Verified Secure Routing

VerifiedSCION

David Basin
ETH Zurich

ICECCS
November 2017

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

VerifiedSCION Team Members

Verification Team

Information Security
David Basin
Tobias Klenze
Ralf Sasse
Christoph Sprenger
Thilo Weghorn

Programming Methodology
Marco Eilers
Peter Müller

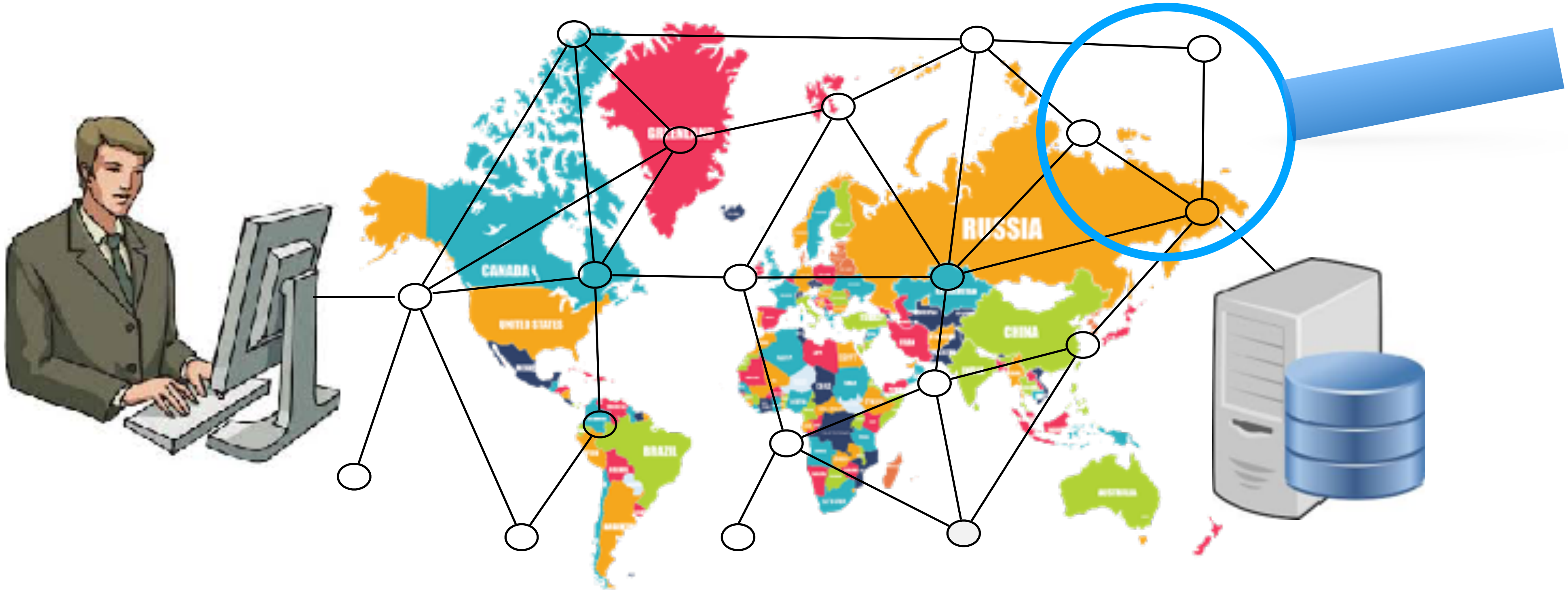
Network Security
Samuel Hitz
Adrian Perrig

