FABRID: Flexible Attestation-Based Routing for Inter-Domain Networks

Cyrill Krähenbühl, Marc Wyss, and David Basin, *ETH Zürich;* Vincent Lenders, *armasuisse;* Adrian Perrig, *ETH Zürich;* Martin Strohmeier, *armasuisse*

https://www.usenix.org/conference/usenixsecurity23/presentation/krahenbuhl

This paper is included in the Proceedings of the 32nd USENIX Security Symposium.

August 9–11, 2023 • Anaheim, CA, USA

978-1-939133-37-3

FABRID: Flexible Attestation-Based Routing for Inter-Domain Networks

Cyrill Krähenbühl
ETH Zürich

Marc Wyss
ETH Zürich

David Basin
ETH Zürich

Vincent Lenders
armasuisse

Adrian Perrig
ETH Zürich

Martin Strohmeier
armasuisse

Abstract

In its current state, the Internet does not provide end users with transparency and control regarding on-path forwarding devices. In particular, the lack of network device information reduces the trustworthiness of the forwarding path and prevents end-user applications requiring specific router capabilities from reaching their full potential. Moreover, the inability to influence the traffic's forwarding path results in applications communicating over undesired routes, while alternative paths with more desirable properties remain unusable.

In this work, we present FABRID, a system that enables applications to forward traffic flexibly, potentially on multiple paths selected to comply with user-defined preferences, where information about forwarding devices is exposed and transparently attested by autonomous systems (ASes). The granularity of this information is chosen by each AS individually, protecting them from leaking sensitive network details, while the secrecy and authenticity of preferences embedded within the users' packets are protected through efficient cryptographic operations. We show the viability of FABRID by deploying it on a global SCION network test bed, and we demonstrate high throughput on commodity hardware.

1 Introduction

The current Internet can be viewed as a black box over which endpoints send traffic without detailed knowledge about the network paths taken. While this black box usage is sufficient for many applications, it can lead to performance, compliance, and security issues when the path does not comply with end-users' required network properties.

For example, some applications require that data does not leave a given jurisdiction, which therefore requires that sensitive traffic is only routed through routers operated in certain countries or regions [11]. Another application is time synchronization, which may require packets to be forwarded only over routers with hardware-based time synchronization support (e.g., PTP [20]) [36]. Governments or critical infrastructure operators may further desire to avoid network equipment from certain manufacturers or specific router versions that pose potential security risks. They may thus wish to route traffic over paths consisting of specific, trusted equipment.

Confidentiality of data in the Internet is achieved through encryption. However, data (payload) encryption has limitations. Metadata such as IP addresses can still be observed, enabling censorship and reducing anonymity. The distribution and protection of keys is challenging and weaknesses in ciphers and modes of encryption, advances in cryptanalysis, or quantum computers may render encryption schemes insecure. Moreover, legacy communication protocols might not support encryption at all. Instead of exclusively relying on encryption, confidentiality and integrity can be enhanced by forwarding data over channels built from trusted network equipment (devices and links), which is believed not to eavesdrop on the communication.

This raises the question of how to implement inter-domain Internet path selection for end users based on attested router properties. There are currently two initiatives with similar design goals, however neither of them can provide flexible, policy-compliant, inter-domain Internet path selection based on individual user needs. First, the IETF draft Trusted Path Routing (TPR) [41] from the Remote ATtestation ProcedureS (RATS) working group [22] has proposed to conduct remote attestation measurements between neighboring routers to find trusted links, and thus create a trusted routing plane within a network based on intra-domain routing protocols such as IS-IS or OSPF. Unfortunately, TPR does not offer endpoint path control and focuses on intra-domain communication.

The second approach is the path-aware Internet architecture SCION [8], which allows individual applications, on endpoints, to influence the forwarding path of their traffic. SCION is an Internet architecture that enables endpoint path control and multi-path communication. SCION is now globally deployed, commercially available, and allows endpoints to control forwarding at the autonomous system (AS) level. Still, the level of transparency and control provided by SCION does not suffice for an endpoint to learn relevant information about network equipment in the internal networks of on-path

ASes, as it does not provide attested router properties for the network elements along a selected path.

In this work, we present Flexible Attestation-Based Routing (FABRID), a novel system that supports the fine-grained selection of the properties of all network elements along an inter-domain path. ASes advertise network devices with specific properties and optionally increase their trustworthiness by publishing remote attestation results to prove the existence of the advertised devices.

FABRID extends SCION, inheriting its properties such as path control at the granularity of border routers, robustness against single link failures, and a scalable global route distribution mechanism. We leverage the backward-compatible extensibility of SCION's control and data planes to enable partial deployment of FABRID within the SCION network. Even when some ASes on the forwarding path do not support it, FABRID still provides some benefits. Experience with IPv6 [18], egress filtering, and BGPsec [32] shows that allowing incremental deployment is a necessity for new Internet technologies since convincing ISPs to adopt them is a challenging and time-consuming process.

The main contributions of this work are the following:

- We design an extensible policy language to describe arbitrary router attributes, which supports customizable policies for users and ASes. Leaking sensitive intra-AS data is prevented by restricting AS policies accordingly.
- We enhance SCION's path control to enable path selection based on user-defined policies and use SCION's public-key infrastructure to authenticate information about on-path network devices by the respective ASes.
- We apply efficient measures for authenticity and secrecy of users' packet-carried policy decisions.
- We implement and evaluate a high-speed variant of FABRID, and demonstrate the system's feasibility by deploying it on a global test bed.

2 Background on SCION

We build FABRID on top of SCION [8], a network architecture that provides strong availability, control, scalability, and transparency guarantees. In SCION, autonomous systems (ASes), which constitute the Internet's building blocks, run a dedicated SCION control service responsible for the discovery, dissemination, and validation of routing information. SCION border routers provide one or multiple *external interfaces* to other ASes, connecting the network link between them, and an *internal interface* for connectivity to border routers in the same AS, the control service, and other local endpoints. All other routers inside an AS are referred to as internal routers.

In SCION, ASes are logically grouped into isolation domains (ISDs). Some ASes inside an ISD are core ASes, which

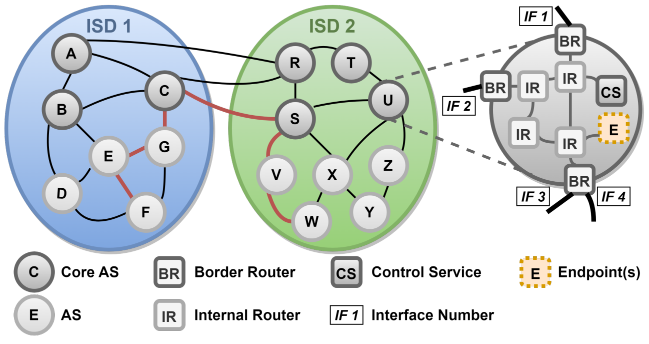

Figure 1: SCION topology example. Assuming the existence of three path segments C-G-E-F, C-S, and S-V-W, these three path segments can be combined into an end-to-end path from a host in AS F to a destination host in AS W.

manage the ISD's trust roots and provide connections to core ASes in the same or other ISDs. Routing in SCION is a per-ISD process for discovering intra-ISD paths. To interconnect ASes in different ISDs, a separate routing process discovers paths between all core ASes. Intra- and inter-ISD routing procedures are implemented through a beaconing process, which is initialized by core ASes disseminating path-segment construction beacons (PCBs). An AS receiving a PCB extends it with local information such as its AS and interface identifiers. An AS signs the PCB and then forwards in to selected neighbors. The AS's public key is certified by SCION's control-plane public-key infrastructure (CP-PKI). SCION enables a diversity of end-to-end paths thanks to the various ways in which up to three path segments can be combined: an up-segment from its AS to a core AS of the same ISD, a core-segment between two core-ASes, and a down-segment between a core- and the destination AS. Shortcuts between up- and down-segments eliminate potential routing inefficiencies.

An endpoint learns segments to the desired destination from the control service of its AS, where the service typically returns a few dozen possible segments. This makes SCION a path-aware networking architecture, as endpoints can influence the forwarding path that their traffic takes through the network on an AS-level. This is in contrast to traditional inter-domain networking, where the network delivers data on paths outside of the endpoints' control.

When sending a packet, the path (a list of ingress/egress interface-pairs) is encoded as packet-carried forwarding state (PCFS) in the packet header, instructing on-path border routers how to forward the packet. PCFS enforces the endpoints' path selection, enables multipath communication, and protects packets from unanticipated re-routing and hijacking attacks. An example SCION topology is shown in Figure 1.

3 Trust Model and System Objectives

In this work, we consider *endpoints* that require communication with other endpoints over paths consisting of devices with specific desired attributes. We intentionally keep the definition of an endpoint abstract—it can refer to a machine, an application, or even a specific flow.

3.1 System Objectives

FABRID's overall objective is to enable endpoints to send traffic over devices or links with desired attributes. We want to support a large variety of use cases, from utilizing specific functionality provided only by certain routers to ensuring that traffic is sent only over trusted hardware. The generalized ideal property of FABRID is the following:

Ideal Objective. *Traffic is only received by devices or links with attributes acceptable to the endpoints.*

In practice, achieving this property is challenging for multiple reasons. Network operators would need to give external entities, e.g., endpoints, insight into communication devices and channels on *all layers*, i.e., on the network-, link-, and physical layer, which might not readily be available. Even given this insight, adjusting the routing of the typically static link-, and physical layer can be difficult. At the same time, routing at the network layer offers more flexibility and is very common, e.g., in traffic engineering, especially given the prevalence of software-defined networking (SDN). Additionally, control over the link-, and physical layer path has limited benefits as the link-, and physical layer path often depends on the network path. Finally, for the trusted hardware use case, all of these devices would require hardware support, e.g., a trusted platform module (TPM). We therefore relax this ideal property as follows:

Realistic Objective. *Traffic is only received by network layer devices (routers) with attributes acceptable to the endpoints.*

This is still a very strong property, as the physical- and link layers are not easily accessible to an adversary. Link layer devices cannot be addressed directly by arbitrary devices on the Internet, and eavesdropping on links is difficult due to physical security measures such as protected premises at ISPs and Internet Exchange Points (IXPs).

To achieve this objective we formulate the following endpoint requirements:

E1 **Path Transparency**: Endpoints can request information on the attributes of on-path devices. Accurate information allows them to select paths to achieve specific objectives.

E2 **Attested Claims**: Claims about router attributes should be attested by the AS (and optionally by the routers themselves) to the endpoints and be non-repudiable.

