Formal Verification of Secure Forwarding Protocols

Tobias Klenze
ETH Zurich
tobias.klenze@inf.ethz.ch

Christoph Sprenger
ETH Zurich
sprenger@inf.ethz.ch

David Basin
ETH Zurich
basin@inf.ethz.ch

***Abstract*—Today's Internet is built on decades-old networking protocols that lack scalability, reliability, and security. In response, the networking community has developed *path-aware* Internet architectures that solve these issues while simultaneously empowering end hosts. In these architectures, autonomous systems construct authenticated forwarding paths based on their routing policies. Each end host then selects one of these authorized paths and includes it in the packet header, thus allowing routers to efficiently determine how to forward the packet. A central security property of these architectures is *path authorization*, requiring that packets can only travel along authorized paths. This property protects the routing policies of autonomous systems from malicious senders.**

The fundamental role of packet forwarding in the Internet and the complexity of the authentication mechanisms employed call for a formal analysis. In this vein, we develop in Isabelle/HOL a parameterized verification framework for path-aware data plane protocols. We first formulate an abstract model without an attacker for which we prove path authorization. We then refine this model by introducing an attacker and by protecting authorized paths using (generic) cryptographic validation fields. This model is parameterized by the protocol's authentication mechanism and assumes five simple verification conditions that are sufficient to prove the refinement of the abstract model. We validate our framework by instantiating it with several concrete protocols from the literature and proving that they each satisfy the verification conditions and hence path authorization. No invariants must be proven for the instantiation. Our framework thus supports low-effort security proofs for data plane protocols. The results hold for arbitrary network topologies and sets of authorized paths, a guarantee that state-of-the-art automated security protocol verifiers cannot currently provide.

I. Introduction

The Internet is a global network of ca. 70,000 independently managed networks, called *autonomous systems* (ASes), which are run by entities such as Internet service providers (ISPs), content providers, or public institutions. Routing is based on the aging Border Gateway Protocol (BGP), a protocol that scales poorly and has no built-in security. In response to these well-known problems, the networking community has been working to augment BGP with security mechanisms [1]–[3], but the proposed solutions have proven to be insufficient, inefficient or to introduce new problems [4]–[6].

In parallel to the proposed BGP enhancements, which trade off performance for security, many researchers have acknowledged the need for a clean-slate approach. The networking community has invested substantial effort into developing novel security protocols with the objective of building a new Internet architecture that is both more efficient and more secure. We focus here on *path-aware* architectures [7]–[13], which, in contrast to the current Internet, provide end hosts with some control over the paths along which they send their packets.

Networking architectures generally consist of a *control plane* and a *data plane* (also called *forwarding plane*). On the control plane, routers exchange topology information and establish paths. In the path-aware Internet architectures that we focus on, the control plane constructs forwarding paths as sequences of cryptographically authenticated forwarding fields, one for each AS on the path. The data plane forwards packets along these paths. The source selects a forwarding path for each packet and includes it in the packet's header (*packet-carried forwarding state*). Routers forward the packets according to their AS' forwarding information, which they extract from the packet's path and validate by checking the associated cryptographic authenticator. Since each packet contains its own forwarding state, routers do not require routing tables (unlike BGP routers).

ASes have routing policies that rule out impractical or uneconomical paths. Path-aware architectures allow end hosts to select paths, but architectures must ensure that the policies of ASes are followed. To protect ASes from malicious sources, the control plane should only construct and authenticate paths consistent with each AS' policy, and the data plane should only forward packets along these paths. The latter property is called *path authorization* and is the central security property of path-aware data planes. The cryptographic authenticators embedded in forwarding paths ensure that malicious end hosts cannot tamper with the paths produced by the control plane to forward packets along unauthorized paths.

The complexity of data plane protocols and their central role in future Internet architectures calls for their formal verification to obtain sufficient assurance about their correctness and security. There are several payoffs from this effort. First, it enables the early detection of protocol flaws and vulnerabilities, avoiding critical exploits and high costs for corrections after deployment has begun. This is especially true for the data plane since it is implemented in large numbers of high-performance software or hardware routers, which are difficult to update after their deployment. Second, a formal proof increases confidence in the architecture's security, thereby fostering its adoption.

Data plane protocols exhibit characteristics that make the verification of path authorization particularly challenging. First, we want to verify this security property over arbitrary network topologies and authorized paths therein, as determined by the control plane. Second, the formalism must be expressive enough to describe (i) path authorization, which is a non-local property that involves all ASes on a path, and (ii) assumptions on

a control plane adversary's capabilities to shorten, modify, and extend certain authorized paths. Third, the number of participants and the message sizes in a protocol run depend on the (unbounded) length of the path embedded in a given packet. We anticipate that state-of-the-art automated security protocol verifiers such as Tamarin [14] and ProVerif [15] could only be used to verify path authorization in relatively small, fixed topologies and fixed sets of authorized paths.

For this reason, we employ a higher-order logic theorem prover, Isabelle/HOL [16], which allows us to model and verify data plane protocols in their full generality. As research in novel path-aware Internet architectures has led to several interesting candidate (families of) data plane protocols, we would like to verify these without the need to restart the specification and verification effort from scratch for each protocol or variant. Specifications and proofs should thus share as much structure as possible. To achieve this, we propose a parametrized framework in Isabelle/HOL for the verification of families of data plane protocols (Fig. 1).

We first develop a simple abstract model of a packet forwarding protocol without an attacker, for which we formulate and easily prove path authorization. We then refine this model into a more concrete one, where we introduce a Dolev–Yao adversary and (generic) cryptographic authenticators, called *hop validation fields* (*hvf*), that protect each AS-level hop along a forwarding path. A key insight is that the only substantial difference between path authorization mechanisms is how the *hvf* is computed. This allows us to define a single skeleton protocol model, which we can instantiate to a wide range of actual protocols. We achieve this by parametrizing the concrete model by (i) a cryptographic hop validation check that must be performed by each AS locally to determine the authorization of the forwarding path, (ii) a function that extracts an entire forwarding path from a hop validation field, and (iii) a set of (cryptographic) terms added to the attacker's knowledge.

We identify five simple *verification conditions* on these parameters that suffice to prove that the concrete model refines the abstract one and therefore inherits the path authorization property. These conditions require the *hvf* to be unforgeable and to contain the path. Our development is also parametrized by an arbitrary network topology and a set of authorized paths constructed by the control plane. Our security proofs hold for all network topologies and control planes that satisfy some realistic assumptions.

To define a concrete data plane protocol in our framework, we instantiate the parameters (i)-(iii) and discharge the five verification conditions. We do so for the data plane of SCION [13], several members of the EPIC protocol family [17], and ICING [18], as well as for variants of these protocols. The instantiations and the associated proofs of the verification conditions are substantially shorter, simpler, and more manageable than redoing a full specification and security proof for each protocol. In particular, discharging the conditions does not involve reasoning about state transitions (unlike, e.g., proving an invariant).

Since path-aware Internet architectures and path authorization are relatively new concepts, there is little existing verification work. The most closely related works are the verification of a weaker AS-local form of path authorization [19] and of different security properties [20] for such architectures. Both of these works mechanize their proofs in Coq using a non-foundational approach, i.e., relying on an axiomatization or external tools. We further discuss related work in §X.

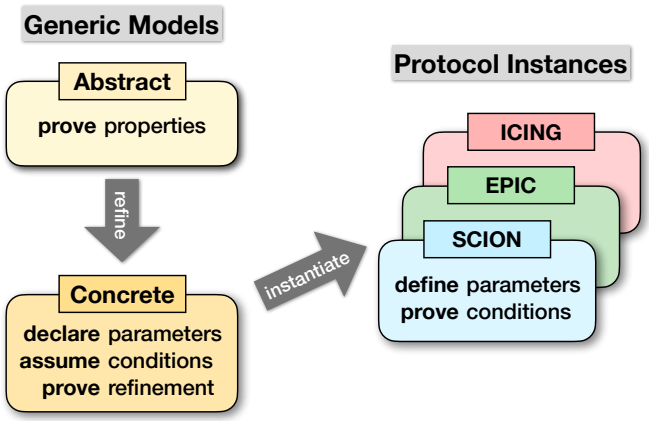

Fig. 1: Overview of our models. Refinement and instantiation preserve properties.

Contributions: Our main contributions are as follows. (i) We develop a generic framework for verifying security properties for a general class of data plane protocols for arbitrary network topologies. This framework has three protocol parameters that are required to satisfy five simple verification conditions. (ii) The five conditions provide insight into the common structure underlying data plane protocols of path-aware Internet architectures. (iii) We instantiate our framework with eight different variants of realistic data plane protocols proposed in the literature and prove that they satisfy path authorization by establishing the parametrized model's verification conditions. (iv) All of our definitions and results are formalized in Isabelle/HOL following a foundational approach, which only relies on the axioms of higher-order logic and thus provides strong soundness guarantees. All results are available online.[1]

II. Problem Domain and Overview

In this section, we provide background on the secure forwarding problem and give an overview of our framework, in particular the protocols and security properties that we verify.

A. Motivation for future Internet architectures

Networking in today's Internet is plagued by numerous performance and security problems. Forwarding uses longest-prefix matching on large routing tables, which scales poorly and requires expensive hardware support. Changes to the network topology trigger routing updates that can lead to outages lasting tens of minutes [21], and in some topologies, BGP does not converge to a stable state at all [22].

[1] https://doi.org/10.5281/zenodo.4515953

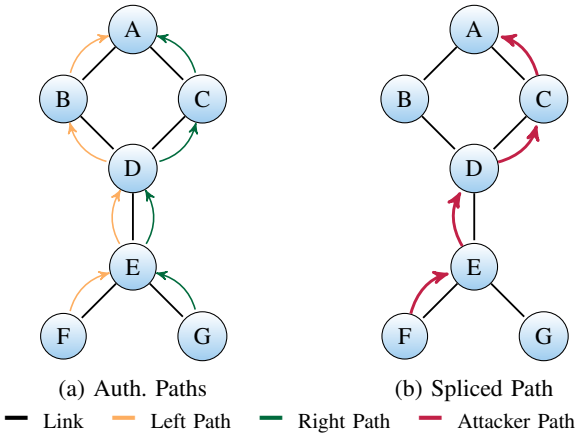

Fig. 2: If *path authorization* holds, a malicious sender at node F cannot splice the two authorized paths in (a) to create the unauthorized forwarding path in (b).

The absence of security in BGP is a major concern. The lack of secure routing allows any of the ca. 70,000 Internet ASes [23] to launch prefix hijacking attacks [24]. Various protocols have been proposed that add security mechanisms to the existing BGP infrastructure, such as BGPSec [3], S-BGP [1], soBGP [25], psBGP [26], and PGBGP [27]. Unfortunately, these additions are insufficient to solve the Internet's problems [4]–[6], or they introduce new problems such as high overhead [28], [29] and kill switches [6]. In short, they trade off security with performance and they fail to address the reliability problems of BGP's convergence-based approach.

Our work instead applies to data planes of a wide range of future Internet architectures that follow a clean-slate path-aware approach [7]–[13]. We first discuss these data planes generically, and present SCION as an example in §II-E, where we also describe SCION's control plane.

B. Data planes of future Internet architectures

Each AS in the Internet administers its internal network including its routing and forwarding mechanisms. We abstract from internal forwarding and consider the networking *between* ASes, which requires entities to agree on a common protocol. Since we view the Internet as a network of ASes, we also refer to ASes as *nodes*. See Fig. 2 for an example of a (tiny) Internet topology. The internal structure of an AS is shown in Fig. 3. Nodes are interconnected at *border routers*, which sit at the edge of each node's network and perform both inter-AS and intra-AS forwarding.

The path-aware Internet architectures that we examine provide end hosts with *path control*, which is the ability to choose from a set of authorized forwarding paths for each destination. End hosts select their desired forwarding path at the granularity of inter-AS links, and embed the path alongside authenticators in each data packet. This *packet-carried forwarding state* removes the need for border routers to keep state for inter-AS forwarding. Path control also empowers end hosts to make

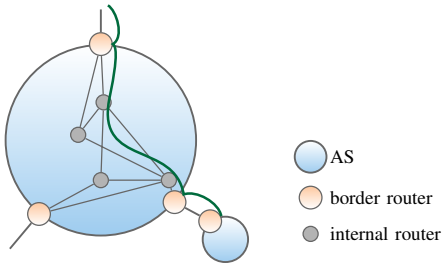

Fig. 3: Internals of AS E of Fig. 2 with one path shown. The internal path between border routers is decided by the AS.

path choices that are suitable for their applications' needs. For instance, Voice-over-IP requires little bandwidth but low latency, whereas data synchronization requires high bandwidth, but latency is less important. These applications can thus use different paths. Moreover, *multipath routing* allows multiple paths between the same source-destination pair to be used simultaneously, even by the same application.

The end host's power of choosing paths is balanced against the interest of ASes. Paths are discovered and authorized on the control plane (cf. §II-E), which must ensure that paths are only authorized if they satisfy the routing policies of ASes.