Scion Design & Development Team

Motivation and Context

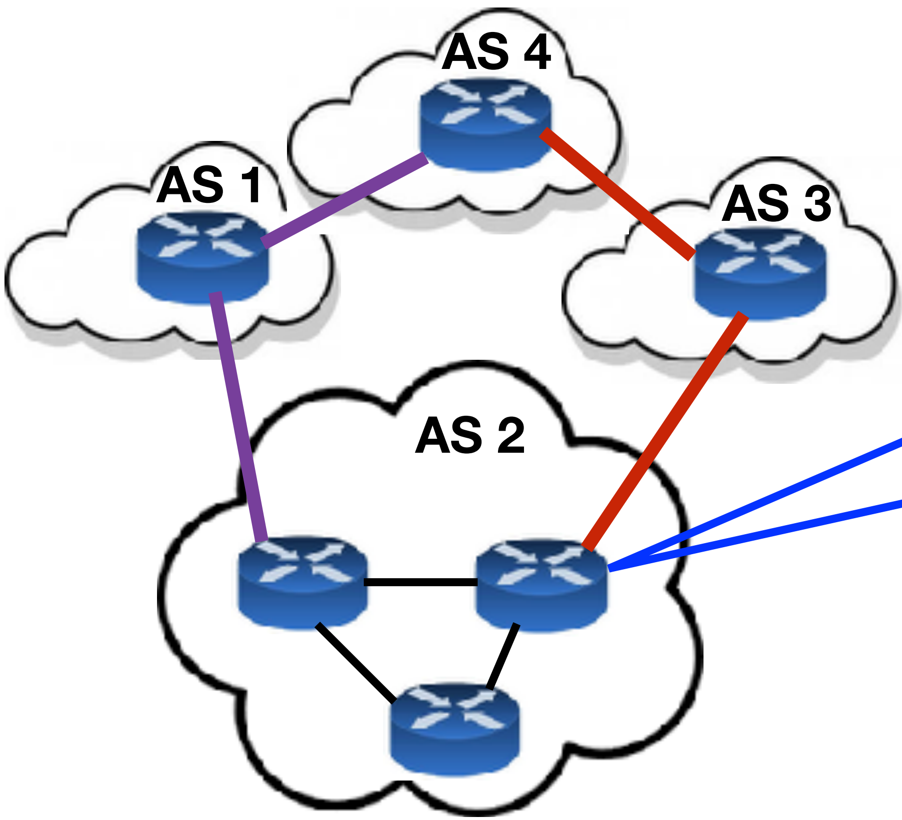
Routing problems with the status quo (inter-AS routing)

Routing Between Autonomous Systems



- The Internet is a network of **Autonomous Systems (ASes).**
- Each AS is itself a network of **routers** run by an institution (e.g., Telco, ISP, company, or university).
- There are 50,000+ ASes in the world.