E3 **Secrecy and Authenticity**: For packets that are meant to follow a path with certain attributes, only legitimate on-path entities should be able to decrypt those attributes embedded in the packet header. Those entities should also verify the authenticity of the attributes and discard packets not passing this verification.

E4 **Path Stability**: A selected path consisting of routers with desired attributes should be stable over time, otherwise traffic might be routed over devices violating the endhost's preference.

E5 **Path Validation**: Even if path stability is ensured by some routing protocol, misconfigured or compromised devices may still violate their forwarding directives. It should therefore be possible for endpoints to verify that their selected path is actually obeyed by on-path routers.

E6 **Fine-Grained Properties**: Specifying desired router attributes should be as flexible as possible, irrespective of endpoints being end hosts, applications, or flows.

Note that some of these requirements are not possible to achieve in the current Internet based on the Border Gateway Protocol (BGP). The most fundamental limitation here is BGP's lack of path stability, whereby routes can unpredictably change at the level of ASes due to rerouting events or hijacking attacks. ASes have fundamentally different requirements:

A1 **Trade Secrets**: An AS should not need to reveal sensitive information about its internal network, but rather be able to decide itself which router attributes to publish.

A2 **DDoS Protection**: Some ASes distribute ingress traffic for the same egress router among multiple paths to achieve better redundancy and higher throughput. Such ASes should have the option to announce router attributes without having to specify a concrete path, as otherwise an adversary can target this path using DDoS.

A3 **Efficient Distribution**: Propagating router attributes to endpoints should be efficient, i.e., incur only low computation and communication overhead.

A4 **Efficient Forwarding**: Sending traffic along routers with specific attributes should not significantly impact performance. In particular, this implies efficient packet generation and reception at the endpoints and high-speed forwarding at routers that scales to today's Internet link capacities. Also, the amount of additional packet meta data should be minimal.

To address the tension between the transparency requirements of endpoints (E1) and what router information an AS is willing to provide (A1), we introduce the concept of policies, which allows ASes to precisely specify the granularity of attributes to be published.

3.2 Policies

In this section, we introduce the concept of *policies*. Policies allow ASes to specify the granularity of the router information they publish (A1), and enable endpoints to validate whether this router information complies with their needs (E1).

Specifically, a **router attribute** is a trait of an intra-AS or border router, such as its manufacturer or hard- and software. A **router policy** is a predicate on a router defined in terms of its attributes. For example "a router is manufactured by *A* or *B*." A router policy may be lifted to a **path policy** by requiring that it holds for every router on a path. An example of a path policy, with which the previous router policy is compliant, is "all on-path routers are manufactured by *A*." In Section 5, we design a language for defining these policies.

An endpoint describes required router attributes using a router policy, which we will refer to as the endpoint's **preference policy**. A preference policy can be used in different ways. For example, it could be defined per-device, per-user, per-application, or even on a per-packet basis (E6). Applications might even be distributed with a predefined policy.

3.3 Trust Model

Which entities endpoints trust is subjective, and we therefore define our trust model from the point of view of individual endpoints. In our model, the granularity of trust is at the level of ASes, i.e., an endpoint may trust some ASes more than others.

We assume that trusted ASes are not malicious: they follow all protocol steps correctly and they implement measures to protect their infrastructure from attacks. However, even highly trusted ASes can make mistakes, and parts of their infrastructure may be misconfigured. We refer to such ASes as *honest-but-clumsy*. In practice, endpoints often have existing trust relationships with some of the ASes, based on business relationships or common legislation. Concrete examples include government offices communicating via multiple ISPs located in their country, companies affected by regulations mandating traffic not to leave specified countries [11], or ISPs offering policy-based path selection for an additional fee. In order for a source endpoint to communicate with a destination endpoint in a remote AS using FABRID, there must exist at least one valid and usable path of trusted ASes.

We also assume that ASes are directly peering such that inter-AS communication does not traverse additional routers and that relevant certificates are secure, i.e., no intermediate entity, such as a certificate authority, in a certificate chain is compromised. Furthermore, communicating endpoints are assumed to trust each other in order to agree on the same on-path router attributes. Our model restricts the capabilities of adversaries on an endpoint-selected path to reading, modifying, injecting, and dropping packets.

Note that we do not make any assumptions about untrusted ASes, which can exhibit arbitrary and even actively malicious behaviour. If FABRID relied on BGP, where traffic can be redirected by malicious off-path adversaries by attacks such as hijacking [7] or blackholing [29], endpoints would additionally need to trust a potentially large set of off-path ASes to not perform such attacks. This observation motivates building FABRID upon SCION, which mitigates re-routing attacks by authenticating routing information using its CP-PKI and by enforcing correct forwarding by means of PCFS.

4 FABRID System Description

FABRID, based on SCION, enables ASes to announce paths along with intra-AS path policies and allows endpoints to select routes based on their own policy preferences. FABRID satisfies the requirements of endpoints and ASes specified in Section 3.1. In particular, an attacker can neither change the forwarding paths and policies chosen by the endpoints nor spoof the path policies announced by the ASes. FABRID introduces minimal processing and state overhead on routers and endpoints and is incrementally deployable.

4.1 Overview

FABRID consists of two main processes. In the control plane, path policies are distributed to endpoints, which compare them for compliance with their local preference policies. In the data plane, endpoints encode the selected policies into their data packets, such that on-path ASes know how to forward them through their internal network. A system overview is provided in Figure 2.

Every participating AS decides which path policies it wants to disseminate. When originating or forwarding a SCION path construction beacon (PCB), the AS extends the PCB with path policies and AS-local identifiers, called policy indices, of internal paths that comply with these path policies. In Figure 2, AS C adds policies P_X and P_Y, AS G adds policy P_Z, and AS E adds all three policies. Endpoints fetch path segments (constructed from the PCBs) from the local control service and filter out paths with untrusted ASes or with path policies that do not comply with their preference policy. If there is no path left, an endpoint can either decide not to send any traffic, or to still use one of the filtered sub-optimal paths. The endpoint E_F in Figure 2 trusts all on-path ASes and decides to send traffic on the selected path despite one AS (AS G) not supporting the desired policy P_X. This could be a sensible decision if P_X corresponds to a "best-effort" policy, for example, PTP support, but might not be acceptable for security-related attributes. After selecting a set of path segments, the contained policy indices are encoded by the endpoint in a SCION packet and used by ingress border routers to forward the packet across an internal path complying with the corresponding policy. For the first AS (AS F in our example), the endpoint can only learn the supported policies, but it can not actively select a

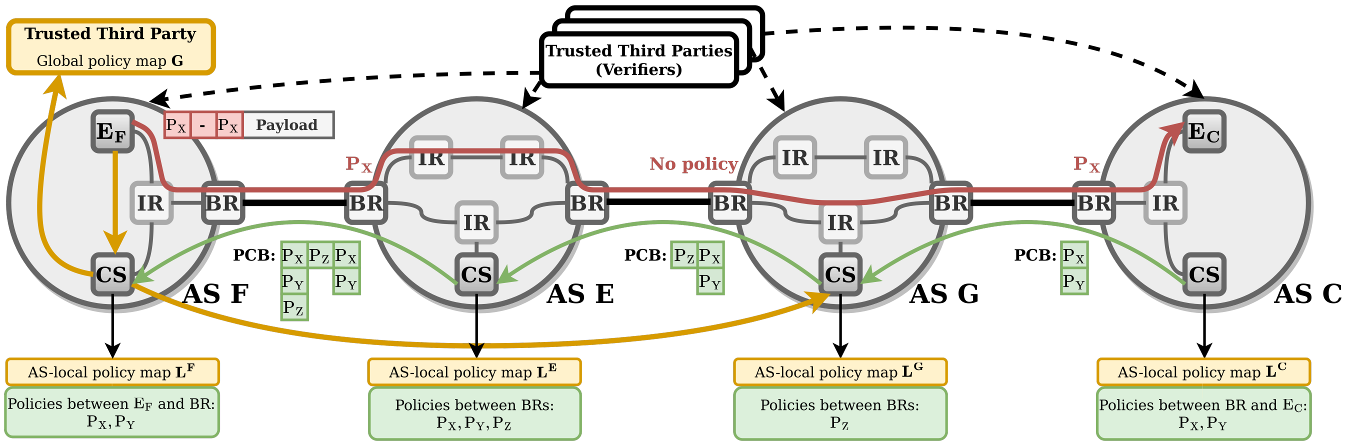

Figure 2: System overview. During beaconing, every AS adds its supported policies to the PCB (green). Endpoint E_F fetches the corresponding path including the added policies from its control service, which resolves and caches unknown policies (orange). For packets destined to E_C, E_F encodes its policy choice P_X in the respective headers, except for AS G, which does not support this policy, and AS F, which supports the policy by default (red). Although not all ASes support P_X, E_F still decides to send its traffic over the path. Optionally, dedicated third parties verify the AS-attested policies through remote attestation.

policy, as there is anyway only one intra-AS path to the egress border router. Therefore, the endpoint does not need to add a policy index to the SCION header for its own AS.

A policy index does not necessarily correspond to a single intra-AS path. The policy index encoded in the packet is evaluated when the packet arrives at the border router. This allows the border router to choose *any* intra-AS path that is compliant with the requested path policy. Hence, intra-AS routes can be changed on-the-fly by network operators, load balancing among multiple intra-AS paths is possible, and fewer details about the intra-AS topology must be revealed.

To prevent malicious entities from negatively impacting the path selection and the endpoint privacy, our solution employs several security mechanisms of SCION. In particular, FABRID encodes path policies in SCION PCBs and makes use of SCION's CP-PKI for verifying signatures from the respective ASes, thereby preventing the spoofing and tampering of policies. Moreover, using EPIC, a data plane extension for SCION, the endpoint can verify that its packet indeed traversed each on-path AS (see Appendix B for details about EPIC). Finally, FABRID authenticates and encrypts the policy index in the packet header, providing privacy regarding an endpoint's policy preferences.

4.2 Policy Announcement

A straightforward approach for disseminating path policies would be for ASes to encode their policy descriptions directly in the PCBs (achieving requirements E1 and A1). However, this is not space efficient since it can substantially increase the size of the PCBs for complex path policies, and therefore violate requirement A3. Moreover, many ASes may have similar path policies, leading to redundancy. We therefore distinguish between globally and locally announced path policies.

Global Announcement Global policies are stored in a global append-only registry managed by a trusted entity such as ICANN [19]. This registry can be described as a map from a *policy identifier* (PID) to the concrete policy description:

$$G : P_G \to S_G\,.$$

Here, P_G is the set of global policy identifiers and S_G denotes the set of global path policy descriptions. The PIDs can be short in practice—we consider a 32-bit number a reasonable representation. The purpose of the global registry is to support globally accessible path policies used by a large fraction of ASes. Entities can then simply refer to the PID instead of a potentially large policy description. Entries in the registry must never be changed or removed to ensure a globally consistent view despite entities caching the registry information.