The forwarding paths embedded into packets consist of a *hop field* for each AS on the path. It includes the AS identifier *id*, as well as the interfaces *prev*, on which the packet is received, and *next*, on which the packet is to be sent out. The AS uses its own networking infrastructure to forward the packet between the border routers adjacent to these interfaces. The first and last hop fields on a path only contain a single interface, as there is no predecessor and successor AS, respectively. The path is fixed by the sending end host and remains static. A moving pointer indicates the current hop field. Each hop field is authenticated by its AS when it is first constructed on the control plane and a *hop validation field* (*hvf*), typically a *message authentication code* (MAC) proving its authorization on the data plane, is embedded in each hop field. Upon receiving a data packet, a border router checks the *hvf* of its hop field. If it is valid, then the path was authorized on the control plane, and the border router forwards the packet.

Depending on the context we use the term hop field and write hf_i to refer to either the unauthenticated (without *hvf*) or the authenticated (with *hvf*) forwarding data of the AS i. In our formal definitions we call the former *abstract* hop field (§IV, Eq. (3)), and the latter *concrete* hop field (§V, Eq. (8)).

There are two variants of path authorization mechanisms that differ in how paths are authorized on the control plane: in *undirected* protocols, each AS authorizes the path in its entirety. In *directed* protocols, each AS only authorizes the partial path consisting of its own hop and all subsequent hops in forwarding direction. For instance, in Fig. 2a, AS D could decide which of the partial paths D-B-A and D-C-A to allow, but once authorization is granted, extensions authorized by E and E's children are also implicitly authorized by D. As we will see in §VI-A and §VIII-C, the two variants require

different control plane assumptions.

In the protocols we study, path authorization still holds even if the malicious sender learns the keys of compromised on-path ASes. However, we must make assumptions to exclude trivial violations of path authorization.

C. Security properties that we verify

We verify two data plane security properties (formally defined in §IV-D): *path authorization* and *detectability*. They protect ASes against malicious senders and compromised ASes.

1) Path authorization: This is the data plane property that all data packets traverse the Internet only along authorized paths. It protects ASes from malicious senders who could try to forge paths that are advantageous to themselves (e.g., by using costly paths that they have not paid for), detrimental to ASes (e.g., uneconomical *valley* paths [30]), or disrupt forwarding entirely (e.g., through loops).

Path authorization is not just a local property; in particular, it is not enough for each hop to authorize its local routing information (*prev* and *next* interfaces), since that would still allow for attacks such as forwarding loops. If a strictly hierarchical structure with defined provider–customer and peering relationships is assumed, forwarding loops can be prevented with only local checks [31]. However, path policies in general cannot be protected with only local checks, as our example in Fig. 2a illustrates. In this example, two paths leading to the destination node A are authorized: the left path F–E–D–B–A, and the right path G–E–D–C–A. Node F is only authorized to use the left path and is forbidden to send packets to A via C. Path authorization implies that an attacker at F cannot craft a packet that traverses the path given in Fig. 2b. Each AS only checking its local forwarding information cannot guarantee this.

2) Detectability: Since the entire forwarding path is contained in each packet, a malicious source cannot hide her presence on the path. *Detectability* states that the actual path that a packet traverses is contained in the path embedded in the packet's header. This property does not prevent source spoofing, but rather ensures that *if* a source is spoofed, then the attacker must be in one of the nodes on the packet's forwarding path, thus ensuring basic accountability for data packets.

D. Security properties that we do not verify

Source and packet authentication allow border routers or the destination to authenticate the sender and packet. *Path validation* allows the destination to verify that the path contained in the packet was actually traversed. We do not verify these data plane properties, for reasons given in §IX-E.

Intra-AS forwarding is out of scope, since each AS exercises control over its own network, and global coordination is not required for intra-AS security. We also do not specify or verify the control plane, as its properties are independent from those of the data plane. For instance, path authorization is independent of the property that a path authorized by the control plane is in accordance with the routing policies of all on-path ASes.

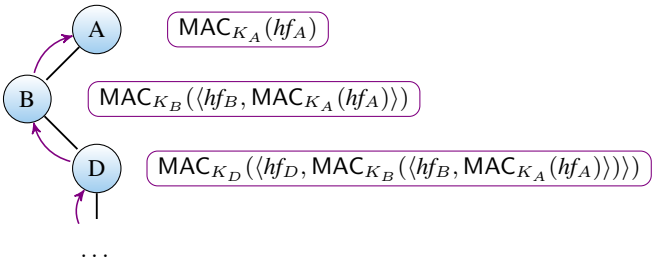

Fig. 4: SCION's hop validation fields that contain nested MACs. The fields hf_i contain the local forwarding data of AS i.

E. SCION control and data plane

We now show a concrete protocol that implements the control and data plane and achieves path authorization. We use a simplified version of SCION as an example.

Authorized paths are established on SCION's control plane using path-discovery *beacons*. Beacons are initialized by a subset of nodes and constructed in the opposite direction of forwarding. Each AS decides which of the beacons it has created or received should be propagated to a given neighbor. Since the AS is implicitly authorizing the further dissemination by the neighbor, SCION is a directed protocol. For every beacon that is propagated, the AS adds a hop field, containing its own forwarding information (in accordance with its routing policy) and cryptographic authenticators. Beacons contain two types of authenticators: signatures, which authenticate the beacons themselves on the control plane, and MACs, which are used to achieve path authorization on the data plane. The signatures are stripped off the beacons before they are embedded into a data plane packet. On the data plane, which transports vastly more packets than the control plane, asymmetric cryptography is too slow to be employed for each packet. Consequently, SCION's data plane relies entirely on symmetric cryptography.

Each AS A has a key K_A that is shared by its border routers. The hop validation field hvf_A is a MAC over A's local forwarding data hf_A, and the MAC of the next hop B:

$$hvf_A = \mathsf{MAC}_{K_A}(\langle hf_A, hvf_B\rangle)\,. \tag{1}$$

Crucially, the MAC is created not only over the local routing information, but also over the next MAC. As Fig. 4 illustrates, this nests the MACs and protects the entire subsequent path. During forwarding, each border router checks the validity of its own MAC embedded in the packet header. The validation of the MAC is substantially faster, and scales better, than looking up authorized paths in a table on each router [13].

F. Verified data plane protocols

Our verification framework applies to both directed and undirected protocols. We analyze two data plane protocols besides SCION and prove path authorization and detectability:

- EPIC [17], a family of directed data plane protocols that provide three levels of security guarantees. We verify levels 1 and 2. EPIC levels 2 and 3 add authentication and path validation mechanisms, which we do not verify.

$\mathbb{N}$, $\mathbb{B}$, $\mathscr{N}$, $\mathscr{I}$	naturals, bools, nodes, interfaces
$A \times B$, A^*	cartesian product, finite sequences
$A_\perp$	option type (disjoint sum of A and $\{\perp\}$)
$A \to B$, $A \rightharpoonup B$	total and partial functions
$dom(f)$, $ran(f)$	function domain, range
$\mathcal{P}(A)$, $(\!\mid x \in A, y \in B \mid\!)$	powerset, set of records
$(\!\mid x = a, y = b \mid\!)$, $x(r)$	concrete record, record field x
$\langle\rangle$, $x \# xs$, $\langle a, b, c\rangle$	empty, cons, concrete sequence
$xs \le ys$, $x \in xs$	sequence prefix, sequence membership
$hd(xs)$, $tl(xs)$	list head and tail if cons, else $\perp$
$xs \cdot ys$, $rev(xs)$	concatenation, sequence reversal

TABLE I: Summary of notation and definitions.

- ICING [18], the undirected data plane protocol in the NEBULA Internet architecture [12]. It also provides path validation, which we do not verify.

We will formalize these protocols in §VII as instances of our parametrized framework.

III. Preliminaries

In this section, we provide background on event systems, refinement, and model parametrization. We introduce relevant notation in Table I. Despite our use of Isabelle/HOL, we largely use standard mathematical notation and deliberately blur the distinction between types and sets.

A. Event systems, invariants, and refinement

Event systems are labeled transition systems, where transitions are labeled with *events*. Formally, an event system is of the form $\mathcal{E} = (\mathbb{S}, s^0, E, \{\xrightarrow{e}\}_{e \in E})$, where $\mathbb{S}$ is a set of states, $s^0 \in \mathbb{S}$ is the initial state, E is a set of events, and $\xrightarrow{e} \subseteq \mathbb{S} \times \mathbb{S}$ is the transition relation corresponding to event e. As usual, we write $s \xrightarrow{e} s'$ for $(s, s') \in \xrightarrow{e}$. The set of states reachable from a state s, written $\mathit{reach}(\mathcal{E}, s)$, is inductively defined by $s \in \mathit{reach}(\mathcal{E}, s)$, and $s' \in \mathit{reach}(\mathcal{E}, s)$ and $s' \xrightarrow{e} s''$ implies $s'' \in \mathit{reach}(\mathcal{E}, s)$. A state property P is a subset of $\mathbb{S}$ (or, equivalently, a predicate on $\mathbb{S}$). A state property P is an *invariant* of $\mathcal{E}$, written $\mathcal{E} \models P$, if $\mathit{reach}(\mathcal{E}, s^0) \subseteq P$.

Given an abstract event system $\mathcal{E}_\mathrm{a} = (\mathbb{S}_\mathrm{a}, s^0_\mathrm{a}, E_\mathrm{a}, \{\xrightarrow{e}_\mathrm{a}\}_{e \in E_\mathrm{a}})$ and a concrete event system $\mathcal{E}_\mathrm{c} = (\mathbb{S}_\mathrm{c}, s^0_\mathrm{c}, E_\mathrm{c}, \{\xrightarrow{e}_\mathrm{c}\}_{e \in E_\mathrm{c}})$ we say that $\mathcal{E}_\mathrm{c}$ *refines* $\mathcal{E}_\mathrm{a}$ if there are refinement mappings $\pi_0 : \mathbb{S}_\mathrm{c} \to \mathbb{S}_\mathrm{a}$ on states and $\pi_1 : E_\mathrm{c} \to E_\mathrm{a}$ on events such that $\pi_0(s^0_\mathrm{c}) = s^0_\mathrm{a}$ and for all $s_\mathrm{c}, s'_\mathrm{c} \in \mathbb{S}_\mathrm{c}$ and $e_\mathrm{c} \in E$ such that $s_\mathrm{c} \xrightarrow{e_\mathrm{c}}_\mathrm{c} s'_\mathrm{c}$ we have $\pi_0(s_\mathrm{c}) \xrightarrow{\pi_1(e_\mathrm{c})}_\mathrm{a} \pi_0(s'_\mathrm{c})$. This is functional forward simulation [32]. Refinement preserves invariants from the abstract to the concrete model, i.e., $\mathcal{E}_\mathrm{a} \models P$ implies that $\mathcal{E}_\mathrm{c} \models \pi_0^{-1}(P)$, where $\pi_0^{-1}(P) = \{s \in \mathbb{S}_\mathrm{c} \mid \pi_0(s) \in P\}$.

In our models, we often use parameterized events and states structured as records. We use the notation

$$e(\bar{x}) : \;\; g(\bar{x}, \bar{v}) \;\triangleright\; \bar{w} := \bar{u}(\bar{x}, \bar{v})$$

to specify such events, where $\bar{x}$ are the event's parameters (the bar representing a vector), $\bar{v}$ are the state record's fields, $g(\bar{x}, \bar{v})$ is the *guard* predicate defining the executability of the event, $\bar{w} \subseteq \bar{v}$ are the updated fields, and $\bar{u}$ are update functions (one for each variable in $\bar{w}$). This notation denotes the transition relation defined by $s \xrightarrow{e(\bar{x})} s'$ iff $g(\bar{x}, s(\bar{v}))$ holds, $s'(\bar{w}) = \bar{u}(\bar{x}, s(\bar{v}))$ and, for the state fields $\bar{z} = \bar{v} - \bar{w}$ that are not updated, $s'(\bar{z}) = s(\bar{z})$. We often use updates of parameterized channel fields holding sets of messages. For example, if m is an integer, the event $\mathbf{send}(A, B, m) : \;\; m > 0 \triangleright ch(A, B) \mathrel{+}= m$ adds packet m to the channel ch between A and B if m is positive, i.e., the intended update is $ch(A, B) := ch(A, B) \cup \{m\}$. Otherwise the state remains unmodified, in particular, $ch(A', B') := ch(A', B')$ for all $(A', B') \neq (A, B)$.

B. Parametrization

The generality of our models rests on their parametrization. A parametrized model may include assumptions on its parameters. An instance must define the parameters and prove the assumptions. For easy identification, we will highlight parameters in gray when they are first introduced. Parametrization is independent of refinement. For instance, a model can be parametrized and concrete at the same time (as is the case in our framework). In our Isabelle/HOL formalization we implement parametrization using *locales* [33].