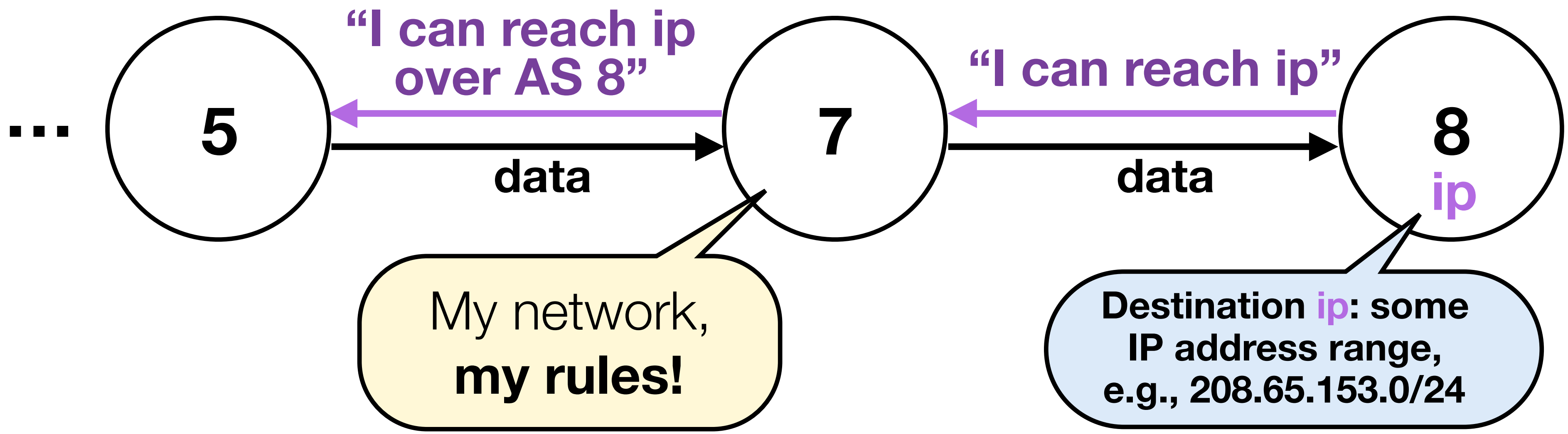
Autonomous systems and routers



1. finds paths **between ASes** (and within ASes)
2. forwards data along paths

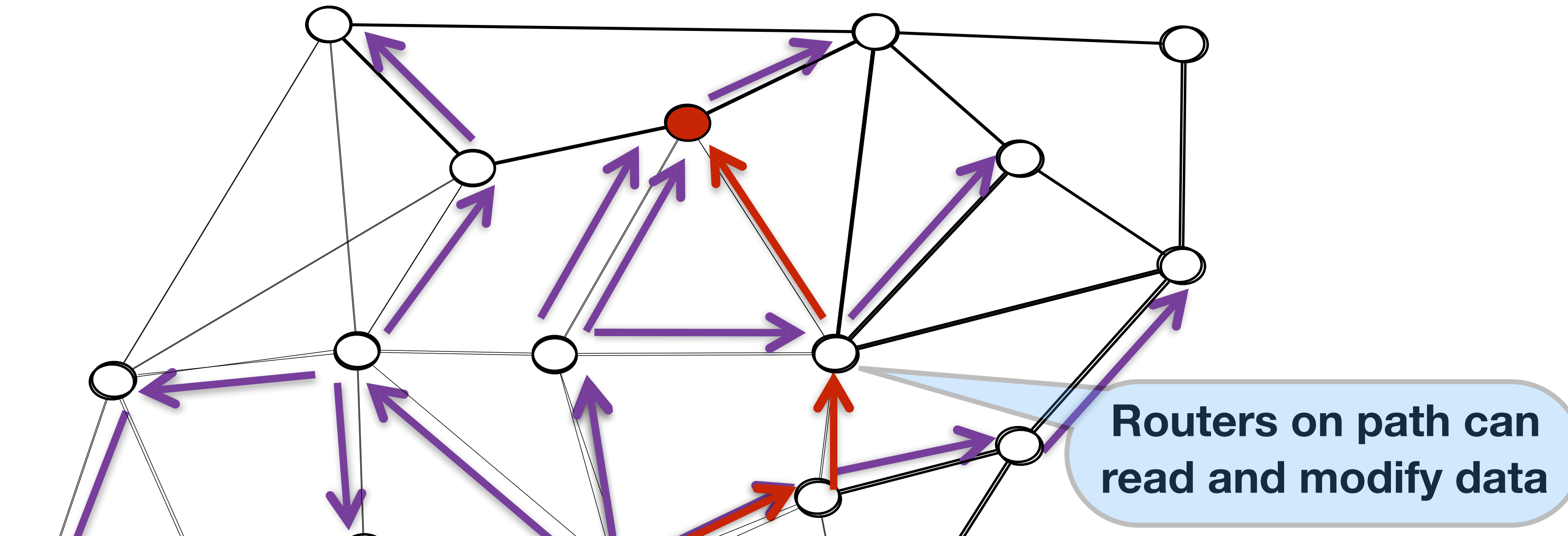
- Multiple paths between ASes: 2,1,4 and 2,3,4
- Computed in background by ***Border Gateway Protocol*** (BGP) and just one will be selected and used to configure routers

Border Gateway Protocol



- ASes announce **paths to destination address ranges**.
  - One path per destination used to configure routers.
  - Data flows back in the **opposite** direction.
- **Policies**
  - Decide on what is accepted, rejected, or propagated.
  - Any AS can announce any address range it wants!

