Jia, L., Lee, S. B., Hu, Y.-C., and Perrig, A. Lightweight source authentication and path validation. In *Proceedings of ACM SIGCOMM* (2014).

[29] Lee, T., Pappas, C., Perrig, A., Gligor, V., and Hu, Y.-C. The case for in-network replay suppression. In *Proceedings of the ACM Asia Conference on Computer and Communications Security (ASIACCS)* (2017).

[30] Lepinski, M., and Kent, S. An infrastructure to support secure Internet routing. RFC 6480, 2012.

[31] Liu, X., Li, A., Yang, X., and Wetherall, D. Passport: secure and adoptable source authentication. In *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation* (2008).

[32] Lowe, G. A hierarchy of authentication specification. In *Computer Security Foundations Workshop (CSFW)* (1997).

[33] Magoni, D., and Pansiot, J.-J. Internet topology modeler based on map sampling. In *Proceedings of the International Symposium on Computers and Communications (ISCC)* (2002).

[34] Maigron, P. Autonomous system number statistics. https://www-public.imtbs-tsp.eu/~maigron/RIR_Stats/RIR_Delegations/World/ASN-ByNb.html.

[35] McCauley, J., Harchol, Y., Panda, A., Raghavan, B., and Shenker, S. Enabling a permanent revolution in Internet architecture. In *Proceedings of ACM SIGCOMM* (New York, NY, USA, 2019).

[36] Naous, J., Walfish, M., Nicolosi, A., Mazieres, D., Miller, M., and Seehra, A. Verifying and enforcing network paths with ICING. In *Proceedings of the ACM International Conference on emerging Networking EXperiments and Technologies (CoNEXT)* (2011).

[37] Perrig, A., Szalachowski, P., Reischuk, R. M., and Chuat, L. *SCION: A Secure Internet Architecture*. Springer, 2017. ISBN 978-3-319-67080-5, https://www.scion-architecture.net/pdf/SCION-book.pdf.

[38] Raghavan, B., and Snoeren, A. C. A system for authenticated policy-compliant routing. *ACM SIGCOMM Computer Communication Review 34*, 4 (2004).

[39] Raghavan, B., Verkaik, P., and Snoeren, A. C. Secure and policy-compliant source routing. *IEEE/ACM Transactions on Networking 17*, 3 (2009).

[40] Rothenberger, B., Roos, D., Legner, M., and Perrig, A. PISKES: Pragmatic Internet-scale key-establishment system. In *Proceedings of the ACM Asia Conference on Computer and Communications Security (ASIACCS)* (2020).

[41] ROTT, J. Intel advanced encryption standard instructions (AES-NI). *Technical Report, Intel* (2010).

[42] STUDER, A., AND PERRIG, A. The Coremelt attack. In *Proceedings of the European Symposium on Research in Computer Security (ESORICS)* (2009).

[43] SUN, Y., EDMUNDSON, A., VANBEVER, L., LI, O., REXFORD, J., CHIANG, M., AND MITTAL, P. RAPTOR: Routing attacks on privacy in Tor. In *Proceedings of USENIX Security Symposium* (2015).

[44] WANG, C., LI, Z., HUANG, X., AND ZHANG, P. Inferring the average AS path length of the Internet. In *Proceedings of the IEEE International Conference on Network Infrastructure and Digital Content (IC-NIDC)* (2016).

[45] WU, B., XU, K., LI, Q., LIU, Z., HU, Y.-C., REED, M. J., SHEN, M., AND YANG, F. Enabling efficient source and path verification via probabilistic packet marking. In *Proceedings of the IEEE/ACM International Symposium on Quality of Service (IWQoS)* (2018).

[46] YANG, X., CLARK, D., AND BERGER, A. W. NIRA: A new inter-domain routing architecture. *IEEE/ACM Transactions on Networking* (2007).

[47] ZHANG, X., HSIAO, H.-C., HASKER, G., CHAN, H., PERRIG, A., AND ANDERSEN, D. SCION: Scalability, control, and isolation on next-generation networks. In *Proceedings of the IEEE Symposium on Security and Privacy* (2011).

A Additional Evaluation Results

Processing-Time Analysis Figure 4 shows a fine-grained processing-time analysis of the router for EPIC L3, highlighting the overhead caused by cryptographic operations. The times include necessary copying and padding of the input to the AES block cipher.

Figure 5 shows the processing time of an EPIC L3 router for different path lengths and EPIC payload sizes. As expected, the processing time is independent of the path length and payload size and shows low deviation of only few percent.

Parallelizability As shown in Fig. 6, the router implementation achieves almost perfectly linear speedup when parallelized over multiple CPU cores. As a consequence, the EPIC router can be easily scaled to larger network links by adding more processing cores or dedicated hardware. The source shows a similar linear speedup as a function of the number of cores, see Fig. 7.