IV. Abstract Model

We define an event system that models the abstract data plane of a path-aware network architecture. This model includes neither cryptography nor an attacker. We prove that it satisfies path authorization and detectability.

To distinguish definitions of this abstract model from those of the concrete model that refines it (§V), we use subscripts 'a' and 'c', respectively.

A. Environment parameters

We model the Internet as a multigraph whose nodes represent ASes and edges represent the network links between them. More precisely, a *network topology* is a triple $(\mathscr{N}, \mathscr{I}, \mathit{target})$, where $\mathscr{N}$ is a set of nodes, $\mathscr{I}$ is a set of interfaces, and

$$\mathit{target} \in \mathscr{N} \times \mathscr{I} \rightharpoonup \mathscr{N} \times \mathscr{I} \tag{2}$$

is a partial bijective function that models links between ASes and is an environment parameter to our model.[2] We say that an interface i is *valid* for a node A, if $(A, i) \in \mathit{dom}(\mathit{target})$, whereby $\mathit{target}(A, i) = (B, j)$ denotes the node B and interface j at the other end of the link. Our definition thus allows for multiple links between a given pair of nodes, with possibly different forwarding policies.

We often reason directly about paths in the network, rather than the network topology. These paths are defined in terms of both nodes and their interfaces. We define a *path* to be a finite sequence of abstract *hop fields* from the set

$$\mathrm{HF}_\mathrm{a} = (\!\mid \mathit{id} \in \mathscr{N},\; \mathit{prev} \in \mathscr{I}_\perp,\; \mathit{next} \in \mathscr{I}_\perp \mid\!). \tag{3}$$

Each hop field contains the local routing information of a node, i.e., its node identifier and the interfaces that identify the links

[2]The bijectivity of *target* models unicast communication (in contrast to, e.g., broadcast), however this is not required as an assumption for our proofs.

dispatch-int$_{\text{a}}(A, m)$: $\mathit{fut}(m) \in \mathit{auth}_{\text{a}}^{\rightleftarrows} \wedge \mathit{hist}(m) = \langle\rangle$ ▷ $\mathit{int}(A) \mathrel{+}= m$	**dispatch-int**$_{\text{c}}(A, m)$: $m \in \mathit{DY}(\mathit{ik}) \wedge \mathit{hist}(m) = \langle\rangle$ ▷ $\mathit{int}(A) \mathrel{+}= m.$
dispatch-ext$_{\text{a}}(A, i, m)$: $\mathit{fut}(m) \in \mathit{auth}_{\text{a}}^{\rightleftarrows} \wedge \mathit{hist}(m) = \langle\rangle \wedge$ $(A, i) \in \mathit{dom}(\mathit{target})$ ▷ $\mathit{ext}^{\text{send}}(A, i) \mathrel{+}= m.$	**dispatch-ext**$_{\text{c}}(A, i, m)$: $m \in \mathit{DY}(\mathit{ik}) \wedge \mathit{hist}(m) = \langle\rangle$ $(A, i) \in \mathit{dom}(\mathit{target})$ ▷ $\mathit{ext}^{\text{send}}(A, i) \mathrel{+}= m.$
send$_{\text{a}}(A, m, \mathit{hf}, i)$: $\mathit{hf} = \mathit{hd}(\mathit{fut}(m)) \wedge \mathit{hf} \neq \bot \wedge A = \mathit{id}(\mathit{hf}) \wedge i = \mathit{next}(\mathit{hf}) \wedge$ $m \in \mathit{int}(A) \wedge (A, i) \in \mathit{dom}(\mathit{target})$ ▷ $\mathit{ext}^{\text{send}}(A, i) \mathrel{+}= \mathit{fwd}_{\text{a}}(m).$	**send**$_{\text{c}}(A, m, \mathit{hf}, i)$: $\mathit{hf} = \mathit{hd}(\mathit{fut}(m)) \wedge \mathit{hf} \neq \bot \wedge A = \mathit{id}(\mathit{hf}) \wedge i = \mathit{next}(\mathit{hf}) \wedge$ $m \in \mathit{int}(A) \wedge (A, i) \in \mathit{dom}(\mathit{target}) \wedge$ $\psi(\mathit{hf}, \mathit{hd}(\mathit{tl}(\mathit{fut}(m))), \mathit{tok}(m))$ ▷ $\mathit{ext}^{\text{send}}(A, i) \mathrel{+}= \mathit{fwd}_{\text{c}}(m).$
recv$_{\text{a}}(A, m, \mathit{hf}, i)$: $\mathit{hf} = \mathit{hd}(\mathit{fut}(m)) \wedge \mathit{hf} \neq \bot \wedge A = \mathit{id}(\mathit{hf}) \wedge$ $m \in \mathit{ext}^{\text{recv}}(A, i) \wedge (A, i) \in \mathit{dom}(\mathit{target})$ ▷ $\mathit{int}(A) \mathrel{+}= m.$	**recv**$_{\text{c}}(A, m, \mathit{hf}, i)$: $\mathit{hf} = \mathit{hd}(\mathit{fut}(m)) \wedge \mathit{hf} \neq \bot \wedge A = \mathit{id}(\mathit{hf}) \wedge$ $m \in \mathit{ext}^{\text{recv}}(A, i) \wedge (A, i) \in \mathit{dom}(\mathit{target}) \wedge$ $i = \mathit{prev}(\mathit{hf}) \wedge$ $\psi(\mathit{hf}, \mathit{hd}(\mathit{tl}(\mathit{fut}(m))), \mathit{tok}(m))$ ▷ $\mathit{int}(A) \mathrel{+}= m.$
deliver$_{\text{a}}(A, m, \mathit{hf})$: $\mathit{fut}(m) = \langle \mathit{hf} \rangle \wedge A = \mathit{id}(\mathit{hf}) \wedge m \in \mathit{int}(A)$ ▷ $\mathit{int}(A) \mathrel{+}= \mathit{fwd}_{\text{a}}(m).$	**deliver**$_{\text{c}}(A, m, \mathit{hf})$: $\mathit{fut}(m) = \langle \mathit{hf} \rangle \wedge A = \mathit{id}(\mathit{hf}) \wedge m \in \mathit{int}(A)$ $\psi(\mathit{hf}, \bot, \mathit{tok}(m))$ ▷ $\mathit{int}(A) \mathrel{+}= \mathit{fwd}_{\text{c}}(m).$

Fig. 5: Events of the abstract (left) and concrete (right) model, with differences highlighted.

to the previous and the next hop on the path. Both interfaces are defined as option types, indicated by the subscript $\bot$. When there is no previous or next hop, we assign $\bot$.

Our model's second environment parameter is

$$\mathit{auth}_{\text{a}} \subseteq \text{HF}_{\text{a}}^{*}, \tag{4}$$

the set of *authorized paths* along which packets are allowed to travel. Packets can also traverse only a part of an authorized path. To account for such partial paths we define $\mathit{auth}_{\text{a}}^{\rightleftarrows}$, the *fragment closure* of auth_{a}, as the set of paths hfs such that there exist a $\mathit{hfs}' \in \mathit{auth}_{\text{a}}$ and paths $\mathit{hfs}_1, \mathit{hfs}_2 \in \text{HF}_{\text{a}}^{*}$ such that $\mathit{hfs}' = \mathit{hfs}_1 \cdot \mathit{hfs} \cdot \mathit{hfs}_2$.

Note that authorized paths cannot be assumed to be actual paths in the network multigraph, since attackers on the control plane can interfere with the path construction (see §VI-A).

Our third parameter is the set of *compromised* nodes

$$\mathcal{N}_{\mathit{attr}} \subseteq \mathcal{N}. \tag{5}$$

All other nodes are called *honest*. This environment parameter only becomes relevant with the introduction of the adversary in the concrete model (§V-C), where the attacker has access to the keys of compromised nodes. We nevertheless introduce it here, since using the same environment parameters in all of our models simplifies our presentation.

The environment assumptions (ASM) expressed over these three parameters are introduced in the refinement of the abstract to the concrete model (§VI-A).

B. State

We model packet forwarding from a node's internal network to an inter-node link, and vice-versa, via two types of asynchronous channels: *internal* (one per node) and *external* (two per inter-node pair, one in each direction). We represent these channels as sets of packets PKT_{a}, defined below.

$$\mathbb{S}_{\text{a}} = (\!|\, \mathit{int} \in \mathcal{N} \to \mathcal{P}(\text{PKT}_{\text{a}}),\\ \mathit{ext} \in \mathcal{N} \times \mathcal{I} \times \mathcal{N} \times \mathcal{I} \to \mathcal{P}(\text{PKT}_{\text{a}}) \,|\!).$$

In the initial state s_{a}^{0}, all channels are empty. We overload the set inclusion operator to apply to states: A packet m is in state s, $m \in s$, iff $m \in \mathit{ran}(\mathit{int}(s)) \cup \mathit{ran}(\mathit{ext}(s))$. For a valid interface i of A with $\mathit{target}(A, i) = (B, j)$, we define $\mathit{ext}^{\text{send}}(A, i) = \mathit{ext}(A, i, B, j)$ and $\mathit{ext}^{\text{recv}}(A, i) = \mathit{ext}(B, j, A, i)$.

In the following definition of packets, we abstract from the payload and only model the packet-carried forwarding state:

$$\text{PKT}_{\text{a}} = (\!|\, \mathit{past} \in \text{HF}_{\text{a}}^{*},\ \mathit{fut} \in \text{HF}_{\text{a}}^{*},\ \mathit{hist} \in \text{HF}_{\text{a}}^{*} \,|\!).$$

A packet consists of the desired future path *fut*, and the (presumed) traversed path *past* in the reverse direction. The full path is $\mathit{rev}(\mathit{past}(m)) \cdot \mathit{fut}(m)$. While this splitting of the path simplifies our proofs, the forwarding path could equivalently be defined as a single sequence with a moving pointer indicating the current position on the path. We call a packet m *authorized*, if $\mathit{fut}(m) \in \mathit{auth}_{\text{a}}^{\rightleftarrows}$. Additionally, each packet records a path *hist*, also in reverse direction. It represents the packet's actual trajectory and is used to express security properties. This can be seen as a history variable.

C. Events

The events of the abstract model are given on the left-hand side of Fig. 5. The life cycle of a packet is captured by the following events: **dispatch-int**$_\text{a}$ creates a new packet containing an authorized future path in the internal channel of a node. The packet is transferred with alternating **send**$_\text{a}$ and **recv**$_\text{a}$ events between internal and external channels, according to the forwarding path contained in the packet. Finally, the packet is delivered to the end host with an event **deliver**$_\text{a}$. The events **dispatch-int**$_\text{a}$ and **deliver**$_\text{a}$ model the interaction with end hosts, whereas **send**$_\text{a}$ and **recv**$_\text{a}$ represent the border routers' packet forwarding actions. The additional **dispatch-ext**$_\text{a}$ event creates and sends a packet directly to an *ext* channel. It is not required for normal data plane operations, but serves to introduce a malicious sender at an inter-AS link in the refinement.

We now describe these events in more detail. The **dispatch-int**$_\text{a}$ and **dispatch-ext**$_\text{a}$ events create an authorized packet by setting its future path to (a fragment of) an authorized path and inserting it into an internal or external channel. The history is set to the empty sequence in both events, and the past path can be set arbitrarily to allow the refinement into attacker events, where the attacker may disguise the origin of the packet. The **send**$_\text{a}$ and **recv**$_\text{a}$ events both use the current hop field, i.e., the hop field at the head of the future path, to determine where the packet should be forwarded. Hence, they require a non-empty future path. The **recv**$_\text{a}$ event transfers a packet from the external channel at (A, i) to A's internal channel. The **send**$_\text{a}$ event takes a packet m from the internal channel and places the transformed packet $\mathit{fwd}_\text{a}(m)$ on the external channel at (A, i). The partial function $\mathit{fwd}_\text{a} : \text{PKT}_\text{a} \rightharpoonup \text{PKT}_\text{a}$ moves the current hop field into the past path and adds it to the history. It is defined for m with $\mathit{fut}(m) \neq \langle\rangle$ by

$$\mathit{fwd}_\text{a}(m) = (\!|\mathit{past} = \mathit{hd}(\mathit{fut}(m)) \mathbin{\#} \mathit{past}(m), \mathit{fut} = \mathit{tl}(\mathit{fut}(m)), \\ \mathit{hist} = \mathit{hd}(\mathit{fut}(m)) \mathbin{\#} \mathit{hist}(m)|\!).$$

We define the functions head $\mathit{hd} : \text{HF}^*_{\text{a}\,\perp} \to \text{HF}_{\text{a}\,\perp}$ and tail $\mathit{tl} : \text{HF}^*_{\text{a}\,\perp} \to \text{HF}^*_{\text{a}\,\perp}$ by $\mathit{hd}(x \mathbin{\#} xs) = x$ and $\mathit{tl}(x \mathbin{\#} xs) = xs$ and by mapping $\langle\rangle$ and $\perp$ to $\perp$ in both functions.