Who controls the Internet?



NSA data storage center Utah

- Control over paths is completely distributed
  - Border Gateway Protocol (BGP): all nodes flood path announcements

- No inbound traffic control

Who controls Internet paths?

Three concrete examples

208.65.153.0/22

208.65.153.0/24

Pakistan Telecommunication Authority

Pakistan DoS against Youtube (2 hours, 2008)

Strange snafu hijacks UK nuke maker's traffic, routes it through Ukraine

Lockheed, banks, and helicopter designer also affected by border gateway mishap.

by Dan Goodin - Mar 13, 2015 5:13pm CET

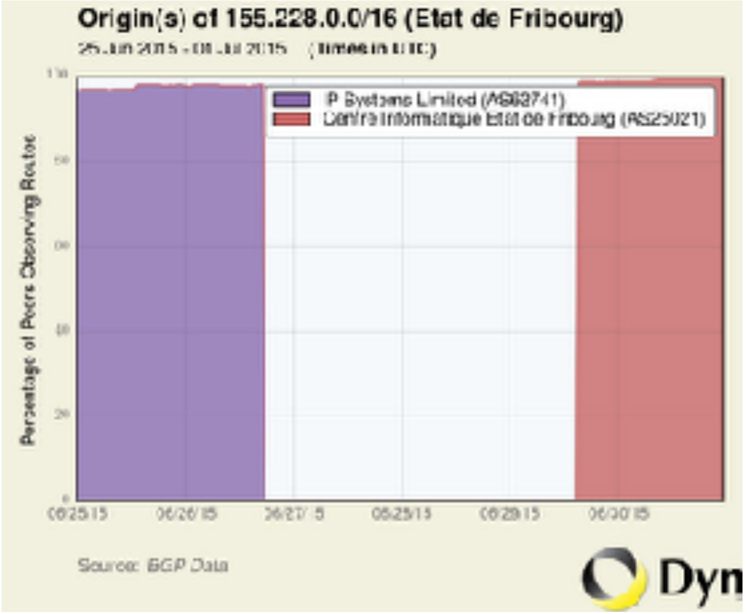
Share Tweet Email

Redirected traffic to UK Atomic Weapons Establishment

Internet traffic for 167 important British Telecom customers—including a UK defense contractor that helps deliver the country's nuclear warhead program—were mysteriously diverted to servers in Ukraine before being passed along to their final destination.

The snafu may have allowed adversaries to eavesdrop on or tamper with communications sent and received by the UK's Atomic Weapons Establishment, one of the affected British Telecom customers. Other organizations with hijacked traffic include defense contractor Lockheed Martin, Toronto Dominion Bank, Anglo-Italian helicopter company AgustaWestland, and the UK