Local Announcement An AS is not restricted to only use globally defined policies, and an AS X can define its own append-only mapping of policy identifiers:

$$L^X : P_L^X \to S_L^X\,.$$

Here, P_L^X is the set of PIDs defined by AS X, and S_L^X denotes its set of path policy descriptions. The mapping L^X is stored at the control service of AS X. The service is extended to also serve path policy requests based on this information. Irrespective of whether an entity requests the policy mapping from a specific AS or from the global registry, the response must always be authenticated using a signature. The control service caches entries from the global registry and other ASes for its local endpoints to reduce the communication overhead.

4.3 Policy Dissemination via PCBs

ASes disseminate their supported path policies to the endpoints by piggybacking this information on SCION PCBs. This added information describes which path policies are supported on specific ingress-egress interface-pairs. Naturally, the interface pair the PCB is traversing is the most relevant; however, SCION also supports shortcut and peering links [8], and therefore path policies for multiple interface-pairs can be specified. Different path policies can be described depending on directionality, that is, an interface-pair (i, e) need not necessarily support the same policies as the interface-pair (e, i). Furthermore, we allow an AS to announce path policies not only for interface-pairs, but also between an interface and an IP address range inside the AS. This enables endpoints to also learn the policies supported by the first and last on-path AS.

Because endpoints have access to the global map (G), as well as to the local policy map of an AS X (L^X), it suffices for the AS to only add the relevant PIDs to the PCB, instead of the full path policy descriptions. For every (global and local) PID it supports, an AS further defines a short 16 bit, non-zero *policy index* (ψ). The short length of the policy index is motivated by its subsequent use in the data plane, where it will be included in the data packets sent by endpoints. The validity period of a policy index is the same as the validity period of the PCB to which it is added. A PCB can be converted to a path segment and thus be used to send traffic usually for a duration of a few hours, after which an endpoint must get path segments from a more up-to-date PCB. An AS can update a policy index such that it points to a different PID, but with the restriction that no two PCBs with overlapping validity periods include different policy index mappings. Naturally, an AS must ensure that it only announces policies that it can support during the whole validity period of a PCB (E4). However, it is free to decide how to route traffic along internal paths that satisfy the announced policies. Some of the most prevalent routing protocols suitable for this purpose are MPLS [34] and segment routing [12]. Based on the preceding description, we formalize the information added to a PCB by AS X in the form of two maps I^X and D^X as follows:

$$I^X : IFIP^X \rightarrow \mathcal{P}(\psi^X),$$
$$D^X : \psi^X \rightarrow P_L^X \cup P_G,$$

where $IFIP^X = ((IF^X \cup IP^X) \times (IF^X \cup IP^X)) \setminus (IP^X \times IP^X)$, IF^X is the set of interfaces, IP^X the set of IP address ranges, and ψ^X the 8 bit space of valid policy indices of AS X. $\mathcal{P}(\psi^X)$ denotes the power set of ψ^X.

To prevent manipulation by unauthorized entities, this path policy information, i.e., I^X and D^X, is signed by AS X (E2). Figure 3 shows the format of a PCB and how it is extended with signed policy information. Although policy information, in the form of the two maps I^X and D^X, is added in the unsigned part of the PCB, this information cannot be modified, since a cryptographic hash of the map content is included in

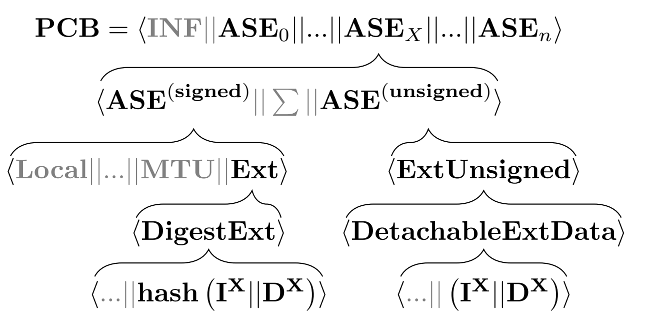

Figure 3: PCB format containing path policy information. FABRID extends the DigestExt and DetachableExtData fields of a SCION PCB by hash($I^X||D^X$) and $I^X||D^X$, respectively.

the signed part of the PCB. Even though the two maps I^X and D^X can be encoded efficiently in terms of space, they can significantly increase the size of a PCB in case (i) there are many on-path ASes adding information to the PCB, (ii) many different policies are announced, or (iii) some ASes have many pairs of interfaces or IP subnets with support for policies. Including only the hash of the maps in the signed part of the PCB allows removing the actual maps from the unsigned part of the PCB if needed (A3). In this case, the endpoints need to actively fetch the maps from their local AS, which fetches them from the corresponding ASes and caches them to reduce network overhead. They can efficiently verify the authenticity of I^X and D^X by computing their hash and comparing it against the hash included in the PCB for AS X.

4.4 Endpoint Policy Selection

An endpoint assembles the desired end-to-end path from up-, core-, and down-segments constructed from the PCBs fetched from the local control service of its AS (see Section 2).

As every PCB also contains the path policies that the corresponding ASes support, the endpoint can verify whether the corresponding segment satisfies its requirements (E1). If some of the ASes that are part of the segment are not trusted, or announce path policies that are not in accordance with the endpoints' needs, the endpoint either decides to accept such untrusted ASes or non-optimal path policies and to still use the segment, or to take advantage of SCION's path-awareness and try to find another segment that actually satisfies its preference policy. An endpoint receives from the control service, by default, many possible segments to reach the requested destination AS, and therefore it has several possible paths and policies that it can readily choose from.

To achieve policy selection at the packet level (E6), the endpoint communicates which policy should be satisfied to each on-path AS, except its local AS, in the form of policy indices (a policy index of zero signifies no specific desired policy). An endpoint can only select a single policy index per on-path AS; an AS can announce local path policies that are conjunctions of other path policies. Upon reception of a data packet, an ingress border router parses the policy index and maps it to an internal path satisfying the corresponding path policy. This mapping, which can be implemented efficiently using a hash table (A4), adds overhead to the processing time of every packet, which is however negligible even for ASes supporting hundreds of policies (Section 6). In particular, its scalability does not depend on the number of endpoints (hosts, applications, or flows) actively sending traffic.

The codomain of the mapping and any further actions executed on a packet depend on the intra-AS routing protocol in use. In MPLS, for example, a policy index is mapped to a set of labels, each corresponding to a single label switched path (LSP). The border router places the selected label in the packet header, which is later again removed at the egress border router. To avoid packet reordering when multiple intra-AS routes are available, the border router can use hashing to always map packets of the same flow to the same intra-AS route. The intra-domain routing protocol and the soft- and hardware of internal routers need not be modified.

If the packet contains a policy index that is not supported, the border router sends back an authenticated control message to the endpoint. Because a border router only receives policy-enabled packets that are source-authenticated (Section 4.5), control messages can not be abused by an attacker to launch reflection DoS attacks against other endpoints.

Upon request, the control service also provides policy information for the intra-AS path between two endpoints of the same AS. Because intra-AS communication does not use the SCION protocol, an endpoint cannot select the preferred policy; however, it learns from the control service which path policies are guaranteed to be satisfied when sending traffic within the AS to the other endpoint, including their validity period. By using short periods, the AS can propagate routing configuration updates quickly to the endpoints. The path policies supported on the route between the endpoint and the egress border router are part of the PCB.

An endpoint's preference policy can be set in various ways, depending on the concrete use case. There can be a single policy for the entire device, or an application-specific policy, e.g., for an application sending sensitive corporate data. Per-application policies are set before the connection establishment is initiated, similar to TAPS [44], which provides fine-grained control over transport protocol parameters to applications. Finally, different policies can be set for individual packet flows or even on a per-packet basis, but requires modifying the endpoint's network stack.

4.5 Securing Policy Indices

The policy indices in data packets must be authenticated, otherwise malicious entities could modify them and thus violate the endpoint's preference policy. Furthermore, indices in the packets should be hidden, such that an index is only accessible by the endpoint and the on-path AS that announced it.

To achieve authenticity and secrecy of the policy indices, we leverage the fast key derivation provided by DRKey [24, 35] and a modified version of EPIC [27]. In EPIC, a source host includes short per-packet hop validation fields (HVFs) in the packet header, which are subsequently verified by the on-path border routers to assess the authenticity of the packet source. Through its path validation feature, EPIC furthermore allows the source and destination endpoints to verify for each packet that it has indeed traversed the ASes of the correct, i.e., the previously selected, path. Therefore, EPIC augments FABRID to satisfy requirement E5. Background on DRKey and EPIC can be found in Appendices A and B and our modification to EPIC that also supports authenticity and confidentiality of the policy indices is described in Appendix C. Through these modifications, FABRID also satisfies requirement E3.

With this solution, unauthorized entities can neither infer nor modify the policies chosen by endpoints. Moreover, it is not possible to detect whether an endpoint chooses a policy at all, as an *encrypted* zero and non-zero index are indistinguishable. Irrespective of its value, a policy index is always 16 bits—also in its encrypted form, which is achieved by one-time pad encryption. Due to the per-packet cryptographic operations, the encrypted policy indices and hop validation fields change for each packet sent, and hence packets with the same policy index also cannot be distinguished.

4.6 Router Attestation

Remote attestation on routers that are equipped with a trusted platform module (TPM) can ensure that the device is running the intended software and is not compromised—and is thus considered "trustworthy" [22]. If every single device on the forwarding path is verified to be trustworthy, then the complete forwarding path is considered trustworthy.

Router attestation can improve FABRID by strengthening the path policy claims of an AS (E2). This is the case when the attestation result provides enough information to conclude that the routers are compliant with an endpoint's preference policy. This information can be used to augment the trust placed in an AS, as an orthogonal measure to the signed path policies in the PCBs. While such router attestation within an AS cannot generate a proof that traffic is actually routed over these devices, it can be used to determine the veracity of path policies claimed in PCBs by proving that a path-policy-compliant intra-AS path *exists*. If an AS is trustworthy, it will comply and send traffic over these policy-compliant routers.

An AS can release the attestation results to a (trusted) third party that then verifies the existence of a path-policy-compliant intra-AS path on behalf of the users. Thereby, the endpoint does not receive the actual attestation result. The third party instead returns a signed statement stating that all path policies were validated based on the AS' attestation results. Alternatively, attestation results could be published on

Table 1: The different sorts and their respective carrier sets.