The **deliver**$_\text{a}$ event models delivering a packet m containing a single hop field in its future path to an end host. However, we do not explicitly model end hosts and their state. Hence, we simply add the packet $\mathit{fwd}_\text{a}(m)$ to the internal channel of the AS and thereby push the last hop field into the *past* and *hist* paths (security properties are expressed over *hist*).

D. Properties

Path authorization states that packets can only traverse the network along authorized paths. This ensures that the data plane enforces the control plane-level routing policies. Formally, for all packets m in a state s, $\mathit{rev}(\mathit{hist}(m)) \in \mathit{auth}^{\rightleftarrows}_\text{a}$. Recall that the order of nodes is reversed in *hist*. We strengthen this to an inductive invariant by adding the future path:

$$\forall m \in s.\ \mathit{rev}(\mathit{hist}(m)) \cdot \mathit{fut}(m) \in \mathit{auth}^{\rightleftarrows}_\text{a}. \tag{6}$$

For the proof, note that new packets are required to be authorized and for existing packets $\mathit{rev}(\mathit{hist}(m)) \cdot \mathit{fut}(m)$ remains invariant during their forwarding.

We furthermore prove that *detectability* is an invariant: all traversed hops are recorded on (i.e., a prefix of) the past path:

$$\forall m \in s.\ \mathit{hist}(m) \leq \mathit{past}(m). \tag{7}$$

This property is independent of auth_a and follows directly from the events' definitions. Our presentation will focus on path authorization, as it is the data plane's central security property.

V. Concrete Parametrized Model

We refine the abstract forwarding protocol into a concrete model. In this model, the packets' hop fields include (generic) cryptographic hop validation fields to secure the authorized paths against a Dolev–Yao attacker (§V-C). The states have the same structure as in the abstract model except that the *int* and *ext* channels now contain concrete packets (§V-A). We present the concrete model's events in §V-D and the refinement in §VI.

The concrete model retains the environment parameters of the abstract model (§IV-A), and adds the three protocol parameters, introduced below. One of them is the cryptographic check that ASes apply to their hop validation fields, which allows us to abstract from the concrete cryptographic mechanism used. We focus on the setting for directed path authorization here, and defer undirected path authorization to §VIII-C. We compare both classes of protocols in §IX-A.

A. Cryptographic terms, hop fields, packets and states

We introduce an algebra $\mathbb{T}$ of cryptographic message terms:

$$\mathbb{T} = \mathscr{N} \mid \mathscr{I}_\perp \mid \mathbb{N} \mid K_{\mathscr{N}} \mid \langle \mathbb{T},\ \mathbb{T},\ \ldots,\ \mathbb{T} \rangle \mid \mathsf{H}(\mathbb{T}).$$

Terms consist of node identifiers, interfaces, natural numbers (e.g., for timestamps), keys (one per node), as well as finite sequences, and cryptographic hashes of terms. We define message authentication codes (MACs) using hashing by $\mathsf{MAC}_k(m) = \mathsf{H}(\langle k, m \rangle)$. Our framework also supports encryption and signatures, which we do not use here.

Concrete hop fields extend the abstract hop fields in HF_a with a *hop validation field* (*hvf*), a cryptographic authenticator that protects the authenticated hop information:

$$\text{HF}_\text{c} = (\!|\, \mathit{id} \in \mathscr{N},\ \mathit{prev} \in \mathscr{I}_\perp,\ \mathit{next} \in \mathscr{I}_\perp,\ \mathit{hvf} \in \mathbb{T} \,|\!). \tag{8}$$

We define the function $\mathit{to}_\text{a} : \text{HF}_\text{c} \to \text{HF}_\text{a}$ projecting concrete hop fields to abstract hop fields by dropping *hvf* and we lift it element-wise to paths.

We next define concrete packets as follows:

$$\text{PKT}_\text{c} = (\!|\, \mathit{tok} \in \mathbb{N},\ \mathit{past} \in \text{HF}^*_\text{c},\ \mathit{fut} \in \text{HF}^*_\text{c},\ \mathit{hist} \in \text{HF}^*_\text{a} \,|\!).$$

The past and future paths are sequences of concrete hop fields, while the history remains abstract. Concrete packets contain an additional *token* field *tok*, which is used by some instances for a source-supplied unique packet identifier. We let $\mathit{terms}(\mathit{hf}) = \{\mathit{hvf}(\mathit{hf})\}$ and lift this function to paths and packets by taking the union of terms in all hop fields and in the past and future

paths, respectively. For a set T of terms and for a hop field, path, or packet x, we write $x \in T$ for $\mathit{terms}(x) \subseteq T$.

The concrete state space $\mathbb{S}_\mathrm{c}$ has the same record structure as the abstract $\mathbb{S}_\mathrm{a}$, but the channels now carry concrete packets.

B. Protocol parameters and authorized paths

We define three protocol parameters. The first is a *cryptographic validation check*

$$\psi : \mathrm{HF_c} \times \mathrm{HF_{c\perp}} \times \mathbb{N} \to \mathbb{B}, \tag{9}$$

which each border router performs to check the validity of its hop field. This parameter abstracts the cryptographic structure of the hop validation field, which is only determined in concrete protocol instances. Here, $\psi(\mathit{hf}_A, \mathit{hf}_B, \mathit{tok})$ holds iff the hop field hf_A is authentic given the next hop field hf_B (if any, and $\perp$ otherwise) and the packet's token *tok*. We also define a function

$$\Psi : \mathrm{HF_c^*} \times \mathbb{N} \to \mathrm{HF_c^*},$$

where $\Psi(\mathit{hfs}, \mathit{tok})$ returns the longest prefix of *hfs* such that for every hop field hf_A in *hfs*, and its successor hop field hf_B ($\perp$, if none exists), $\psi(\mathit{hf}_A, \mathit{hf}_B, \mathit{tok})$ holds. We call a path *hfs cryptographically valid* if $\Psi(\mathit{hfs}, \mathit{tok}) = \mathit{hfs}$.

We define the set of concrete authorized paths, $\mathit{auth}_\mathrm{c} \subseteq \mathrm{HF_c^*}$, as the set of paths *hfs* that are cryptographically valid, and whose projection to $\mathrm{HF_a^*}$ is authorized:

$$\mathit{auth}_\mathrm{c} = \{\mathit{hfs} \mid \Psi(\mathit{hfs}, \mathit{tok}) = \mathit{hfs} \wedge \mathit{to}_\mathrm{a}(\mathit{hfs}) \in \mathit{auth}_\mathrm{a}\}.$$

Similar to the abstract model, a concrete packet m is authorized, if $\mathit{fut}(m)$ is a fragment of an authorized path, i.e., $\mathit{fut}(m) \in \mathit{auth}_\mathrm{c}^{\rightleftarrows}$. To improve readability, we will often omit the parameters hf_B and *tok* from ψ and Ψ and sometimes leave the projection to_a implicit.

To achieve path authorization, protocols use the *hvf* to protect the future (abstract) path. The second protocol parameter is

$$\mathit{extract} : \mathbb{T} \to \mathrm{HF_a^*}, \tag{10}$$

which is intended to extract this path from a given *hvf*. For instance, in SCION, the hop validation field consists of a MAC over the hop's *id* and interfaces and, recursively, the next hop's *hvf*, allowing for such extraction. We lift *extract* to hop fields by $\mathit{extract}(\mathit{hf}) = \mathit{extract}(\mathit{hvf}(\mathit{hf}))$ and to paths by defining $\mathit{extract}(\langle\rangle) = \langle\rangle$ and $\mathit{extract}(\mathit{hf} \# \mathit{hfs}) = \mathit{extract}(\mathit{hf})$. In §VI-B, we will define a consistency condition requiring that, on cryptographically valid paths derivable by the attacker, *extract* coincides with to_a.

The third protocol parameter is a set of cryptographic terms

$$\mathit{ik}_0^+ \subseteq \mathbb{T}. \tag{11}$$

It allows protocol instances to give the attacker additional terms and is used in the definition of the intruder knowledge below.

$$\frac{t \in H}{t \in \mathit{DY}^{\downarrow}(H)} \qquad \frac{\langle t_1, \ldots, t_n\rangle \in \mathit{DY}^{\downarrow}(H)}{t_i \in \mathit{DY}^{\downarrow}(H)}\ 1 \le i \le n$$

$$\frac{t \in \mathscr{N} \cup \mathscr{I}_\perp \cup \mathbb{N}}{t \in \mathit{DY}^{\uparrow}(H)} \qquad \frac{t \in H}{t \in \mathit{DY}^{\uparrow}(H)}$$

$$\frac{t \in \mathit{DY}^{\uparrow}(H)}{\mathit{Hash}(t) \in \mathit{DY}^{\uparrow}(H)} \qquad \frac{t_1 \in \mathit{DY}^{\uparrow}(H) \ \cdots \ t_n \in \mathit{DY}^{\uparrow}(H)}{\langle t_1, \ldots, t_n\rangle \in \mathit{DY}^{\uparrow}(H)}$$

Fig. 6: Rules for Dolev–Yao message decomposition ($\mathit{DY}^{\downarrow}$) and composition ($\mathit{DY}^{\uparrow}$).

C. Attacker model

We model a Dolev–Yao adversary who can eavesdrop on and inject new packets in all *int* and *ext* channels, but only has access to the keys of compromised nodes. We first define the attacker's message derivation capabilities, which are used in the attacker events, introduced in §V-D.

As usual, we model the attacker's knowledge as a set of terms and her message derivation capabilities as a closure operator *DY* on sets of terms. Our formalization of *DY* is based on [34] and defines $\mathit{DY}(H) = \mathit{DY}^{\uparrow}(\mathit{DY}^{\downarrow}(H))$ for a set of terms H as the composition of two closure operators defined by the rules in Fig. 6. The decomposition closure $\mathit{DY}^{\downarrow}(H)$ closes H under the projection of sequences to their elements and the composition closure $\mathit{DY}^{\uparrow}(H)$ includes all public terms (i.e., node identifiers, interfaces, and numbers) and closes H under the construction of sequences and hashes.

We define the intruder knowledge in a state $s \in \mathbb{S}_\mathrm{c}$ as the Dolev–Yao closure (*DY*) of the set of terms $\mathit{ik}(s)$, defined by

$$\mathit{ik}_0 = \{\mathit{terms}(x) \mid x \in \mathit{auth}_\mathrm{c}\} \cup \{K_i \mid i \in \mathscr{N}_{\mathit{attr}}\}, \tag{12}$$

$$\mathit{ik}(s) = \mathit{ik}_0 \cup \mathit{ik}_0^+ \cup \bigcup_{m \in s} \mathit{terms}(m). \tag{13}$$

The set $\mathit{ik}(s)$ is the union of the initial intruder knowledge ik_0, consisting of authorized paths (i.e., the *hvf* of their hop fields) and compromised nodes' keys, additional terms ik_0^+, and all terms in packets of state s.

Invariant. Since in reachable states s all packets $m \in s$ are intruder-derivable, $\mathit{DY}(\mathit{ik}(s)) = \mathit{DY}(\mathit{ik}_0 \cup \mathit{ik}_0^+)$, which we show is an invariant. We will henceforth use $\mathit{DY}(\mathit{ik}_0 \cup \mathit{ik}_0^+)$ instead of $\mathit{DY}(\mathit{ik}(s))$.

D. Events

Each event of the abstract model is refined into a similar event of the concrete model (Fig. 5, right). In the events' guards we omit the state and just write *ik*. The concrete model retains the same packet life-cycle of the abstract model (§IV-C). The **dispatch-int**$_\mathrm{c}$ and **dispatch-ext**$_\mathrm{c}$ events can send arbitrary attacker-derivable packets, instead of only authorized packets as in the abstract model. To defend against the attacker, we introduce interface and cryptographic checks in **send**$_\mathrm{c}$, **recv**$_\mathrm{c}$, and **deliver**$_\mathrm{c}$. We now discuss the concrete model's events in more detail.

1) Attacker events: The two attacker events **dispatch-int**$_c$ and **dispatch-ext**$_c$ model the attacker's active capabilities to send a packet on any AS' internal or external channel, regardless of whether the AS is honest or compromised. In both events, the packet m created by the attacker may contain arbitrary past and future paths, but its hop validation fields must be derivable, i.e., $\mathit{terms}(m) \subseteq \mathit{DY}(\mathit{ik}(s))$. Note that the event **dispatch-int**$_c$ still covers honest senders, as the attacker knows all authorized paths.

Similar to their abstract counterparts, both events set the history field *hist* to $\langle\rangle$. The motivation for this is to rule out unavoidable attacks where an attacker with access to an AS' external channels modifies a packet's forwarding path. For example, consider Fig. 2a and suppose that the attacker has access to D's external channels. Then D may receive a packet arriving on the left path from F, exchange its forwarding path by the right path, and forward the modified packet to C, thereby (trivially) violating path authorization. By resetting the history we effectively consider all packets sent by the attacker as new ones. Consequently, path authorization must hold separately for the packets before and after the replacement of the forwarding path by the attacker. One can also argue that an attacker arbitrarily modifying packets en-route makes it generally impossible to correlate packets sent by the attacker with those the attacker has previously received.