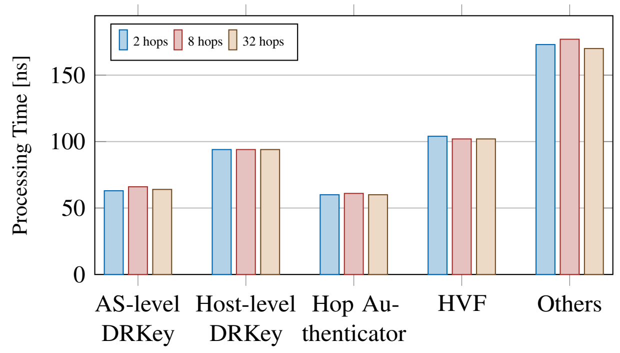

Figure 4: EPIC L3 router processing times for different sub-tasks. The category 'Others' aggregates all non-cryptographic operations, for example checking the expiration time, writing the updated hop-validation field, or increasing the hop pointer.

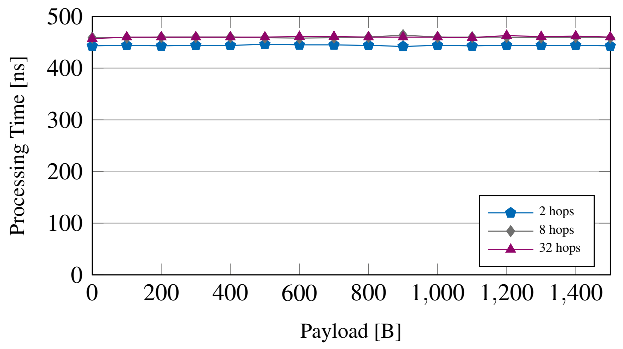

Figure 5: EPIC L3 router processing time as a function of the payload for $\ell \in \{2, 8, 32\}$.

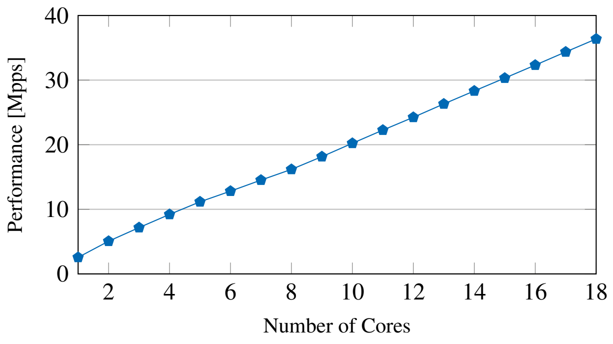

Figure 6: Forwarding performance of a router, as a function of the number of used cores measured for $p = 0$ and $\ell = 2$. As the packet-processing time is independent of p and ℓ as shown in Figs. 4 and 5, this result is also valid for larger packets and longer paths.

Processing Time at the Source The processing time at the source for EPIC L3 is depicted in Fig. 8. It increases linearly with both the AS-level path length (due to the HVF for each hop) and in the EPIC payload (due to the destination validation field).

B Path Validation for Routers

We describe EPIC L4, which modifies EPIC L3 to further achieve path validation for routers:

P7 *Path validation for routers*: Each router A_i can verify that the packet traversed all honest ASes from H_S to A_i on the path intended by H_S.

This protocol otherwise has the same security properties and communication overhead as EPIC L3.

In EPIC L4, the source of the packet *obfuscates* the HVFs of all ASes by XOR-ing them with cryptographic results of previous ASes. Unless the previous ASes on the path reverse this obfuscation, the HVF of an AS is invalid. As obfuscation values, we propose to use another piece $C_i^{[3]} = C_i[\![2l_{\text{val}}:3l_{\text{val}}]\!]$ of C_i defined in Eq. (12) (assuming $l_{\text{PRF}} \geq 3 \cdot l_{\text{val}}$). The source of a packet now initializes the HVF by

$$V_{i;0}^{(4)} := C_i^{[1]} \oplus C_{i-1}^{[3]} \oplus C_{i-2}^{[3]} \oplus \cdots \oplus C_{i-2^k}^{[3]} \qquad (16)$$

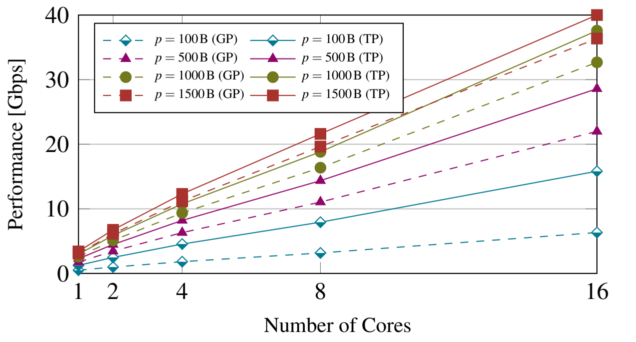

Figure 7: EPIC L3 packet-generation performance at the source, plotted against different number of cores and payload sizes, and for $\ell = 8$. The legend entries 'TP' and 'GP' denote the throughput and goodput, respectively.