Sort	Carrier Set	Type
Manufacturer (M)	$\mathcal{M}$	private enterprise number (int) [21]
Software Component (C)	$\mathcal{C}$	unique ID
Tag (T)	$\mathcal{T}$	string
Tag Issuer (I)	$\mathfrak{I}$	URI (string)
Name (N)	$\mathcal{N}$	string
Version (V)	$\mathcal{V}$	version scheme (string)
Router (R)	$\mathcal{R}$	unique ID
Path (P)	$\mathcal{P}$	unique ID

a public append-only log. However, publicly sharing attestation results can be problematic as this might reveal sensitive information about the AS-internal network and is thus only done if approved by the respective AS (A1).

5 Policy Specification

In FABRID, desirable router attributes are specified by router policies, which are predicates on a router's configuration. When this predicate is satisfied for a router's configuration, the policy is said to comply with the configuration. By specifying policies as formulas in a sorted first-order-logic (FOL) referencing relevant router attributes, users can define arbitrarily complex policies. The system remains extensible, since this language can be enriched with new router attributes, should they become relevant in the future.

5.1 Syntax and Semantics

Policies are encoded in a sorted FOL with equality as formulas with free variables. In addition to the general FOL syntax, given in Appendix D.1, we define the sorts M, C, T, I, N, V, R, and P (see Table 1), the function symbols `tag` (of type $C \to T$), `issuer` (of type $T \to I$), `name` (of type $C \to N$), `version` (of type $C \to V$), `manufacturer` (of type $R \to M$), the predicates `onPath` (of type $P \times R$) and `software` (of type $R \times C$), and the comparison predicates $<$, $\leq$, $\geq$, and $>$ (of type $V \times V$).

A path consists of a set of routers (ignoring their order) and each router is described by its router setup. A router setup specifies the router's manufacturer and the router's software stack, which consists of software components. A software component is specified by a name, a version, and a globally unique tag, issued by a tag issuer. Appendix D.2 contains a more detailed description of the semantics and interpretation.

5.2 Policy Definition

Both user preference policies (E1) and path policies (A1) are defined as router policies. However, they are used for different purposes, since a user preference policy defines all acceptable router setup(s) of a user, whereas a path policy defines all possible router setups on a path.

We define a router policy as an open formula of the form $Pol(r)$, with one free variable r of sort R. A router policy accepts a router setup iff the formula $Pol(r)$ is satisfied ($\models$) in an interpretation I with respect to an assignment α, which assigns a router to the free variable r, where $\alpha : \{r\} \to \mathcal{R}$.

$$I, \alpha \models Pol(r)$$

A **path policy** ($PathPol_p$) advertised for a path p must accept the setups of all on-path routers (R):

$$\forall r' \in \mathcal{R} : \texttt{onPath}(p, r') \to PathPol_p(r')$$

An example of a path policy covering routers with a specific version v of a software s issued by issuer i and a manufacturer m, using `manu()` as a shorthand for `manufacturer()`, is

$$\begin{aligned} PathPol_p(r) \;:=\; & \texttt{manu}(r) = m \land \exists c \in \mathcal{C} : \texttt{software}(r, c) \\ & \land \texttt{name}(c) = s \land \texttt{issuer}(\texttt{tag}(c)) = i \land \texttt{version}(c) = v \end{aligned}$$

A **user preference policy** $PrefPol_u(r)$ describes acceptable on-path router setups of user u. The user automatically decides to accept or reject a path p based on its preference policy and the path policy $PathPol_p(r)$. If $PathPol_p(r)$ is contained in $PrefPol_u(r)$, then u accepts p, otherwise u rejects p.

An example of a user preference policy that requires routers from either manufacturer m_1 or m_2, where a critical software (s_{crit}) was recently updated (at least version $v_{\min}$), is

$$\begin{aligned} PrefPol_u(r) \;:=\; & (\texttt{manu}(r) = m_1 \lor \texttt{manu}(r) = m_2) \\ & \land \forall c \in \mathcal{C} : (\texttt{software}(r, c) \land \texttt{name}(c) = s_{\text{crit}} \\ & \land \texttt{issuer}(\texttt{tag}(c)) = i) \to \texttt{version}(c) \geq v_{\min} \end{aligned}$$

Determining whether a path policy is contained in a user's preference policy is equivalent to determining query containment. Note, however, that query containment can not always be evaluated efficiently. In general, query containment in FOL is known to be undecidable [30]. Even in a restricted subset of FOL, namely FOL formulas consisting of a disjunction of conjunctive queries, query containment is NP hard [6]. We expect that most path policies are relatively simple, and that user preference policies are not overly complex either. Additionally, the outcome of complex query containment evaluations can be cached locally with minimal overhead as long as the policies are in use. As discussed in Section 4.3, path policies are not shared directly in PCBs, but instead fetched on demand by the endpoint from the global policy repository or the local repository of the respective AS, while preference policies are defined locally and never shared with other entities.

6 Implementation and Evaluation

In this section, we demonstrate that FABRID works efficiently in practice. Parties interested in this research can review our implementation and modify it according to their needs with minimal effort. Because our per-packet authentication and encryption operations pose an additional overhead to the sender and the on-path border routers, we also implement and evaluate a high-speed version of those components. For a path consisting of four ASes and packets with payloads of 1000 B, the sender and the router achieve 3.29 Gbps and 14.62 Gbps per processing core, respectively. The performance increases linearly with the number of cores. We also experimentally demonstrate the feasibility of TPM-based remote router attestation for use in FABRID.

6.1 Deployment in SCIONLab

We implement FABRID in SCIONLab [26, 37], a global network testbed that enables research and experimentation with SCION. Users can join SCIONLab through one or more SCION ASes. While some ASes are connected through dedicated links, the majority of inter-AS links are IP overlays. For our purposes, we set up two ASes and connect them to SCIONLab at two different access points, each in a different country. The first AS P is modified such that the control service extends beacons with its set of policy indices. We also adapt its border routers, such that they simulate intra-AS routes with different latencies. Our second AS S contains a SCION-enabled endpoint, which we modify such that it prompts the user to not only select a path, but also one policy per policy-aware on-path AS. The endpoint then creates data packets carrying the chosen policies. We implement this by means of a hop-by-hop extension [3], which is ignored by all on-path ASes that do not support FABRID, but parsed and handled by AS S. Thus no changes are necessary in SCION ASes that are outside of our control. For the evaluation, we measure the round-trip time of control message packets for different policies selected by the user. We took precautions to not harm any real network infrastructure: Our experiments were directed at specific benchmarks and small in scale. With one packet per second, our evaluation adds negligible overhead, and thus does not negatively impact other parties. We obtained permission from the SCIONLab operators to conduct our measurements. As an alternative to this setup, we also provide a system for simulation-based evaluation allowing the execution of arbitrary Internet topologies on a single machine.[1] This supports the fast verification of our results, and low-effort modifications to test new ideas.

[1] https://github.com/mawyss/scion/tree/policy_routing

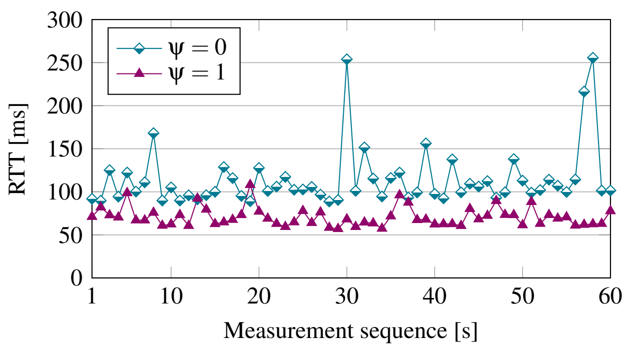

Figure 4: Round-trip time (RTT) measurements for different selected policies over a period of 60 s conducted at an endpoint in source AS S for a 5-hop path going through on-path AS P. A policy index (ψ) of zero and one corresponds to the standard and low-latency intra-AS route, respectively.

6.2 High-Speed Implementation

To secure the policy indices in the packet header, we leverage DRKey [24, 35], a protocol to efficiently derive symmetric keys, and EPIC [27], a system for fast per-packet source authentication and path validation in SCION, modified to ensure the authenticity and confidentiality of the policy indices. To demonstrate that our modifications to EPIC still allow for high-speed packet processing, we implement and evaluate the components affected by our changes, namely the source endpoint and the border routers (the EPIC protocol running on the destination endpoint is not modified). Such an evaluation is important because we encrypt, authenticate, and verify the policy-indices on a per-packet basis, which must be fast enough for practical deployments. Our modifications are described in Appendix C. We implement the source endpoint mechanism to create per-packet validation fields and the corresponding verification procedure at the border router using Intel DPDK [10], instantiate MAC computations with AES-CBC-MAC, and rely on Intel AES-NI [15] to speed up AES block cipher computations. Our testbed consists of a commodity machine (Intel Xeon, 2.1 GHz), which executes our implementation that is to be evaluated, and a Spirent SPT-N4U, which serves both as bandwidth generator (when evaluating the border router) and bandwidth monitor (when evaluating the source endpoint). Both machines are connected by four 40 Gbps bidirectional Ethernet links. We evaluate both implementations (endpoint, border router) individually as they never run at the same time on the commodity machine.

6.3 Evaluation

We now present our results of the RTT measurements in SCIONLab and the high-speed evaluation of the policy-enabled sender and router. Additionally, we evaluate the control-plane overhead and TPM-based remote attestation.

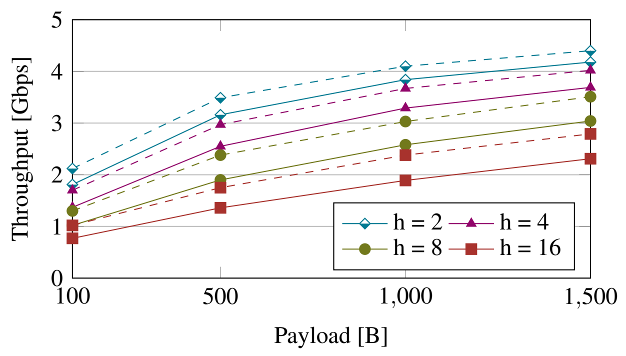

Figure 5: Source endpoint packet generation performance for different number of AS-level hops (h) and payloads, using one CPU core. Dashed lines correspond to the original EPIC without support for policies.