2) Honest events: The honest events are **send**$_c$, **recv**$_c$, and **deliver**$_c$. To secure the protocol against the attacker introduced in this model, border routers now perform two validation checks: First, upon receiving a packet from another node, **recv**$_c$ includes the guard $i = \mathit{prev}(\mathit{hf})$ to check that interface i over which the packet is received matches the interface *prev* of the packet's current hop field *hf*. Second, all honest events check the cryptographic *hop validation field* that is added to hop fields in this refinement using the check $\psi(\mathit{hf}, \mathit{hf}', \mathit{tok})$, where *hf*, *hf*′, and tok are the packet's current hop field, next hop field, and token, respectively. This check ensures that the hop field (and indeed the whole or partial path) is authorized.

The events **send**$_c$ and **deliver**$_c$ use the function fwd_c to forward a packet. This function is defined similarly to fwd_a, but the *tok* field is not modified and the hop field being moved from the future to the past path is converted from HF_c to HF_a using to_a before it is added to $\mathit{hist}(m)$.

VI. Refinement

The refinement proof rests on several global *assumptions* about the control plane and and on a set of *conditions* about the authentication mechanism used. To establish that a concrete protocol satisfies the path authorization and detectability properties, it suffices to define the protocol's authentication mechanism and discharge the associated conditions (see §VII).

A. Control plane assumptions

We define environment assumptions about the authorized paths auth_a constructed by the control plane. There are two types of assumptions. First, there are two assumptions about the correct functioning of the control plane, which is independent of the data plane. Second, additional assumptions define a closure on the set of authorized paths in order to exclude attacks on the routing policies of colluding ASes. These are upper and lower bounds on auth_a, respectively.

The first control plane assumption is that authorized paths are *terminated*: the first hop field's *prev* is $\perp$ and the last hop field's *next* is $\perp$, except for when the respective hop field belongs to an attacker. Second, we assume that authorized paths are *interface-valid*: interfaces of adjacent hop fields on a path point to the same link, except for when both hop fields belong to attacker nodes. This exception accounts for out-of-band communication by adversaries (*wormholes*).

To formalize interface validity, we introduce the interface validity predicate $\phi : \mathrm{HF}_a \times \mathrm{HF}_{a\perp} \to \mathbb{B}$. In the following, we let hf_A (resp. hf_B) denote a hop field for which $\mathit{id}(\mathit{hf}_A) = A$ (resp. $\mathit{id}(\mathit{hf}_B) = B$). The parameters of ϕ are the current hop field hf_B and a preceding hop field hf_A. For the first hop field, which has no predecessor, no interface check is necessary.

$$\begin{aligned}\phi(\mathit{hf}_B, \perp) &= \mathsf{true}\\ \phi(\mathit{hf}_B, \mathit{hf}_A) &= (\mathit{target}(A, \mathit{next}(\mathit{hf}_A)) = (B, \mathit{prev}(\mathit{hf}_B)))\\ &\quad \vee (A \in \mathscr{N}_{\mathit{attr}} \wedge B \in \mathscr{N}_{\mathit{attr}})\end{aligned}$$

We define $\Phi : \mathrm{HF}_a^* \to \mathrm{HF}_a^*$ as the longest prefix of a sequence of hop fields *hfs* such for all hop fields hf_B on *hfs* and their respective predecessor hf_A on *hfs*, $\phi(\mathit{hf}_B, \mathit{hf}_A)$ holds. ASM 1 and ASM 2 formalize the correctness of the control plane.

ASM 1: Terminated: A hop field *hf* with $\mathit{id}(\mathit{hf}) \notin \mathscr{N}_{\mathit{attr}}$ on $\mathit{hf} \# \mathit{hfs} \in \mathit{auth}_a$ (resp. on $\mathit{hfs} \cdot \langle \mathit{hf} \rangle \in \mathit{auth}_a$) has $\mathit{prev}(\mathit{hf}) = \perp$ (resp. $\mathit{next}(\mathit{hf}) = \perp$).

ASM 2: Interfaces valid: All paths $\mathit{hfs} \in \mathit{auth}_a$ are interface-valid: $\Phi(\mathit{hfs}) = \mathit{hfs}$.

The second type of assumptions extends the set of authorized paths with certain path modifications to avoid trivial violations of path authorization where an attacking source violates the routing policy of compromised ASes. In this section, we only present path modifications possible in directed protocols.

For example, assume that ASes E and F in Fig. 2 are compromised. Since E is on the right path G–E–D–C–A, she can take the suffix E–D–C–A, change her own hop field (and issue a new *hvf* using the compromised key) such that *prev* points to F, and prepend a new hop field for F to obtain the path F–E–D–C–A. None of these changes require consent from the other on-path nodes, since each AS only decides on the authorization of the partial paths with its respective AS at the beginning and path extensions are implicitly authorized. Hence, this only constitutes an attack on the compromised ASes E and F's own routing policy, not on any honest AS' policy. To account for these unavoidable attacks, we accept such path modifications as authorized in ASM 3–ASM 6.

ASM 3: Empty & Single: $\langle\rangle \in \mathit{auth}_a$ and $\langle \mathit{hf} \rangle \in \mathit{auth}_a$ for all $\mathit{id}(\mathit{hf}) \in \mathscr{N}_{\mathit{attr}}$.

ASM 4: Prepend: If the first hop field belongs to the attacker, she can prepend another attacker hop field. Formally, if $\mathit{hf}_B \# \mathit{hfs} \in \mathit{auth}_a$, $B \in \mathscr{N}_{\mathit{attr}}$, and $A \in \mathscr{N}_{\mathit{attr}}$ then $\mathit{hf}_A \# \mathit{hf}_B \# \mathit{hfs} \in \mathit{auth}_a$.

ASM 5: Suffix: The attacker can take a path's *suffix* if her hop field is the suffix' head. Formally, if $hfs' \cdot hf_A \# hfs \in auth_{\text{a}}$ and $A \in \mathcal{N}_{attr}$ then $hf_A \# hfs \in auth_{\text{a}}$.

ASM 6: Modify: If the first hop field belongs to the attacker, she can modify its *prev*. Formally, if $hf_A \# hfs \in auth_{\text{a}}$, $next(hf'_A) = next(hf_A)$ and $A \in \mathcal{N}_{attr}$ then $hf'_A \# hfs \in auth_{\text{a}}$. Note that $id(hf'_A) = id(hf_A) = A$.

These are not merely assumptions of our protocol model or the specific attacker model that we employ but are inherent to the path authorization mechanism of directed protocols. Undirected protocols (such as ICING) require that the entire path is authorized by each on-path AS. As we show in §VIII-C, ASM 3–ASM 6 can then be replaced by weaker assumptions.

B. Conditions on authentication mechanisms

We define five conditions that relate the three protocol parameters ψ, *extract*, and ik_0^+ introduced in Eqs. (9)–(11) with each other and with the environment parameters $\mathcal{N}_{attr}$ and $auth_{\text{a}}$ (via $auth_{\text{c}}$). These conditions are needed in the refinement proof in §VI-C. We will have to prove these conditions for any instance of the concrete model, which poses no significant difficulty for the instances we present in §VII.

COND 1 and COND 2 together require that the attacker cannot derive valid hop fields for honest nodes that are not already contained in $auth_{\text{c}}$. They also constrain the parameter ik_0^+, since instances cannot provide the attacker with terms that allow her to create valid but unauthorized hop fields.

COND 1: Attacker knowledge derivation: $hf \in DY(ik_0 \cup ik_0^+)$, $\psi(hf, hf', tok)$, and $id(hf) \notin \mathcal{N}_{attr}$ imply $hf \in DY^{\downarrow}(ik_0 \cup ik_0^+)$.

COND 2: Attacker knowledge decomposition: $hf \in DY^{\downarrow}(ik_0 \cup ik_0^+)$ and $\psi(hf, hf', tok)$ imply $\exists hfs \in auth_{\text{c}}.\ hf \in hfs$.

COND 3 and COND 4 relate $\Psi(hfs)$, the longest cryptographically valid prefix of *hfs*, to $extract(hfs)$, which extracts the subsequent path from the first hop field in *hfs*. In particular, on a cryptographically valid path they coincide (modulo to_{a}). For instance, consider the SCION path given in Fig. 4. hvf_D is $\mathsf{MAC}_{K_D}(\langle hf_D, \mathsf{MAC}_{K_B}(\langle hf_B, \mathsf{MAC}_{K_A}(hf_A)\rangle)\rangle)$ and in this instance *extract* would be defined to extract the forwarding data from the nested MACs, i.e., $extract(hvf_D) = \langle hf_D, hf_B, hf_A\rangle$. This is exactly the path in Fig. 4 of D and following ASes.

COND 3: Path prefix of extract: $\Psi(hfs) \leq extract(hfs)$.

COND 4: Extract prefix of path: If $\Psi(hfs) = hfs$, then $extract(hfs) \leq hfs$.

Finally, COND 5 requires the *hvf* to protect the *tok* field. This ensures that a hop field that is valid for a certain token cannot be used to forward a packet with a different token.

COND 5: Token protected: $\psi(hf, hf', tok)$ and $\psi(hf, hf'', tok')$ imply $tok' = tok$.

We prove the following lemmas, which are helpful for the refinement proof below. The first lemma states that the extraction of a cryptographically valid path is the path itself.

Lemma 1. *If $\Psi(hfs) = hfs$ then $extract(hfs) = hfs$.*

The second lemma asserts that the valid prefix of any extension of an attacker-extractable hop field is authorized.

Lemma 2. *Suppose $hf \in DY^{\downarrow}(ik_0 \cup ik_0^+)$ for some hop field hf. Then $\Psi(hf \# hfs) \in auth_a^{\rightleftarrows}$ for all paths hfs.*

Proof. If *hf* is invalid, i.e., $\neg\psi(hf, hd(hfs), tok)$ for some *tok*, then $\Psi(hf \# hfs) = \langle\rangle$ and the conclusion holds by ASM 3. Otherwise, we can apply COND 2 and obtain hfs', hfs_0, hfs_1, and hfs_2 such that $hfs' \in auth_{\text{c}}$, $hfs' = hfs_0 \cdot hfs_1$, and $hfs_1 = hf \# hfs_2$. Since $hfs' \in auth_{\text{c}}$, $\Psi(hfs') = hfs'$ and thus also $\Psi(hfs_1) = hfs_1$. Then we can apply Lemma 1 and obtain $extract(hfs_1) = extract(hf) = hfs_1$. Since hfs_1 is a suffix of the authorized path hfs', we have $extract(hf) \in auth_{\text{a}}^{\rightleftarrows}$. Finally, from COND 3, we have $\Psi(hf \# hfs) \leq extract(hf)$. Since $auth_{\text{a}}^{\rightleftarrows}$ is closed under prefixing, $\Psi(hf \# hfs) \in auth_{\text{a}}^{\rightleftarrows}$. □

C. Refinement proof

To show that the concrete model refines the abstract one (assuming ASM and COND), we first define the refinement mapping $\pi_0 \colon \mathbb{S}_{\text{c}} \to \mathbb{S}_{\text{a}}$ on states as the element-wise mapping of the *int* and *ext* channels under a function that maps concrete packets to abstract packets. We again overload to_{a}, which was defined as mapping concrete (sequences of) hop fields to abstract ones in §V-A and define $to_{\text{a}} : \text{PKT}_{\text{c}} \to \text{PKT}_{\text{a}}$ by

$$to_{\text{a}}(m) = (\!|\, past = to_{\text{a}}(past(m)),\ fut = \Phi(to_{\text{a}}(\Psi(fut(m)))),\ hist = hist(m)\ |\!).$$

Because of the interface and cryptographic checks that we introduce, no forwarding occurs on invalid hop fields. We thus map $fut(m)$ to $\Phi(to_{\text{a}}(\Psi(fut(m))))$. This describes the mapping of packets in *int* channels. For packets in *ext* channels, we modify the Φ function to additionally check that the first hop field is interface-valid with the channel, but to simplify the presentation, we elide this check here. The refinement mapping $\pi_1 \colon E_{\text{c}} \to E_{\text{a}}$ maps each event on the right side of Fig. 5 to the corresponding event on the left side, where packet and hop field parameters are transformed using to_{a}.

In the refinement proof, we show that each concrete event e can be simulated by its abstract counterpart $\pi_1(e)$. This is straightforward for the honest events, since the concrete model only adds guards and the concrete guards imply the validity of the first hop field (ensuring that *fut* is not mapped to $\langle\rangle$). The state updates of these events preserve the refinement relation. The difficult cases are the attacker events. In particular, we must show that the concrete dispatch events' guards imply their abstract counterparts. This is formalized in the following theorem stating that the attacker can only derive paths that, restricted to their valid prefix, are authorized.