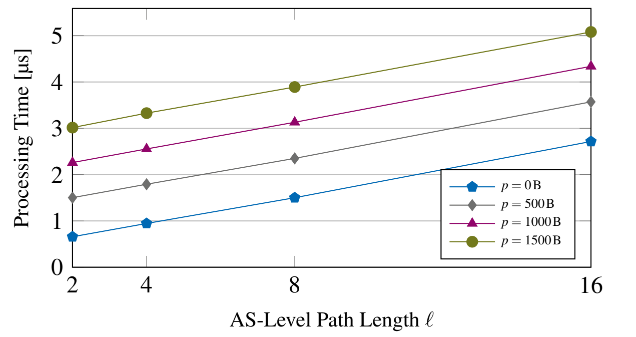

Figure 8: EPIC L3 packet-processing time at the source on a single core for different EPIC payload sizes and path lengths.

and intermediate ASes update future HVFs by XOR-ing them with $C_i^{[3]}$. Note that we are not (de-)obfuscating all subsequent ASes but only those at *exponentially growing* distances. Doing this for all subsequent ASes would enable colluding ASes to easily skip ASes on the path and deceive subsequent routers by XOR-ing the validation fields of the skipped ASes. Table 5 presents the evolution of the HVF values $V_i^{(4)}$ as the packet traverses four ASes. The source obfuscates the HVFs such that they will have the value $C_i^{[2]}$ upon reception by the destination if and only if all routers processed the packet successfully.

For EPIC L4, path validation for routers (P7) is achieved under the following honesty assumption in addition to those described in Table 2: on any contiguous part of the path of at least three hops there is a majority of honest ASes.

Hop-Skipping Attack For property P7 in EPIC L4, colluding ASes may be able to deceive ASes on the future path to accept a packet, even if ASes on the past path were skipped, by analyzing HVFs and recovering the obfuscation values $C_i^{[3]}$ for skipped ASes i. However, due to the exponential distances used for obfuscation, this de-obfuscation requires at least one more colluding AS than the number of skipped ASes. An example with two skipped ASes is shown in Fig. 9.

Note that in EPIC L4, if the HVFs V_i were obfuscated with $C_j^{[3]}$ for all $j < i$ (instead using exponential distances), two colluding ASes could always recover the obfuscation values of the ASes between them. Thus, any two colluding ASes could create a wormhole that would be detectable by H_S and H_D but not by subsequent ASes.

Table 5: Values of HVFs in EPIC L4 as a packet is forwarded from A_1 to A_4. Colors indicate α in $C_i^{[\alpha]}$.

	after processing by				
HVF	H_S	A_1	A_2	A_3	A_4
$V_1^{(4)}$	$C_1^{[1]}$	$C_1^{[2]}$	$C_1^{[2]}$	$C_1^{[2]}$	$C_1^{[2]}$
$V_2^{(4)}$	$C_2^{[1]} \oplus C_1^{[3]}$	$C_2^{[1]}$	$C_2^{[2]}$	$C_2^{[2]}$	$C_2^{[2]}$
$V_3^{(4)}$	$C_3^{[1]} \oplus C_2^{[3]} \oplus C_1^{[3]}$	$C_3^{[1]} \oplus C_2^{[3]}$	$C_3^{[1]}$	$C_3^{[2]}$	$C_3^{[2]}$
$V_4^{(4)}$	$C_4^{[1]} \oplus C_3^{[3]} \oplus C_2^{[3]}$	$C_4^{[1]} \oplus C_3^{[3]} \oplus C_2^{[3]}$	$C_4^{[1]} \oplus C_3^{[3]}$	$C_4^{[1]}$	$C_4^{[2]}$

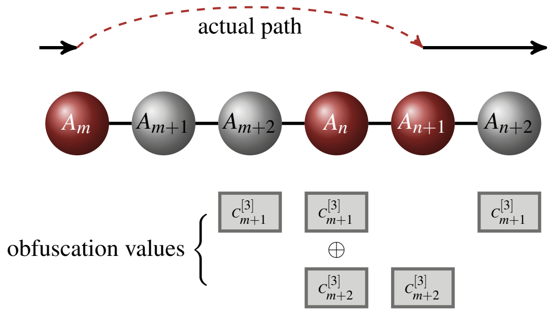

Figure 9: Example where colluding malicious ASes A_m, A_n, and A_{n+1} skip two intermediate ASes ($n = m + 3$) and deceive future ASes on the path to accept the diverted packet. The pattern of obfuscation values produced by A_{m+1} and A_{m+2} for following ASes is drawn below their nodes. As the attacker sees $C_{m+1}^{[3]} \oplus C_{m+2}^{[3]}$ as well as $C_{m+2}^{[3]}$, he can remove the obfuscation normally removed by A_{m+1} and A_{m+2} from all future HVFs. If only A_n but not A_{n+1} were controlled by the attacker, he would only see $C_{m+1}^{[3]} \oplus C_{m+2}^{[3]}$ and therefore could not deceive A_{n+1}.