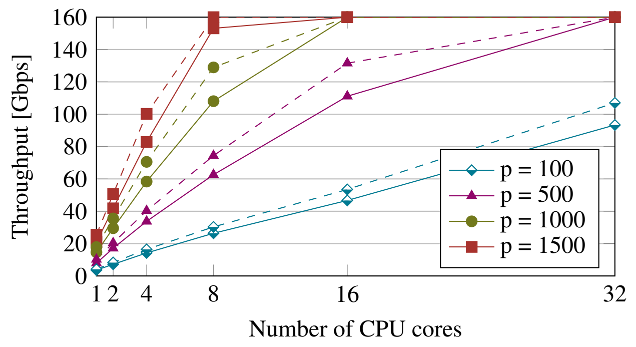

Figure 6: Border router packet forwarding performance for various payload sizes and different number of CPU cores. Dashed lines have the same meaning as in Figure 5.

RTT Measurements In our SCIONLab setup, AS *P* announces two policy indices, where index zero denotes a standard intra-AS path, and index one a low-latency path. We choose this type of policy because, in contrast to router attributes such as manufacturer or software version, latency is a characteristic directly measurable by the communicating endpoints. The results of our experiments performed by an endpoint in AS *S* are shown in Figure 4. While packets traversing the standard intra-AS route of on-path AS *P* travel for 114 ms on average and suffer from high jitter, packets carrying a policy index equal to one take on average 70 ms per round-trip and never exceed 110 ms.

Endpoint Evaluation Figure 5 shows the evaluation results for a single CPU core of the source endpoint. When sending packets with a payload of 1000 B on a path consisting of four AS-level hops (the average path length in today's Internet [17, 28, 43]), the endpoint can generate traffic at a rate of 3.29 Gbps. As expected, the throughput increases with the size of the packet payload and decreases with the number of on-path ASes, where the latter is caused by the per-AS encryption and authentication operations. Although one core of the source endpoint already provides particularly high performance, the results can be scaled linearly with respect to the number of CPU cores dedicated to the generation of policy-enabled traffic. For payloads of 1000 B, FABRID is 7-20 % slower compared to the original version of EPIC, depending on the number of ASes on the path. Because the endpoint can still generate traffic at gigabit rates even on a single core, this overhead does not impact its practicality.

Router Evaluation The results of the border router evaluation are given in Figure 6. The router's forwarding performance depends on the size of the packet payload and the number of CPU cores used, but is independent of the number of on-path ASes, because the router only validates the packet header fields dedicated to its own AS. For instance, using 16 cores and processing packets with a payload of 1000 B, the router can handle the full capacities of all its links, achieving efficient forwarding (A4) with a throughput of 160 Gbps.

From fine-grained timing measurements we discovered that the router's per-packet processing overhead is only slightly higher than the standard EPIC router operations. In particular, checking the authenticity of the encrypted policy index takes additional 15 ns, decrypting the encrypted policy index amounts to 52 ns, and looking up an intra-AS path corresponding to the policy index increases the processing overhead by 35 ns when 1000 indices are supported in total. This corresponds to a decrease in throughput of ~17 % compared to the original version of EPIC. Due to EPIC's efficient design, this relative overhead is noticeable. However, in real-world deployments, this is unproblematic because (i) the absolute number of cores needed to compensate for this overhead is still low and (ii) the linear scalability of the border router still allows ASes to deploy FABRID in various environments with different network capacities. The router throughput and latency are independent of the number of communicating endpoints or their selected policies, since for every packet the same operations are executed. The number of policy indices stored at a router only affects intra-AS path lookup, which is efficient due to its implementation based on a hash table.

PCB Size Overhead The communication overhead imposed on the PCB by a single AS is shown in Figure 7, where policy information is encoded using protocol buffers [14]. For a core AS announcing 500 interface-pairs with five policies each, with a total of 100 supported policies, the overhead is 7.5 kB for the map I plus 0.8 kB for the map D. If the policy information is detached, there is a overhead of 18 B.

Remote Attestation Through measurements on a CISCO NCS 540 device [9], i.e., a router with TPM support, we show the feasibility of the remote attestation proposal from Section 4.6. Figure 8 shows the request/response size and processing time overhead to fetch Platform Configuration Register (PCR) measurements from a single router. These measurements are, among others, used to infer which soft-

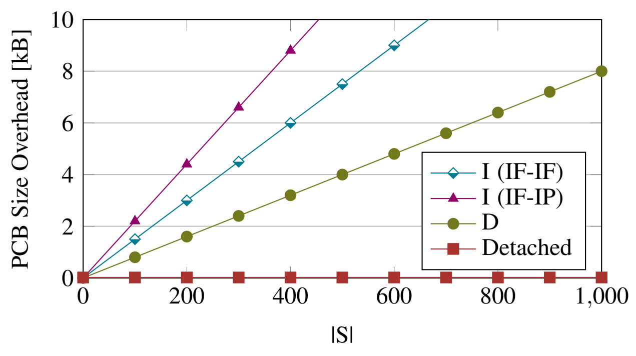

Figure 7: Per-AS PCB size overhead in kilobytes introduced for the maps I, D (see Section 4.3), and the detached extension. The x-axis is the number of elements in set S, which is a subset of either $\mathrm{IF} \times \mathrm{IF}$, $(\mathrm{IF} \times \mathrm{IP}) \cup (\mathrm{IP} \times \mathrm{IF})$, or ψ^X. The number of announced policy indices per IF-IF or IF-IP pair is five.

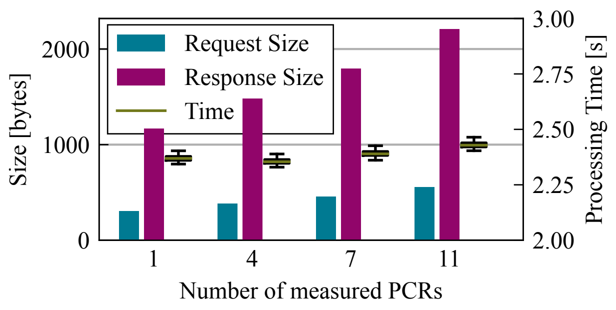

Figure 8: Request and response sizes and processing times for fetching various numbers of PCR measurements.

ware is running on this router. The request and response sizes scale linearly with the number of measured PCRs, but never exceed 2.2 kB. Retrieving PCRs takes around 2.4 s and thus policy-violating routers can quickly be detected. A verifier can validate the trustworthiness of a network in a few seconds by attesting multiple routers in parallel. To prevent abuse, an AS enabling remote attestation must explicitly authorize verifiers to use that feature. The AS might implement additional security measures such as rate-limiting of requests.

7 Security and Scalability Analysis

We now analyze FABRID's security and scalability.

7.1 Security

We first discuss how FABRID satisfies the security requirements of endpoints and ASes (E2, E3, E5, A1, A2), and then show that the Realistic Objective from Section 3.1 is achieved with respect to the trust model described in Section 3.3.

Attested Claims (E2) Claims about the mapping of interface or IP pairs to policy indices (I^X) and the mapping of policy indices to policy identifiers (D^X) are attested by the respective AS through a signature contained in the PCB. When requesting the mapping from local policy identifiers to FOL policy descriptions of some AS X (L^X), a signature over this mapping is included in the response. Similarly, the mapping from global policy identifiers to FOL policy descriptions (G) is signed by the trusted third party. The use of signatures to authenticate the policy information ensures the authenticity and non-repudiation of policy information, which can be confirmed by every entity possessing the corresponding public keys. Furthermore, L^X and G are append-only, so any modifications to existing entries can trivially be detected. Finally, claims about device properties are verified by trusted third parties through remote attestation (see Section 4.6). Those parties vouch for the correctness of the information extracted from the attestation results and published to ASes and endpoints.

Secrecy and Authenticity (E3) The source of the packet, i.e., the three-tuple consisting of ISD, AS, and IP address, is authenticated by EPIC based on symmetric keys derived by DRKey. Our customized adaption of EPIC further ensures the authenticity and confidentiality of the policy indices stored in the header of data packets. This prevents on-path adversaries from tampering with the selected policies. The default policy index of zero, which indicates no preference regarding any path policies, is indistinguishable from any other policy index in their encrypted form. Moreover, a policy index is encrypted on a per-packet basis, and therefore an adversary can not infer whether any two packets carry the same policy indices.

Trade Secrets (A1) FABRID also preserves an AS provider's trade secrets, as the kind of network information published can be chosen by the ASes individually and at fine granularity. Thus, any undesired leakage of sensitive topology information can always be prevented. Even if, for example, an AS announces a policy corresponding to recently patched devices, an attacker cannot infer that traffic sent over paths with different policies will be sent over devices with known vulnerabilities. However, it is the provider's responsibility to prevent unintended leakage.

Path Validation (E5) Path validation provided by EPIC allows an endpoint to verify for every packet sent, whether it has actually traversed all desired on-path ASes. This prevents the violation of the endpoint's forwarding directives by misconfigured entities that are part of our trust model, i.e., honest-but-clumsy ASes. EPIC achieves non-probabilistic guarantees, while significantly outperforming other path validation protocols in terms of communication overhead [27].

DDoS Protection (A2) FABRID does not force ASes to announce intra-AS paths directly, as this could be abused for targeted DDoS attacks. Instead, ASes only publish path policies, which can be mapped to multiple intra-AS paths, allowing for better distribution of ingress traffic. Because the source and policy index of every packet are authenticated, an on-path AS can implement further measures to protect against DDoS attacks such as rate-limiting based on the packet's source and even the selected policy index.

Summary (Realistic Objective) To show that FABRID achieves the Realistic Objective from Section 3.1 given our trust model, we assume that a path consisting of trusted ASes exists between an endpoint and its communication peer. If this is not the case, the endpoint does not send any traffic, and the objective is trivially achieved.
Because the policy claims are all attested by the respective ASes, the endpoint can be sure that those claims have not been modified by unauthorized parties. However, according to our trust model, trusted ASes are considered honest-but-clumsy. An AS can be potentially misconfigured to (i) announce wrong policies in the PCBs, to (ii) forward packets arriving on the ingress interface along an internal path violating the specified policies, or to (iii) forward packets on a different inter-domain path than defined in the PCFS.
The first misconfiguration alone is unproblematic, since an ingress border router will not forward traffic carrying unsupported policies. To mitigate the combination of the first two misconfigurations, the trusted third party verifiers detect through remote attestation that the announced policies are indeed implemented. Lastly, the endpoint can verify through path validation that the selected forwarding path is indeed followed by all on-path ASes. Therefore, we conclude that traffic is only sent along routers satisfying the endpoint-selected policies, i.e., routers with acceptable attributes, hence FABRID indeed achieves the Realistic Objective from 3.1.