Theorem 3 (Attacker refinement)**.** *If $m \in DY(ik_0 \cup ik_0^+)$ then $\Phi(to_a(\Psi(fut(m)))) \in auth_a^{\rightleftarrows}$.*

Proof. We prove this theorem by induction over $hfs = fut(m)$. Here, we only sketch the proof and focus on the interesting cases in which at least two hop fields are left, i.e., $hfs = hf_A \# hf_B \# hfs'$, and both have valid hop validation fields and interfaces. Recall that the subscript identifies the node; i.e., we

use hf_A to denote a hop field for which $id(hf_A) = A$ holds. As above, we elide the projection to_{a} to improve readability.

- $A \notin \mathcal{N}_{attr}$: If the attacker can derive hf_A without K_A, then by COND 1 hf_A must already be in $DY^{\downarrow}(ik_0 \cup ik_0^{+})$. Then by Lemma 2, we have $\Psi(hfs) \in auth_{\mathrm{a}}^{\rightleftarrows}$ and by fragment closure also $\Phi(\Psi(hfs)) \in auth_{\mathrm{a}}^{\rightleftarrows}$ as required.
- $A \in \mathcal{N}_{attr}$ and $B \notin \mathcal{N}_{attr}$: This is the most difficult case. By COND 1, $hf_B \in DY^{\downarrow}(ik_0 \cup ik_0^{+})$, and by COND 2, we obtain hfs_{gen} such that $hfs_{gen} \in auth_{\mathrm{c}}$ and $hf_B \in hfs_{gen}$. Paths in $auth_{\mathrm{a}}$ and, by extension, paths in $auth_{\mathrm{c}}$ are terminated (ASM 1). However, by the case assumption, hf_B is interface-valid with hf_A as the preceding AS and thus cannot be terminated. Hence, there must be a *hf'* preceding hf_B on hfs_{gen}. As $hfs_{gen} \in auth_{\mathrm{c}}$, there exist hfs_{pre} and hfs_{post} such that

  $$hfs_{gen} = hfs_{pre} \cdot hf' \# hf_B \# hfs_{post} \in auth_{\mathrm{c}}.$$

  Since $hf' \in DY^{\downarrow}(ik_0 \cup ik_0^{+})$, we can apply Lemma 2 and have $\Psi(hf' \# hf_B \# hfs') \in auth_{\mathrm{a}}^{\rightleftarrows}$. As authorized hop fields are valid, $\psi(hf', hf_B, tok)$ for some *tok*. Hence for some hfs'_{pre} and hfs'_{post}

  $$hfs'_{pre} \cdot (hf' \# \Psi(hf_B \# hfs')) \cdot hfs'_{post} \in auth_{\mathrm{a}}.$$

  Finally, we use the assumptions on authorized paths to show that the attacker can remove the hop fields hfs'_{pre} preceding *hf'* (ASM 5) and swap out *hf'* for hf_A (ASM 6). To apply these assumptions, we must show that $id(hf') \in \mathcal{N}_{attr}$ and that *hf'* and hf_A have the same *id* and *next*. hf_B is interface-valid with the predecessor hf_A (by the case assumption) and with the predecessor *hf'* (by the assumption on the interface-validity of authorized paths, ASM 2). Thus hf_A and *hf'* must have the same AS identifier $id(hf') = A \in \mathcal{N}_{attr}$ and interface $next(hf') = next(hf_A)$. Hence, we have $\Psi(hfs) \in auth_{\mathrm{a}}^{\rightleftarrows}$ and by fragment closure also $\Phi(\Psi(hfs)) \in auth_{\mathrm{a}}^{\rightleftarrows}$.
- $A \in \mathcal{N}_{attr}$ and $B \in \mathcal{N}_{attr}$: This case uses the suffix and prepend assumptions on authorized paths of §VI-A. By the induction hypothesis $\Phi(\Psi(hf_B \# hfs')) \in auth_{\mathrm{a}}^{\rightleftarrows}$. By case assumption of the validity of hf_B, there is a hfs_{pre}, hfs_{post} such that $hfs_{pre} \cdot hf_B \# \Phi(\Psi(hfs')) \cdot hfs_{post} \in auth_{\mathrm{a}}$. By ASM 5, we can take the suffix: $hf_B \# \Phi(\Psi(hfs')) \cdot hfs_{post} \in auth_{\mathrm{a}}$. Finally, ASM 4 allows prepending hf_A to this authorized path.

COND 5 is required to show that *hfs* and hfs_{gen} are valid for the same *tok* in the second case above (however, we have elided tokens from the presentation). ASM 3 is needed in cases we have not shown, e.g., when $A \in \mathcal{N}_{attr}$ and hf_B is invalid. □

VII. Instances

We now instantiate the concrete parametrized model to several protocols from the literature and variants thereof. To do so, we instantiate the model's protocol parameters and prove the associated conditions (§VI-B).

A. SCION

SCION embeds a MAC in each hop validation field. The *tok* field is not used. In this instance, hvf_A for each hop A with a next hop B is the MAC computed over the abstract hop field of A (containing *prev*, *next* and *id*), the abstract hop field of B, and the hvf_B field, using the symmetric key K_A shared by all border routers in the AS A:

$$hvf_A = \mathsf{MAC}_{K_A}(\langle hf_A, hf_B, hvf_B \rangle)\,. \tag{14}$$

The last AS A has no successor, so $hvf_A = \mathsf{MAC}_{K_A}(hf_A)$.

We instantiate ψ with the check of Eq. (14) and set $ik_0^{+} = \emptyset$. In this and all following instances, we only define *extract* for valid patterns; all other patterns are mapped to $\langle\rangle$.

$$extract(\mathsf{MAC}_{K_A}(hf_A)) = hf_A$$
$$extract(\mathsf{MAC}_{K_A}(\langle hf_A, hf_B, hvf_B \rangle)) = hf_A \# extract(hvf_B)$$

To show that this model of SCION inherits the security properties proven in the parametrized models, we prove the parametrized model's conditions COND 1–COND 5 (§VI-B). First, the intruder knowledge only contains keys and MACs, which cannot be decomposed, hence $DY^{\downarrow}(ik_0) = ik_0$. We prove COND 1, COND 2, and COND 5 by unfolding the definitions of ik_0, $auth_{\mathrm{c}}$, and ψ. We establish COND 3 and COND 4 by routine inductions over *hfs*.

Variants. By dropping hf_B from Eq. (14) we obtain the variant described in §II-E where we instantiate ψ with the check Eq. (1). The proof of the conditions is almost identical.

B. EPIC

Protocols with a packet-carried forwarding state face a dilemma: secure authenticators must be long, but low communication overhead requires short authenticators. EPIC uses short authenticators, but proposes a mechanism that leverages a stateful *replay-suppression system* to limit the impact of a successful brute-force attack.

The key insight behind EPIC is that the security goals of ASes are driven by economic interests that are unaffected by a few individual packets being forwarded on unauthorized paths. Hence, instead of *preventing* brute-force attacks, EPIC guarantees that each successful attack can only be used to send *one* packet along an unauthorized path. This is ensured by end-host-generated, one-time hop validation fields instead of the static validation fields of SCION.

Protocol details. The EPIC level 1 protocol uses a static *hop authenticator* σ, which is created on the control plane and almost identical to the hop validation field of Eq. (1):

$$\sigma_A = \mathsf{MAC}_{K_A}(\langle hf_A, \mathsf{H}(\sigma_B) \rangle)\,. \tag{15}$$

If A has no successor, then $\sigma_A = \mathsf{MAC}_{K_A}(hf_A)$.

End hosts obtain the hop authenticators for a path (which are public), generate a unique token *tok* for each packet, and compute for each hop field the packet-specific *hvf* as

$$hvf_A = \langle \mathsf{H}(\sigma_A), \mathsf{MAC}_{\sigma_A}(tok) \rangle. \tag{16}$$

To check the validity of the *hvf*, a border router A re-computes its own σ_A (using its key K_A and $\mathsf{H}(\sigma_B)$ from the successor hop field) and then re-computes hvf_A from σ_A and the token included in the packet.

The hop validation field is shortened to a few bytes in EPIC, and can, with considerable but realistic effort, be brute-forced by an attacker. This allows the attacker to send packets that have valid authenticators, but are unauthorized. While the shortening of the hop fields is not reflected in the above equation, it motivates us to introduce a stronger attacker model that reflects the brute-force abilities of the attacker in §VIII-B.

Two mechanisms in EPIC limit the effects of such attacks: First, the *hvf* is bound to a packet's token. Second, a replay-suppression system at each border router prevents multiple packets with the same token from being forwarded. Consequently, a brute-force attack on the *hvf* can only be used to send a single packet.

Only by brute-forcing the underlying static hop authenticator σ could the attacker dynamically create valid but unauthorized *hvf* for arbitrary tokens and thus send an unlimited number of packets. However, σ is a long authenticator. Hence, the success probability of such a brute-force attack is negligible.

Formalization. We instantiate the predicate ψ with the conjunction of Eqs. (15) and (16). We define *extract* such that it first extracts the hop authenticator, and then the path. Patterns not covered below map to $\langle\rangle$.

$$\begin{aligned}
&\mathit{extract}(\langle h, \mathsf{MAC}_{\sigma_A}(\mathit{tok})\rangle) = \mathit{extract'}(\sigma_A)\\
&\mathit{extract'}(\mathsf{MAC}_{K_A}(\mathit{hf}_A)) = \mathit{hf}_A\\
&\mathit{extract'}(\mathsf{MAC}_{K_A}(\langle \mathit{hf}_A, \mathsf{H}(\sigma_B)\rangle)) = \mathit{hf}_A \mathbin{\#} \mathit{extract'}(\sigma_B)
\end{aligned}$$

According to our definition of the intruder knowledge given in Eq. (12), the attacker knows the *hvf* values of all authorized paths. We define ik_0^+ such that the attacker additionally knows all hop authenticators of authorized paths.

We show that EPIC is an instance of our concrete parametrized model, and thus inherits the security properties proven in the abstract model. The proof is closely related to that of the SCION instance, but requires additional case distinctions since ik_0^+ provides the attacker with more ways to derive terms, i.e., from hop authenticators. We also need to prove a lemma stating that if a hop authenticator from ik_0^+ is used to create a *hvf* of a valid hop field with some *tok* value, then that hop field is contained in an authorized path.

Variants. We verify EPIC level 1 in a strong attacker model (see §VIII-B). We verify level 2, also in the same model. We do not model or verify the separate replay suppression system.

C. ICING

Among the protocols we study, ICING [18] provides the strongest security properties, albeit at the cost of the highest overhead [17]. It allows ASes to authorize the entire path and is thus an instance of the undirected setting. This requires a separate concrete model, whose parameters, assumptions, and conditions slightly differ from those presented above. We discuss this model in §VIII-C.

ICING uses *proofs of consent* (PoCs) to achieve path authorization. These are created by applying a pseudorandom function (PRF) using a *tag key* on the entire forwarding path. The tag key for each AS is derived from its master key K_A, and the local hop field hf_A. In our symbolic model, PRFs and MACs are modeled identically, we thus use MACs in our definitions. We use these PoCs as hop validation fields:

$$\mathit{hvf}_A = \mathsf{MAC}_{\langle K_A,\ \mathit{hf}_A\rangle}(\mathit{rev}(\mathit{past}(m)) \cdot \mathit{fut}(m))\,. \tag{17}$$

We define $ik_0^+ = \emptyset$. The function *extract* requires extracting the entire path (past and future) in the undirected setting:

$$\mathit{extract}(\mathsf{MAC}_{\langle K_A,\ \mathit{hf}_A\rangle}(\mathit{hfs})) = \mathit{hfs}. \tag{18}$$

Variants. We verified two other versions of the protocol. In the first one, the hop validation field consists of ICING's *path authenticator*, which includes an expiration timestamp and a path hash besides the PoC. These additional details are not essential for achieving path authorization and detectability but minimize the gap between the model and proposed protocol. We define ik_0^+ to consist of all authorized PoCs, since the attacker cannot extract them directly from the packets in this version. The second version is a further simplified variant of ICING compared to the one presented first, which omits the hop field in the key input of the MAC computation.

VIII. Extensions

We now describe three features of our formalization that we elided to simplify the presentation: additional authenticated fields, a stronger attacker model based on an oracle, and undirected authorization schemes.

A. Additional authenticated fields

To allow for more accurate modeling of protocols, our formalization includes additional per-hop and per-packet fields, which are included in auth_a and must thus be included in the authentication mechanisms defined by instances.

For instance, in SCION, packet headers include an expiration time that is fixed on the control plane and included in the MAC computation of the hop validation field. Consequently, paths have a limited lifespan and must be replaced on a regular basis. To model this, our formalization includes an *authenticated info field* of type $\mathbb{T}$ associated with each packet. In our SCION instance, this value is set to the expiration time.