7.2 Scalability

FABRID proposes modifications to a global, inter-domain routing infrastructure and must thus work efficiently at scale (A3 and A4). Internet-wide scalability of the underlying SCION architecture and EPIC has been analyzed in previous work [25, 27]. We therefore discuss only the *additional* processing, network, and state overhead of FABRID.

FABRID's most critical component for scalability is the data plane of endpoints and border routers. Section 6.3 shows the low processing overhead compared to EPIC despite using per-hop encrypted and authenticated policy indices in data-plane packets. The packet header overhead for data-plane packets is linear with the AS–path-length (16 bit per AS), which has little effect on the goodput. Finally, border routers need to store the policy index to intra-AS path mapping, which easily fits into memory.

Regarding the control plane, the additional network overhead is as low as one hash per on-path AS due to the use of detachable extensions in PCBs. The inter-domain network overhead for fetching detachable extension data and policy descriptions is greatly reduced through caching at the *local* control service, only requiring intra-AS communication from endpoints. The remaining overhead, such as the control service configuring intra-AS paths or distributing policies to border routers, depends only on the size of the intra-AS topology and thus scales independently of the global network.

8 Discussion

In this section, we discuss limitations and challenges related to the detection of misbehavior and real-world deployment, and potential extension beyond our core design.

Detecting Misbehavior We distinguish between two types of router characteristics: (i) quantifiable characteristics that can be measured by endpoints, like hard-/software feature support, and (ii) unquantifiable characteristics that are difficult or impossible to measure, like security properties, location, or jurisdiction. ASes might not conform to their announced path policies, e.g., due to economic incentives. Detecting that an AS does not conform to its announced unquantifiable router attributes is very challenging and thus restricts endpoints to rely on, i.e., assume, the correctness of policies announced by trusted ASes as described in Section 3.3. In reality this assumption does not always hold. For example, some users might not trust their ISPs, in which case FABRID's objectives are inherently impossible to achieve. Still, there are many scenarios where the endpoint trusts on-path ASes regarding their unquantifiable router characteristics. A government agency, for example, is likely to trust providers in the same country, but will not necessarily trust foreign router manufacturers.

For preference policies concerning quantifiable attributes, the trust model can often be relaxed. In this case, an endpoint can detect misbehavior and switch to a different policy-compliant path. An endpoint might even be able to pinpoint the misbehaving AS by probing multiple policy-compliant paths using a network tomography approach [23].

Deployment A large number of network protocols have never seen major real-world deployment despite their usefulness. RFC 8170 [39] contains several design recommendations for the deployment and transition of new network protocols: clear incentives for early adopters, an incremental deployment model, estimation of the total cost including a way to bill benefactors, and extensibility. FABRID must also overcome these deployment challenges to achieve widespread global adaption. Both the incremental deployment (Section 4.3) and extensibility (Section 5) of FABRID have been discussed already. ASes should be able to bill the endpoints profiting from policy-enabled paths, for example based on an on-demand or flat-rate model. One possible solution is for an AS to bill its neighboring AS based on the quantity and quality of policy-compliant paths consumed by the aggregate of all flows. An endpoint is then billed by its own provider or the provider offers this service for free to attract customers. A more elaborate approach is to let endpoints directly purchase the use of path policies from the desired ASes, which is more complex and requires prior communication with all on-path ASes, however. Interestingly, in addition to the mitigations discussed in Section 7, billing further helps preventing DDoS attacks as it imposes monetary costs on the attacker.

Since FABRID extends SCION, which is seeing real-world deployment [4, 25, 38], and it does not require additional capabilities compared to SCION (e.g., SCION border routers already use AES for per-packet operations), existing SCION components do not need to be replaced. Additionally, the protocols we rely on are already either partially (EPIC [2]) or fully (DRKey [1]) implemented in SCION. However, an AS deploying FABRID must first become SCION-enabled, which requires changes to its inter-domain routing and its border routers. Additionally, endpoint devices may lack capabilities, such as native AES support. Finally, in addition to the overhead discussed in Section 6.3, large ASes with complex internal networks and many internal and border routers may incur additional management overhead for keeping a precise inventory, deciding on which topology information to reveal, and incorporating their traffic engineering policies.

Bidirectional Policy Selection In FABRID, an endpoint chooses the policy on the forward path, while the destination endpoint chooses the policy on the backward path. In some cases, an endpoint may wish to select both a forward and a backward policy-compliant path.

This can for example be achieved through a negotiation protocol, or, if the destination endpoint has no policy preferences (e.g., a publicly available service), by letting the source endpoint dictate the policies to be used by the destination endpoint. Because the policy indices are encrypted, the endpoints must trust each other as they can not verify the policies chosen by their peer in the headers of the received packets.

One possible mechanism to ensure policy compliance in both directions is for the source endpoint to piggyback the policy indices for the return path in its packets, where the indices are authenticated and encrypted using a shared symmetric key between the end hosts provided by DRKey. With an additional flag, the source endpoint can signal the destination not to send any return packets in case it does not want to use those indices. This approach requires identical forward and return paths, but guarantees that traffic on the return path (i) traverses source-trusted ASes only and (ii) is compliant with the source's policies. Working out this mechanism in full detail and evaluating other alternatives for achieving bidirectional policy compliance remain as future work.

9 Related Work

Platypus [33] is a source-routing protocol allowing endpoints to compose paths from multiple Internet routes through intermediate waypoints. Platypus enables fine-grained control over the forwarding path even for intra-AS routes. This approach is undesirable for ASes that do not want to disclose their internal topology. Moreover, Platypus requires an endpoint to readily know the desired waypoints.

Alcatraz [5] prevents malicious exfiltration, alteration, and forwarding of data on network devices by leveraging trusted execution environments provided by Intel SGX. Alcatraz assumes an environment controlled by a single operator. The throughput achieved on routers was below 1.5 Gbps per core, preventing use of Alcatraz in corporate networks and data centers. Alcatraz could potentially be used as an intra-AS data plane for FABRID according to predefined rules based on the path policies of the AS.

Trusted Path Routing (TPR) [41] allows to enforce that sensitive traffic traversing a network is forwarded only through trustworthy devices. What comprises sensitive traffic is specified through IP address ranges, which are associated with a trusted topology. To decide which devices to include in this topology, adjacent devices equipped with a Trusted Platform Module (TPM) mutually verify their trustworthiness through remote attestation [40]. TPR could complement FABRID, i.e., to provide intra-AS forwarding over a trusted topology consisting of policy-compliant attested routers.

Besides EPIC, there are several other systems for source authentication and path validation. OPT [24] and ICING [31] have a lower goodput ratio (ratio between goodput and throughput) compared to EPIC, which is due to longer packet header fields. Furthermore, ICING causes significantly higher processing overhead at routers than EPIC. PPV [45] does not provide path validation to the source but enables the destination to probabilistically validate parts of the path. Similarly, with Hummingbird [16], routers only sample packets probabilistically but based on symmetric keys shared between neighboring routers. MASK [13] only authenticates the packet's source to a single on-path router.

10 Conclusion

Properties desired from on-path forwarding devices are inherently subjective: Some applications require specific router capabilities, while others want to forward traffic only along trusted manufacturer devices, as they might not consider encryption alone sufficient for confidentiality. This motivates the need for (i) transparent, attested, and non-repudiable claims about on-path device properties, (ii) path selection by individual users, end hosts, applications, or even flows, as well as (iii) protection against hijacking attacks.

The main security takeaways of this work are that the above properties can be achieved by using an expressive policy language on router properties, without compromising AS and endpoint privacy, performance, and scalabilty. Namely, we design and implement a system for inter-domain router-based path selection, including a flexible policy language allowing individual router policies per endpoint, that achieves these properties by leveraging the security and extensibility of SCION, remote router attestation, and efficient path validation. FABRID creates exciting opportunities for ISPs (e.g., creation of new business models and services) as well as for end users (e.g., path selection based on fine-grained policies).

Acknowledgments

We would like to thank Ralph Mohnhaupt, Bernard Botteron, Chennakesava Reddy Gaddam, Eric Voit, Rakesh Kandula, Vitus Andreoli, Annu Singh, and Georg Aebi from CISCO for providing us with TPR-capable routers and for their support regarding the setup of the measurement testbed. We further thank Juan Angel García-Pardo for integrating the routers into our network. We are grateful to the anonymous reviewers for their helpful feedback on the manuscript, and our shepherd for guidance during the preparation of the final version of the paper. We gratefully acknowledge support from ETH Zurich, the Zurich Information Security and Privacy Center (ZISC), armasuisse Science and Technology, and the Werner Siemens-Stiftung (WSS) Centre for Cyber Trust at ETH Zurich.

References

[1] Anapaya Systems. Dynamically Recreatable Key (DRKey) Infrastructure Documentation. https://scion.docs.anapaya.net/en/latest/cryptography/drkey.html, 2022.

[2] Anapaya Systems. EPIC Design Documentation. https://scion.docs.anapaya.net/en/latest/EPIC.html, 2022.

[3] Anapaya Systems. SCION extension header specification. https://scion.docs.anapaya.net/en/latest/protocols/extension-header.html, 2022.

[4] Anapaya Systems. SCION-Internet: The New Way To Connect. https://perma.cc/J3T7-KYL9, 2022.

[5] Daniele E. Asoni, Takayuki Sasaki, and Adrian Perrig. Alcatraz: Data exfiltration-resilient corporate network architecture. In *Proceedings of the IEEE International Conference on Collaboration and Internet Computing (CIC)*, October 2018.

[6] Ashok K. Chandra and Philip M. Merlin. Optimal implementation of conjunctive queries in relational data bases. In *Proceedings of the Annual ACM Symposium on Theory of Computing (STOC)*, 1977.

[7] Shinyoung Cho, Romain Fontugne, Kenjiro Cho, Alberto Dainotti, and Phillipa Gill. BGP hijacking classification. In *Proceedings of the Network Traffic Measurement and Analysis Conference (TMA)*, 2019.

[8] Laurent Chuat, Markus Legner, David Basin, David Hausheer, Samuel Hitz, Peter Müller, and Adrian Perrig. *The Complete Guide to SCION*. Springer International Publishing, 2022.

[9] CISCO. System security configuration guide for cisco NCS 540 series routers. https://www.cisco.com/c/en/us/td/docs/iosxr/ncs5xx/system-security/76x/b-system-security-cg-76x-ncs540/implementation-of-trustworthy-systems.html, 2022.

[10] DPDK Project. Data Plane Development Kit. https://dpdk.org, 2022.

[11] European Parliament and Council of the European Union. Regulation (EU) 2016/679 of the european parliament and of the council – recital 83. *Official Journal of the European Union*, 59(119), April 2016.