ICING allows ASes to include arbitrary cryptographically protected forwarding information in an opaque string called *tag*. Our formalization defines abstract and concrete hop fields in an extensible way, such that additional data of type $\mathbb{T}$ can be added in instances. This can be used to model additional forwarding data that must be protected, such as ICING's *tag*.

These extra fields enable a more realistic modeling of existing protocols and make it more likely that future protocols can be modeled. For instance, if a new protocol includes flags indicating a path's priority, or introduces access control fields to allow only some user classes to use certain paths, no changes to the abstract and parametrized models are required to prove the protocol's security. Adding these fields requires some changes to our definitions but does not add significant complexity.

B. Strong attacker model

Legner et al. [17] propose a strong attacker model for EPIC, which reflects the fact that an adversary can with some effort brute-force correct *hvf* fields for individual *tok* values.

We model this attacker capability with an oracle. The concrete model is additionally parameterized by a predicate $\mathscr{O} : \mathbb{N} \to \mathbb{B}$. In the EPIC instance, the ik_0^+ set additionally contains all valid (but possibly unauthorized) *hvf* fields that are created over *tok* values such that $\mathscr{O}(tok)$ holds. While this strictly strengthens the attacker, her events must be restricted to rule out trivial attacks. We add the guard $\neg\mathscr{O}(tok(m))$ to the **dispatch-int**$_c$ and **dispatch-ext**$_c$ events to prevent the attacker from sending packets whose *hvf* fields are directly obtained from the oracle and thus trivially constitute an attack. We also add $\neg\mathscr{O}(tok(m))$ to the premises of COND 1 and COND 2.

The instance proofs are similar to the proof in the basic attacker model. However, they must additionally account for the attacker obtaining valid hop fields from the oracle.

C. Undirected authorization schemes

For brevity, we have focused on directed authorization schemes, where each AS only controls the authorization of the traversal of subsequent ASes (in forwarding direction) and the traversal of previous ASes is outside of its control. This setting allows the attacker to legitimately extend and change her own path on the control plane without consent by subsequent ASes, and hence requires the control plane assumptions ASM 3–ASM 6.

We have a separate parametrized model for the undirected authorization scheme, where the entire path must be approved by all on-path ASes. The control plane assumptions can be relaxed, and ASM 3–ASM 6 are replaced by the weaker assumption stating that an attacker can create authorized paths consisting entirely of compromised nodes, i.e., $hfs \in auth_a$ if $id(hf) \in \mathscr{N}_{attr}$ for all $hf \in hfs$. In this model, the cryptographic check parameter has the entire path (including the past path) as an argument: $\psi : \mathrm{HF}_c \times \mathrm{HF}_c^* \times \mathbb{N} \to \mathbb{B}$. The parameter *extract* retains its type, but returns the entire path (including past path) instead of just the future path. Since each hop validation field contains the entire path, COND 3 and COND 4 are replaced by an assumption that for a valid *hf*, i.e., $\psi(hf, hfs, tok)$, *extract* returns the entire path, i.e., $extract(hf, hfs, tok) = hfs$ holds.

In the undirected setting, the entire path is embedded in each *hvf* and cannot be modified unless it is completely under the attacker's control. Induction is neither required to show the refinement of the dispatch events in the concrete model nor to show the conditions in the ICING instance model, and proofs are significantly easier than in the directed setting.

IX. Discussion

A. Undirected vs. directed protocols

We briefly compare undirected and directed protocols. As shown in §VIII-C, undirected protocols achieve path authorization under weaker assumptions. While this sounds desirable, the required underlying changes to how paths are authorized on the control plane also have drawbacks. First, the beacons creating paths in undirected protocols must complete a round-trip: the first leg to discover the path and the second leg to authorize it. In contrast, directed protocols can achieve both in a single leg, where transmission is in the opposite direction of construction. Second, the control plane must mediate between conflicting path policies by ASes. If there is no path that satisfies the constraints by all on-path ASes, then no forwarding can occur. In directed protocols it is simpler to exclude this possibility, for instance by mandating that each AS disseminates at least one beacon from a given AS to each of its neighbors. In summary, there is a trade-off between these protocol classes that depends highly on the control plane and overall architecture.

B. Differences between models and real protocols

Our models abstract from real protocols in ways that are typical for protocol verification. Real protocols operate on bitstrings rather than typed terms, and they contain fields that we do not model, since they are unimportant to the path authorization security mechanism. Our models also overapproximate the behaviors allowed by the actual protocols, which is sound for safety properties. While our events accurately model the checks that border routers perform, the interaction with end hosts is simplified, as we do not differentiate end hosts within an AS and do not model intra-AS topologies.

Other than these abstractions, our instance models differ from the actual protocols in the following ways:

1) SCION: SCION is a complex architecture that includes many features that are necessary for Internet-scale operation. Beacons do not directly establish paths between any pair of ASes, but only between large ASes (e.g., Tier 1 providers) and their customer ASes and between the large-scale ASes themselves. Such partial paths (called *segments*) can then be reversed and concatenated by end hosts to connect distant ASes. While path authorization only holds for each segment individually, segments cannot be combined arbitrarily: there are rules that ensure that the economic interests of ASes are respected. Nevertheless these rules only provide *local* properties and not global properties (such as path authorization, which is expressed over entire paths). SCION also allows peering links between ASes, which similarly to segment combinations, are only authorized locally. For these reasons, we do not include segment combinations or peering links in our framework.

In contrast to our models, the SCION and EPIC protocols do not include the AS identifier (*id*) in hop fields. Nevertheless, in both protocols, a *hvf* uniquely identifies the AS for which the hop field is valid, since the key of the AS is used in the MAC computation. Hence, it would be possible to refine the instance model to a model in which hop fields do not explicitly contain the identifier. Alternatively, one could change the parametrized model to remove the AS identifier. This would require additional conditions.

2) EPIC: EPIC trims hop authenticators σ to a short length to reduce space overhead. We model the trimming of σ by a hash function. Similar to hashing, trimming makes it difficult to recover the original value. The trimming enables brute-force attacks, which we model by the oracle discussed in §VIII-B.

Formalization of framework	**LoC**
Infrastructure (Dolev–Yao, event system, etc.)	2125
Abstract model & network model	644
Concrete model (w/o Theorem 3)	759
Theorem 3 for directed setting	497
Theorem 3 for undirected setting	230
Total	**4255**
Formalization of instances	**LoC**
SCION	277
SCION (simplified)	272
EPIC Level 1 basic attacker	360
EPIC Level 1 strong attacker	397
EPIC Level 2 strong attacker	426
ICING (w/ extension)	301
ICING (as presented)	229
ICING (simplified)	239
Executability & assumption consistency	417
Total	**2918**

TABLE II: Overview of Isabelle/HOL formalization.

3) ICING: We leave out ICING's *proofs of provenance*. They are cryptographic authenticators used for path validation and are unrelated to path authorization. In the original specification, they are combined with the PoCs using XOR.

C. Formalization details and statistics

Our formalization in Isabelle/HOL closely follows the models and proofs described in this paper, modulo differences in notation and the extensions discussed in §VIII. Most of the proof burden is handled in the abstract and concrete parametrized models. In particular, the crux of the proof, Theorem 3, is part of the concrete model. A substantial portion of the instance models is boilerplate definitions and proofs that only vary slightly between the instances. Table II gives an overview of the different parts of our framework and the lines of Isabelle/HOL code associated with them.

D. Consistency of environment assumptions and executability

All instance models are still parametrized by the environment parameters, i.e., the underlying Internet topology defined by *target*, the set of authorized paths $auth_{\mathrm{a}}$ and the set of compromised nodes $\mathcal{N}_{attr} \subseteq \mathcal{N}$. We instantiate these parameters with the topology and authorized paths given in Fig. 2 and $\mathcal{N}_{attr} = \{F\}$. We discharge the associated assumptions ASM 1–ASM 6 from §VI-A in this example model to show their consistency. Furthermore, we show the executability of the instantiated event system for the EPIC level 1 protocol in the strong attacker model, showing that it is indeed possible to send a packet from a source to a destination, i.e., the model's events can be executed in the correct order.

E. Unverified data plane security properties

1) Source and packet authentication: These properties allow for the identification of a packet's origin and in some cases its header and content by ASes or the destination. The challenge in designing protocols that provide these properties is that they require keys between the source and the authenticating entity. Naïve solutions, such as using public key cryptography per packet, or distributing symmetric keys between each pair of entities are prohibitively inefficient. Hence, protocols often use dynamic key derivation techniques such as DRKey [35], [36]. With shared keys in place, authentication by a router or the destination can be easily implemented, modeled, and verified as a single-message two-party protocol. In contrast to network-wide properties like path authorization and detectability, the verification of source and packet authentication does not require any of the special features listed in the introduction. In particular, the network topology need not be modeled explicitly, and the set of authorized paths is irrelevant for this property. This makes it feasible to use automated tools such as Tamarin and ProVerif, in which protocol analysis is simpler than in Isabelle/HOL. For these reasons, we exclude source and packet authentication from our verification framework.

2) Path validation: This property proves to subsequent ASes and the destination that all previous hops on the path embedded in the packet were indeed traversed. While path validation is provided by some architectures [37], there are two reasons why it is less critical than the properties presented above. First, path validation only establishes a lower bound on the set of ASes that have been traversed and does not stop on-path attackers from sending copies of packets to ASes that are not part of the sender's intended path. Second, if there is at most one on-path attacker, then the much simpler packet authentication property is sufficient to imply path validation for the destination. By having the destination authenticate a packet (including the embedded path), any change of the embedded path by an on-path attacker will be detected by the destination even without a mechanism that provides path validation. Hence, path validation becomes only relevant for the destination if there are at least two colluding on-path attackers. For these reasons, we do not verify path validation in this work.

X. Related Work

As mentioned in the introduction, there exists relatively little work on the verification of path authorization for packet forwarding in path-aware internet architectures. We review those works here as well as other research on the verification of secure routing (i.e., path construction) protocols.

A. Data plane protocols for path-aware architectures

Over the past two decades, several other path-aware architectures have been developed [7]–[10]. Several of these use forwarding tables or other state on routers (instead of cryptographic authenticators) to achieve path authorization [9], [10], which does not fit into our framework. Others are not specified in sufficient detail to allow for formal verification [8] or only achieve local properties without considering full path authorization over multiple hops [7]. Finally, some data plane protocols [37], including OPT [35], focus only on source authentication and path validation, neither of which we verify.

A recent proposal [38], which we call Anapaya-SCION, achieves the same directed path authorization as SCION, but

its MAC does not include the next hop field's MAC. Instead, a single per-path field that combines all subsequent MACs with XOR is included in the MAC computation and updated at each hop. We do not verify this variant, as we currently do not support XOR or packet field updates by routers.

B. Verification of secure data plane protocols

Chen et al. [19] define SANDLog, a Prolog-style declarative language for specifying both data and control plane protocols. They also present an invariant proof rule for SANDLog programs and a verification condition generator, which targets Coq. They verify route authenticity of S-BGP and both route authenticity (on the control plane) and data path authenticity (on the data plane) of SCION. Hence, their coverage of SCION is more comprehensive than ours. However, their data plane property is weaker than our path authorization, as it only guarantees that each traversed hop appears on some authorized path, but does not relate successively traversed hops.

Zhang et al. [20] prove source authentication and path validation properties of the OPT packet forwarding protocols [35]. These properties differ from those that we study here. They use LS^2, a logic for reasoning about secure systems, in combination with axioms from Protocol Composition Logic (PCL) [39]. They directly embed their logic's axioms and prove the protocols' properties in Coq. As PCL does not have a formal semantics (cf. [40]), the soundness of their approach is questionable. In contrast, we use a foundational approach that only relies on the axioms of higher-order logic and on definitions.

C. Verification of secure routing protocols

Cortier et al. [41] propose a process calculus for modeling routing protocols, including a model of the network topology and a localized Dolev-Yao adversary. They propose two constraint-based NP decision procedures for analyzing routing protocols for a bounded number of sessions. The first one analyzes a protocol for any network topology, i.e., it decides whether there exists a network topology for which there is an attack on the protocol. The second procedure analyzes a protocol for a given network topology. They also define a logic to express properties such as loop-freedom and route validity. They analyze two ad-hoc routing protocols from the literature. This work is extended to protocols with recursive tests in [42].

Cortier, Degrieck, and Delaune [43] prove a reduction result showing that for proving path validity it is sufficient to consider just five topologies of four nodes. Path validity is similar to our ASM 2 but omitting interfaces. They then analyze two ad-hoc routing protocols using ProVerif.

D. Verification of network configurations

A different line of research is devoted to the verification of network configurations. Earlier work focused purely on the data plane [44]–[46] while more recent work also takes the control plane into account [47]–[51]. Verified properties include reachability, isolation, way-pointing, and loop freedom. These works are restricted to a setting with a fixed, concrete network topology and they do not consider security properties.