[12] C. Filsfils, S. Previdi, L. Ginsberg, B. Decraene, S. Litkowski, and R. Shakir. Segment Routing Architecture. RFC 8402, IETF, July 2018.

[13] Songtao Fu, Ke Xu, Qi Li, Xiaoliang Wang, Su Yao, Yangfei Guo, and Xinle Du. MASK: Practical source and path verification based on Multi-AS-Key. In *Proceedings of the IEEE/ACM International Symposium on Quality of Service (IWQoS)*, 2021.

[14] Google. Protocol Buffers. https://developers.google.com/protocol-buffers/, 2022.

[15] Shay Gueron. Intel Advanced Encryption Standard (AES) new instructions set. Technical report, Intel Corporation, 2010.

[16] Anxiao He, Xiang Li, Jiandong Fu, Haoyu Hu, Kai Bu, Chenlu Miao, and Kui Ren. Hummingbird: Dynamic path validation with hidden equal-probability sampling. *IEEE Transactions on Information Forensics and Security*, 2023.

[17] Bradley Huffaker, Marina Fomenkov, Daniel Plummer, David Moore, and K Claffy. Distance metrics in the internet, August 2002.

[18] Geoff Huston. IPv6 / IPv4 comparative statistics. https://bgp.potaroo.net/v6/v6rpt.html, 2022.

[19] ICANN. ICANN: Internet corporation for assigned names and numbers. https://www.icann.org/, 2022.

[20] Institute of Electrical and Electronics Engineers (IEEE). IEEE standard for a precision clock synchronization protocol for networked measurement and control systems. IEEE 1588-2019, 2019.

[21] Internet Assigned Numbers Authority (IANA). IANA private enterprise numbers. https://www.iana.org/assignments/enterprise-numbers/enterprise-numbers, 2022.

[22] Internet Engineering Task Force (IETF). Remote ATtestation ProcedureS (RATS). https://datatracker.ietf.org/wg/rats/about/, 2022.

[23] Grigorios Kakkavas, Despoina Gkatzioura, Vasileios Karyotis, and Symeon Papavassiliou. A review of advanced algebraic approaches enabling network tomography for future network infrastructures. *Future Internet*, 12(2):20, January 2020.

[24] Tiffany Hyun-Jin Kim, Cristina Basescu, Limin Jia, Soo Bum Lee, Yih-Chun Hu, and Adrian Perrig. Lightweight source authentication and path validation. In *Proceedings of the ACM SIGCOMM Conference*, pages 271–282, 2014.

[25] Cyrill Krähenbühl, Seyedali Tabaeiaghdaei, Christelle Gloor, Jonghoon Kwon, Adrian Perrig, David Hausheer, and Dominik Roos. Deployment and scalability of an inter-domain multi-path routing infrastructure. In *Proceedings of the International Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, 2021.

[26] Jonghoon Kwon, Juan A. García-Pardo, Markus Legner, François Wirz, Matthias Frei, David Hausheer, and Adrian Perrig. SCIONLAB: A next-generation internet testbed. In *Proceedings of the IEEE International Conference on Network Protocols (ICNP)*, 2020.

[27] Markus Legner, Tobias Klenze, Marc Wyss, Christoph Sprenger, and Adrian Perrig. EPIC: Every packet is checked in the data plane of a path-aware Internet. In *Proceedings of the USENIX Security Symposium*, 2020.

[28] Damien Magoni and Jean-Jacques Pansiot. Internet topology modeler based on map sampling. In *Proceedings of the International Symposium on Computers and Communications (ISCC)*, pages 1021–1027, February 2002.

[29] Loïc Miller and Cristel Pelsser. A taxonomy of attacks using BGP blackholing. In *Proceedings of the European Symposium on Research in Computer Security (ESORICS)*, 2019.

[30] Andrzej Mostowski. Impossibility of an algorithm for the decision problem in finite classes. *Journal of Symbolic Logic*, 15(3):229–229, 1950.

[31] Jad Naous, Michael Walfish, Antonio Nicolosi, David Mazières, Michael Miller, and Arun Seehra. Verifying and enforcing network paths with ICING. In *Proceedings of the International Conference on Emerging Networking Experiments and Technologies (CoNEXT)*, 2011.

[32] National Institute of Standards and Technology (NIST). NIST RPKI monitor. https://rpki-monitor.antd.nist.gov/, 2022.

[33] Barath Raghavan and Alex C. Snoeren. A system for authenticated policy-compliant routing. In *Proceedings of the ACM SIGCOMM Conference*, 2004.

[34] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol Label Switching Architecture. RFC 3031, IETF, January 2001.

[35] Benjamin Rothenberger, Dominik Roos, Markus Legner, and Adrian Perrig. PISKES: Pragmatic Internet-scale key-establishment system. In *Proceedings of the ACM Asia Conference on Computer and Communications Security (ASIACCS)*, 2020.

[36] Ruben Salazar, Tim Godfrey, Norm Finn, Clint Powell, Ben Rolfe, and Malik Seewald. Utility applications of time sensitive networking. White paper, Institute of Electrical and Electronics Engineers (IEEE), September 2019.

[37] SCIONLab. The SCIONLab research network. https://www.scionlab.org/, 2022.

[38] Swisscom AG. Enhancing WAN connectivity and services for Swiss organisations with the next-generation internet. https://perma.cc/8FXC-4JPA, 2022.

[39] D. Thaler. Planning for Protocol Adoption and Subsequent Transitions. RFC 8170, IETF, May 2017.

[40] Eric Voit, Henk Birkholz, Thomas Hardjono, Thomas Fossati, and Vincent Scarlata. Attestation results for secure interactions. Internet-Draft draft-ietf-rats-ar4si-03, IETF Secretariat, September 2022. https://www.ietf.org/archive/id/draft-ietf-rats-ar4si-03.txt.

[41] Eric Voit, Chennakesava Reddy Gaddam, Guy Fedorkow, Henk Birkholz, and chenmeiling. Trusted Path Routing. Internet-Draft draft-voit-rats-trustworthy-path-routing-06, Internet Engineering Task Force, September 2022. Work in Progress, https://www.ietf.org/archive/id/draft-voit-rats-trustworthy-path-routing-06.txt.

[42] David Waltermire, Brant A. Cheikes, Larry Feldman, and Greg Witte. Guidelines for the creation of interoperable software identification (SWID) tags, 2016.

[43] Cun Wang, Zhengmin Li, Xiaohong Huang, and Pei Zhang. Inferring the average AS path length of the internet. In *Proceedings of the IEEE International Conference on Network Infrastructure and Digital Content (IC-NIDC)*, pages 391–395, September 2016.

[44] M. Welzl and S. Gjessing. A Minimal Set of Transport Services for End Systems. RFC 8923, IETF, October 2020.

[45] Bo Wu, Ke Xu, Qi Li, Zhuotao Liu, Yih-Chun Hu, Martin J. Reed, Meng Shen, and Fan Yang. Enabling efficient source and path verification via probabilistic packet marking. In *Proceedings of the IEEE/ACM International Symposium on Quality of Service (IWQoS)*, 2018.

A DRKey

To encrypt and authenticate endpoint-selected policies, we rely on the fast key derivation provided by DRKey, which in turn enables us to leverage and extend the per-packet cryptographic operations introduced in EPIC.

The DRKey [24, 35] system allows ASes and endpoints to efficiently exchange symmetric keys. To make the symmetric keys issued by AS A available to its infrastructure components, such as border routers, DRKey does not store the keys at those components, but instead enables the components to recompute the keys efficiently on the fly. For this purpose, AS A keeps a single secret (symmetric) key K_A known by its relevant components (the control service and all border routers). When it receives a key request by AS B, it responds with an AS-level symmetric key $\mathrm{K}_{A\to B}$ computed with a pseudorandom function (PRF):

$$\mathrm{K}_{A\to B} := \mathrm{PRF}_{\mathrm{K}_A}(B). \tag{1}$$

Upon receiving this AS-level key, AS B can use it to further derive symmetric keys for its endpoints. It computes symmetric keys between endpoint H_B (in AS B) and AS A, and between endpoint H_B and endpoint H_A (in AS A) as follows:

$$\mathrm{K}_{A\to B:H_B} := \mathrm{PRF}_{\mathrm{K}_{A\to B}}(H_B), \tag{2}$$
$$\mathrm{K}_{A:H_A\to B:H_B} := \mathrm{PRF}_{\mathrm{K}_{A\to B}}(H_A, H_B). \tag{3}$$

Here, the comma separating the endpoint identifiers denotes concatenation, and the colons in the key names make explicit to which AS an endpoint belongs to. For traffic going through a border router of AS A and originating from endpoint H_B, the router can recompute $\mathrm{K}_{A\to B:H_B}$ based solely on its secret key K_A, plus the AS-identifier B and the endpoint source address H_B, which are part of the SCION packet header. For communication with destination endpoint H_A, the source endpoint H_B has to explicitly request $\mathrm{K}_{A:H_A\to B:H_B}$ from the control service of AS B, as it does not have the necessary AS-level key $\mathrm{K}_{A\to B}$ to derive it autonomously. Similarly, if the endpoint H_A wants to know $\mathrm{K}_{A:H_A\to B:H_B}$ for packets received from H_B, it must request this key from its local control service of AS A, as H_A is neither in possession of K_A nor $\mathrm{K}_{A\to B}$. Endpoint and AS identifiers are not secret, but to compute $\mathrm{K}_{A:H_A\to B:H_B}$, the key $\mathrm{K}_{A\to B}$ is needed, which can either be derived directly at border routers and the control service of AS A from K_A, or it needs to be fetched by AS B from the control service of AS A.

B EPIC

Leveraging DRKey's efficient key-derivation, EPIC [27] enables per-packet source authentication and path validation in SCION. *Source authentication* at border routers protects both the network and the destination endpoint, as it allows filtering unauthentic packets early and before they reach any bottleneck links. Through *path validation*, an endpoint can verify whether its traffic indeed followed the selected path on the AS-level.

To authenticate a source endpoint H_S in AS A_0 to on-path border routers, the source endpoint computes a per-packet hop validation field (HVF) for each AS A_i ($1 \le i \le n$) on the selected path, which it includes in the SCION packet header:

$$\mathrm{K}_i := \mathrm{K}_{A_i\to A_0:H_S} \tag{4}$$
$$\mathrm{V}_i := \mathrm{MAC}_{\mathrm{K}_i}(\mathrm{tsPkt}, A_0, H_S, \sigma_i)\ [0{:}\ell_{\mathrm{val}}] \tag{5}$$

Here, the function MAC_K computes a message authentication code with key K and tsPkt denotes a high-precision timestamp added to the packet header that is unique for every packet sent by source endpoint H_S.[2] The notation X[a:b] denotes the substring from byte a (incl.) to byte b (excl.) of X, and the HVF is thus defined as the first ℓ_{val} bytes of the MAC output. To verify a packet's source, a border router recomputes the HVF of its AS and compares it to the one contained in the packet header.

To provide path validation to the source endpoint, the border routers replace the HVFs in the packet with the next ℓ_{val} bytes of the MAC output, i.e., $[\ell_{\mathrm{val}}{:}2\ell_{\mathrm{val}}]$. This serves as proof that the packet indeed traversed the ASes on the selected path. To communicate this information to the source endpoint, the destination endpoint H_D returns a packet containing the updated HVFs and tsPkt of the original packet, authenticated with its symmetric key $\mathrm{K}_{A_\ell:H_D\to A_0:H_S}$.

[2] The last input to the MAC, σ_i, is an authenticator included by EPIC to achieve a property called *path authorization*, which protects the routing decision of ASes from malicious endpoints. The computation of σ_i is irrelevant for our work.

Through the deployment of a duplicate-suppression system, EPIC furthermore allows ASes to filter replayed packets. Also, an EPIC-enabled source endpoint authenticates the whole packet including the payload for the destination endpoint, and therefore also includes a corresponding destination validation field in the packet header.

C Securing Path Policy Indices

To provide authenticity and confidentiality of the policies selected by endpoints, we encrypt the corresponding indices and add them to the input of the EPIC per-packet authenticator computation from Equation (5):

$$\psi_i^E = \psi_i \oplus (\mathrm{AES}_{\mathrm{K}_i}(\mathrm{tsPkt})\ [0{:}\ell_{\mathrm{pol}}]) \tag{6}$$

$$\mathrm{V}_i = \mathrm{MAC}_{\mathrm{K}_i}(\mathrm{tsPkt}, A_0, H_S, \sigma_i, \psi_i^E)\ [0{:}\ell_{\mathrm{val}}] \tag{7}$$

Here, ψ_i denotes the plaintext and ψ_i^E the encrypted policy index for on-path AS i, respectively. The length of the policy index in bytes is given by ℓ_{pol}. Again, the notation X[a:b] denotes the substring from byte a (incl.) to byte b (excl.) of X, and the comma-separated inputs to the MAC function are concatenated. This approach follows the Encrypt-then-MAC idea, where we use the CTR mode for encryption and where the timestamp tsPkt serves as nonce. The counter is always zero, i.e., omitted, because the length of the input (tsPkt) is shorter than the AES block size. Alternatively, the computation of ψ_i^E in Equation (6) can be understood as one-time pad encryption using the key $\mathrm{AES}_{\mathrm{K}_i}(\mathrm{tsPkt})$. The source endpoint adds tsPkt, ψ_i^E and V_i for every on-path AS i to its data packet.

The reason we do not directly apply the AES block cipher to the packet timestamp and the policy index (i.e., without using the bitwise XOR operation) is because this would result in an encrypted policy index with a size of 16 B (AES block size). With our approach, the encrypted policy index consists of only ℓ_{pol} bytes, which allows for shorter packet headers.

Upon reception of a data packet, the ingress border router of the i-th on-path AS checks that the timestamp is current, derives K_i (Equations 2 and 4), and re-computes V_i (Equation 7) and compares it to the V_i contained in the packet. If they do not match, the packet is dropped. Otherwise, the router decrypts ψ_i^{Enc} using $\oplus$ to obtain the policy index ψ_i. It then replaces the HVF with the the ℓ_{val} next bytes of the MAC output and, based on a local lookup table, forwards the packet such that it traverses the intra-AS network in a policy-compliant manner. In case there is no entry in the table for this policy index, the router sends back a control message to the source endpoint and drops the packet. The destination endpoint is not modified, it checks the destination validation field and returns an authenticated confirmation containing the updated HVFs and tsPkt of the original packet, such that the source can verify the AS-level path the packet traversed.

Through those modifications, we extend EPIC to not only achieve source authentication and path validation, but also secrecy and authenticity of the policy indices. Because policy indices are encrypted on a per-packet basis, an attacker cannot infer whether two encrypted indices describe the same plaintext policy index by only looking at the packet headers. Hence, such an attacker can not even deduce whether the source endpoint has chosen any policies at all, as a policy index of zero is indistinguishable from any other policy index after encryption.

D Policy Specification Details

In this section, we provide additional details on the syntax of the policy language and our policy encoding.

D.1 Policy Syntax

The syntax of our policy language is based on the following general syntax for FOL with equality. In addition to the sorts, function symbols, and predicates defined in Section 5.1, we have

Sorts: A set of sorts $\{I_1, I_2, ..., I_n\}$

Vars: A set of sorted variables $\{v_1, v_2, ...\}$, where each variable v_k is of some sort I, written v_k^I.

Function Symbols: A set of function symbols. The input parameters and the output of a function (i.e., its signature) are of specific sorts: $f : I_{i_1} \times I_{i_2} \times ... \times I_{i_n} \to I_r$

Constants: A set of sorted constants (i.e., function symbols with arity 0)

Predicates: A set of predicates. The input parameters of an n-ary predicate are of the sorts $I_{i_1}, I_{i_2}, ..., I_{i_n}$, where $n \geq 1$

Terms are defined inductively:

Vars $\subseteq$ Terms

Constants $\subseteq$ Terms

if $t_1^{I_{t_1}}, t_2^{I_{t_2}}, ..., t_k^{I_{t_k}} \in$ Terms and f is a function with signature $I_{t_1} \times I_{t_2} \times ... \times I_{t_k} \to I_r$, then $f(t_1, t_2, ..., t_k)^{I_r} \in$ Terms

Formulas are defined inductively:

if $t_1^I, t_2^I \in$ Terms for some sort I, then $t_1 = t_2 \in$ Formulas

if $t_1^{I_1}, t_2^{I_2}, ..., t_k^{I_k} \in$ Terms and P is a predicate with signature $I_1 \times I_2 \times ... \times I_k$, then $P(t_1, t_2, ..., t_k) \in$ Formulas

if $\phi \in$ Formulas and $\theta \in$ Formulas, then

$$\neg\phi,\ \ \phi\wedge\theta,\ \ \phi\vee\theta,\ \ \phi\rightarrow\theta \quad \in \text{Formulas}$$

if $\phi \in$ Formulas and $x^I \in$ Vars is a variable of sort I, then

$$\forall x^I : \phi,\ \ \exists x^I : \phi \quad \in \text{Formulas}$$

if $t_1^I, t_2^I \in$ Terms are terms of sort I (note that in our policy language, inequality is only defined for sort V), then

$$t_1 < t_2,\ \ t_1 \leq t_2,\ \ t_1 > t_2,\ \ t_1 \geq t_2 \quad \in \text{Formulas}$$

D.2 Semantics and Interpretation

An interpretation for our sorted FOL with equality is defined by a set of carrier sets interpreting each sort, a set of relations interpreting each predicate, and a set of functions (including constants with arity 0) interpreting each function symbol. We first define interpretation of the sorts in our language:

Manufacturer (Sort M): Entity creating and distributing routers. Let $\mathcal{M}$ be the set of all manufacturers.

Software Component (Sort C): A software product, which is identified by a unique tag, typically assigned by the owner of the software product, and may support versioning. Let $\mathcal{C}$ be the set of all software components.

Tag (Sort T): A software component is clearly identified by a globally unique tag. Let $\mathcal{T}$ be the set of all possible tags. The function `tag`: $\mathcal{C} \rightarrow \mathcal{T}$ returns the tag of a software component.

Tag Issuer (Sort I): The tag of a software component is assigned by a tag issuer, which is identified by a URI. Let $\mathfrak{I}$ be the set of all possible issuers. The function `issuer`: $\mathcal{T} \rightarrow \mathfrak{I}$ returns the issuer of a tag.

Name (Sort N): The name of a software component, which allows the identification of a software component with different versions. Let $\mathcal{N}$ be the set of all possible names assigned to a software component. The function `name`: $\mathcal{C} \rightarrow \mathcal{N}$ returns the name of a software component.

Version (Sort V): A software component may be identified by a specific version. Let $\mathcal{V}$ be a totally ordered set of all possible version numbers for different version schemes defined in the NIST standard for Software Identification Tags [42, 5.1.2]. The function `version`: $\mathcal{C} \rightarrow \mathcal{V}$ returns the version of a software component or the default version ("1.0.0").

Router (Sort R): A device produced by a specific manufacturer and running a clearly defined software stack. Let $\mathcal{R}$ be the set of all possible routers.

Path (Sort P): A sequence of routers but we simplify the representation of a path as a set since order and repetition are not relevant for path policies. The elements of a path are thus defined by the relation `onPath`: $\mathcal{P} \times \mathcal{R}$. Let $\mathcal{P}$ be the set of all possible paths.

Router Setup: A router setup consists of its manufacturer and software stack, i.e., the set of software components providing the router functionality. It is thus defined by the function and relation, `manufacturer`: $\mathcal{R} \rightarrow \mathcal{M}$ and `software`: $\mathcal{R} \times \mathcal{C}$.

Based on the above, we define an interpretation as follows:

Sorts: The sorts and their respective carrier sets are defined in Table 1. The concrete values used to encode these router attributes are described in Appendix D.3.

Predicates: Predicate symbols are interpreted by relations. The predicate for the total order on $\mathcal{V}$ overloads the comparison operator (relation) $\leq$ over $V \times V$. This total order is defined by the version scheme (e.g., SemVar where versions use a three-part version number such that $1.2.3 \leq 1.3.2$). The remaining operators $<, \geq$, and $>$ can be defined in terms of $\leq$[3]. Finally, there are the relations `onPath` and `software`.

Functions: The function symbols `manufacturer`, `tag`, `issuer`, `name`, and `version` are interpreted by their respective functions. Constants, i.e., function symbols with arity 0, are interpreted by the carrier set of their respective sort as described in Table 1.

D.3 Policy Encoding

Our policies are based on two building blocks: manufacturers and software components with their respective attributes (tags, issuers, names, and versions). To communicate policies among different entities, we need a common set of values for each building block. We encode the manufacturer using the private enterprise numbers specified by IANA [21]. These numbers are globally consistent unique identifiers for different manufacturers. The software components including their related attributes are encoded using `SoftwareIdentity` elements from the NIST standard for Software Identification Tags [42].

[3] $p < q := p \leq q \wedge p \neq q$, $p \geq q := q \leq p$, and $p > q := q < p$