XI. Conclusion

The verification of future Internet architectures is a challenging problem, since (i) automated protocol verification tools lack the expressiveness to reason about arbitrary network topologies and (ii) the relevant protocols are likely to undergo changes before their eventual standardization and deployment. General guarantees for evolving protocols require general specifications and proofs that abstract from the idiosyncrasies of particular protocol instances. Our parameterized framework satisfies these requirements and substantially reduces the per-protocol specification and verification work compared to restarting verification from scratch for each protocol. For each protocol instance, one must only define the three parameters and prove the five conditions to establish path authorization and detectability. Our abstractions are general enough to cover a whole class of protocols proposed in the literature.

As future work, we plan to extend our framework with support for an XOR operator on terms, and en-route hop field updates to enable verification of Anapaya-SCION. In order to obtain guarantees that the router implementations behave according to the verified model, we will apply the Igloo methodology [52] to soundly link model verification with code verification.

An interesting research question is how additional properties, such as source authentication and path validation, can be integrated in a parametrized framework, even when not all instances support these properties. Such an integration would allow for the verification of EPIC level 3, full ICING, and OPT. Moreover, it would be interesting to investigate the existence of a reduction result in the vein of Cortier et al.'s work [43] for path-aware Internet architectures.

Acknowledgments

We thank Sofia Giampietro, Markus Legner, Adrian Perrig and the anonymous reviewers for their insights, careful reading of the manuscript and helpful suggestions.

References

[1] S. Kent, C. Lynn, and K. Seo, "Secure Border Gateway Protocol (S-BGP)," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 4, 2000.

[2] R. Bush, "Origin validation operation based on the Resource Public Key Infrastructure (RPKI)," RFC 7115, 2014.

[3] M. Lepinski and K. Sriram, "BGPsec Protocol Specification," RFC 8205, Sep. 2017. [Online]. Available: https://rfc-editor.org/rfc/rfc8205.txt

[4] D. Cooper, E. Heilman, K. Brogle, L. Reyzin, and S. Goldberg, "On the risk of misbehaving RPKI authorities," in *Proceedings of the ACM Workshop on Hot Topics in Networks (HotNets)*, 2013, pp. 1–7.

[5] Q. Li, Y.-C. Hu, and X. Zhang, "Even rockets cannot make pigs fly sustainably: Can BGP be secured with BGPsec?" in *Proceedings of the NDSS Workshop on Security of Emerging Networking Technologies (SENT)*. Internet Society, 2014.

[6] B. Rothenberger, D. E. Asoni, D. Barrera, and A. Perrig, "Internet kill switches demystified," in *Proceedings of the European Workshop on Systems Security (EuroSec)*, 2017.

[7] B. Raghavan and A. C. Snoeren, "A system for authenticated policy-compliant routing," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 4, 2004.

[8] B. Bhattacharjee, K. Calvert, J. Griffioen, N. l Spring, and J. Sterbenz, "Postmodern internetwork architecture," University of Kansas, Tech. Rep. ITTC Technical Report ITTC-FY2006-TR-45030-01, Feb. 2006.

[9] X. Yang, D. Clark, and A. W. Berger, "NIRA: A new inter-domain routing architecture," *IEEE/ACM Transactions on Networking*, vol. 15, no. 4, pp. 775–788, 2007.

[10] P. B. Godfrey, I. Ganichev, S. Shenker, and I. Stoica, "Pathlet routing," in *Proceedings of ACM SIGCOMM*, 2009.

[11] X. Zhang, H.-C. Hsiao, G. Hasker, H. Chan, A. Perrig, and D. Andersen, "SCION: Scalability, control, and isolation on next-generation networks," in *Proceedings of the IEEE Symposium on Security and Privacy*, 2011.

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

[13] A. Perrig, P. Szalachowski, R. M. Reischuk, and L. Chuat, *SCION: A Secure Internet Architecture*. Springer, 2017. [Online]. Available: https://doi.org/10.1007/978-3-319-67080-5

[14] S. Meier, B. Schmidt, C. Cremers, and D. Basin, "The TAMARIN prover for the symbolic analysis of security protocols," in *Computer Aided Verification (CAV)*. Springer, 2013, pp. 696–701.

[15] B. Blanchet, "An efficient cryptographic protocol verifier based on prolog rules," in *Proceedings. 14th IEEE Computer Security Foundations Workshop, 2001*, pp. 82–96.

[16] T. Nipkow, L. C. Paulson, and M. Wenzel, *Isabelle/HOL - A Proof Assistant for Higher-Order Logic*. Springer, 2002, vol. 2283. [Online]. Available: https://doi.org/10.1007/3-540-45949-9

[17] M. Legner, T. Klenze, M. Wyss, C. Sprenger, and A. Perrig, "EPIC: Every packet is checked in the data plane of a path-aware Internet," in *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, 2020, pp. 541–558. [Online]. Available: https://www.usenix.org/conference/usenixsecurity20/presentation/legner

[18] J. Naous, M. Walfish, A. Nicolosi, D. Mazières, M. Miller, and A. Seehra, "Verifying and enforcing network paths with ICING," in *Proceedings of the Seventh COnference on Emerging Networking EXperiments and Technologies (CoNEXT)*. ACM, 2011. [Online]. Available: https://doi.org/10.1145/2079296.2079326

[19] C. Chen, L. Jia, H. Xu, C. Luo, W. Zhou, and B. T. Loo, "A program logic for verifying secure routing protocols," *Logical Methods in Computer Science*, vol. Volume 11, Issue 4, 2015. [Online]. Available: https://lmcs.episciences.org/1620

[20] F. Zhang, L. Jia, C. Basescu, T. H.-J. Kim, Y.-C. Hu, and A. Perrig, "Mechanized network origin and path authenticity proofs," in *Proceedings of the 2014 ACM Conference on Computer and Communications Security*.

[21] E. Katz-Bassett, C. Scott, D. R. Choffnes, Í. Cunha, V. Valancius, N. Feamster, H. V. Madhyastha, T. E. Anderson, and A. Krishnamurthy, "LIFEGUARD: practical repair of persistent route failures," in *SIGCOMM 2012*. ACM, pp. 395–406.

[22] T. G. Griffin and G. Wilfong, "An analysis of BGP convergence properties," ser. SIGCOMM '99. New York, NY, USA: ACM, 1999, p. 277–288.

[23] T. Bates, P. Smith, and G. Huston, "CIDR report," https://www.cidr-report.org/as2.0/, 2020.

[24] H. Ballani, P. Francis, and X. Zhang, "A study of prefix hijacking and interception in the internet," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 4, pp. 265–276, 2007.

[25] R. White, "Securing BGP through secure origin BGP (soBGP)," *Business Communications Review*, vol. 33, no. 5, pp. 47–47, 2003.

[26] T. Wan, E. Kranakis, and P. C. van Oorschot, "Pretty secure BGP, psBGP," in *Proceedings of the Network and Distributed System Security Symposium, NDSS 2005, San Diego, California, USA*, 2005. [Online]. Available: https://www.ndss-symposium.org/ndss2005/pretty-secure-bgp-psbgp/

[27] J. Karlin, S. Forrest, and J. Rexford, "Pretty good BGP: Improving BGP by cautiously adopting routes," in *Proceedings of the IEEE International Conference on Network Protocols*. IEEE, 2006.

[28] Y. Gilad, A. Cohen, A. Herzberg, M. Schapira, and H. Shulman, "Are we there yet? On RPKI's deployment and security." in *NDSS*, 2017.

[29] NIST, "RPKI monitor," https://rpki-monitor.antd.nist.gov, 2020.

[30] L. Gao, "On inferring autonomous system relationships in the Internet," *IEEE/ACM Trans. on Networking*, vol. 9, no. 6, pp. 733–745, 2001.

[31] L. Gao and J. Rexford, "Stable internet routing without global coordination," *IEEE/ACM Trans. Netw.*, vol. 9, no. 6, p. 681–692, Dec. 2001.

[32] N. A. Lynch and F. W. Vaandrager, "Forward and backward simulations: I. untimed systems," *Inf. Comput.*, vol. 121, no. 2, 1995.

[33] C. Ballarin, "Locales: A module system for mathematical theories," *J. Autom. Reason.*, vol. 52, no. 2, pp. 123–153, 2014. [Online]. Available: https://doi.org/10.1007/s10817-013-9284-7

[34] L. Paulson, "The inductive approach to verifying cryptographic protocols," *J. Computer Security*, vol. 6, 1998. [Online]. Available: http://www.cl.cam.ac.uk/users/lcp/papers/Auth/jcs.pdf

[35] T. H.-J. Kim, C. Basescu, L. Jia, S. B. Lee, Y.-C. Hu, and A. Perrig, "Lightweight source authentication and path validation," in *Proceedings of the 2014 ACM Conference on SIGCOMM*, 2014, p. 271–282. [Online]. Available: https://doi.org/10.1145/2619239.2626323

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

[37] K. Bu, A. Laird, Y. Yang, L. Cheng, J. Luo, Y. Li, and K. Ren, "Unveiling the mystery of Internet packet forwarding: A survey of network path validation," *ACM Comput. Surv.*, vol. 53, no. 5, Sep. 2020. [Online]. Available: https://doi.org/10.1145/3409796

[38] Anapaya Systems, "SCION header specification," https://scion.docs.anapaya.net/en/latest/protocols/scion-header.html, 2020.

[39] A. Datta, A. Derek, J. C. Mitchell, and A. Roy, "Protocol composition logic (PCL)," *Electr. Notes Theor. Comput. Sci.*, vol. 172, pp. 311–358, 2007.

[40] C. J. F. Cremers, "On the protocol composition logic PCL," in *Proceedings of the 2008 ACM Symposium on Information, Computer and Communications Security, ASIACCS 2008, Tokyo, Japan, March 18-20, 2008*, M. Abe and V. D. Gligor, Eds. ACM, 2008, pp. 66–76. [Online]. Available: https://doi.org/10.1145/1368310.1368324

[41] M. Arnaud, V. Cortier, and S. Delaune, "Modeling and verifying ad hoc routing protocols," *Inf. Comput.*, vol. 238, pp. 30–67, 2014. [Online]. Available: http://dx.doi.org/10.1016/j.ic.2014.07.004

[42] ——, "Deciding security for protocols with recursive tests," in *Automated Deduction – CADE-23*. Springer, 2011, pp. 49–63.

[43] V. Cortier, J. Degrieck, and S. Delaune, "Analysing routing protocols: Four nodes topologies are sufficient," in *Principles of Security and Trust*. Springer, 2012, pp. 30–50.

[44] P. Kazemian, G. Varghese, and N. McKeown, "Header space analysis: Static checking for networks," in *Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2012*, 2012, pp. 113–126.

[45] A. Khurshid, X. Zou, W. Zhou, M. Caesar, and P. B. Godfrey, "Veriflow: Verifying network-wide invariants in real time," in *NSDI 2013*, pp. 15–27.

[46] D. Kozen, "NetKAT – A formal system for the verification of networks," in *Proc. of Programming Languages and Systems - 12th Asian Symposium, APLAS 2014*, ser. Lecture Notes in Computer Science, vol. 8858. Springer, 2014, pp. 1–18. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-12736-1_1

[47] A. Gember-Jacobson, R. Viswanathan, A. Akella, and R. Mahajan, "Fast control plane analysis using an abstract representation," in *Proceedings of the ACM SIGCOMM*. ACM, 2016, pp. 300–313. [Online]. Available: https://doi.org/10.1145/2934872.2934876

[48] S. K. Fayaz, T. Sharma, A. Fogel, R. Mahajan, T. D. Millstein, V. Sekar, and G. Varghese, "Efficient network reachability analysis using a succinct control plane representation," in *12th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2016*. USENIX Association, 2016, pp. 217–232. [Online]. Available: https://www.usenix.org/conference/osdi16/technical-sessions/presentation/fayaz

[49] K. Weitz, D. Woos, E. Torlak, M. D. Ernst, A. Krishnamurthy, and Z. Tatlock, "Scalable verification of border gateway protocol configurations with an SMT solver," in *Proc. ACM Program. Lang.*, ser. OOPSLA 2016, New York, NY, USA, 2016, p. 765–780. [Online]. Available: https://doi.org/10.1145/2983990.2984012

[50] R. Beckett, A. Gupta, R. Mahajan, and D. Walker, "A general approach to network configuration verification," in *Proceedings of the ACM SIGCOMM*. ACM, 2017, pp. 155–168. [Online]. Available: https://doi.org/10.1145/3098822.3098834

[51] ——, "Abstract interpretation of distributed network control planes," *Proc. ACM Program. Lang.*, vol. 4, no. POPL, pp. 42:1–42:27, 2020. [Online]. Available: https://doi.org/10.1145/3371110

[52] C. Sprenger, T. Klenze, M. Eilers, F. A. Wolf, P. Müller, M. Clochard, and D. Basin, "Igloo: Soundly linking compositional refinement and separation logic for distributed system verification," *Proc. ACM Program. Lang.*, vol. 4, no. OOPSLA, Nov. 2020. [Online]. Available: https://doi.org/10.1145/3428220