Ukraine ISP hijacks UK routes including UK Atomic Weapons



Fribourg's government address space stolen for 3 days by SPAMers

Scion

Routing as it should be

SCION Project
Secure Future Internet Architecture

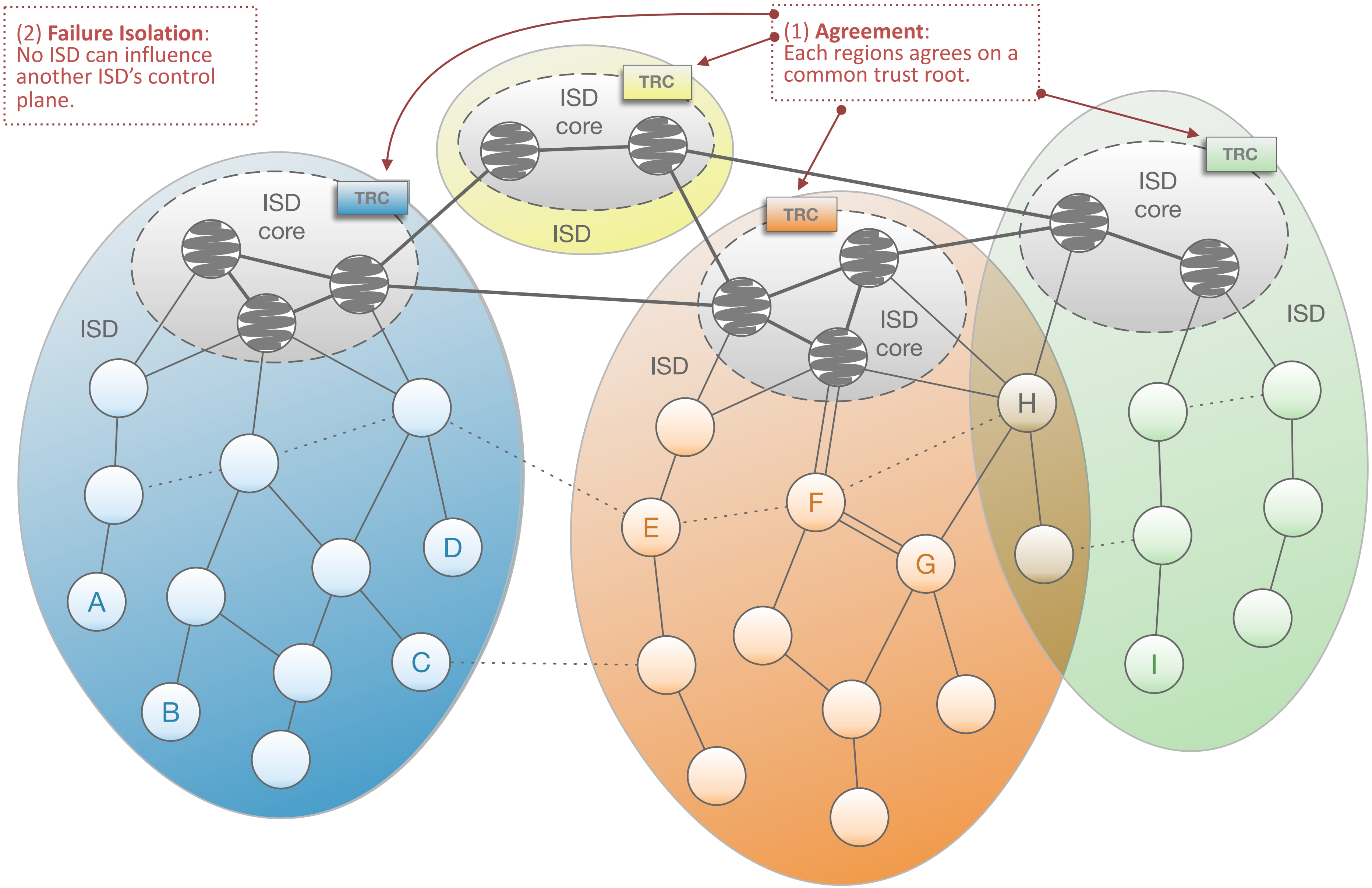
- Design & Implementation, 75+ man years
- Design of routing / forwarding protocols, support ecosystem, and numerous extensions
- Clean slate, yet compatible with existing Internet
- Not just a research prototype, growing deployment: 26 ASes on 3 continents

- See **<u>www.scion-architecture.net</u>** and related publications CACM 2017, IEEE S&P 2011, CCS 2015, NDSS 2016, S&P 2016

SCION Overview

- **Isolation Domains** (ISD)
- **Control Plane**: routing
  - Path exploration
  - Path registration
  - Path resolution
- **Data Plane**: packet forwarding

SCION Isolation Domain (ISD)

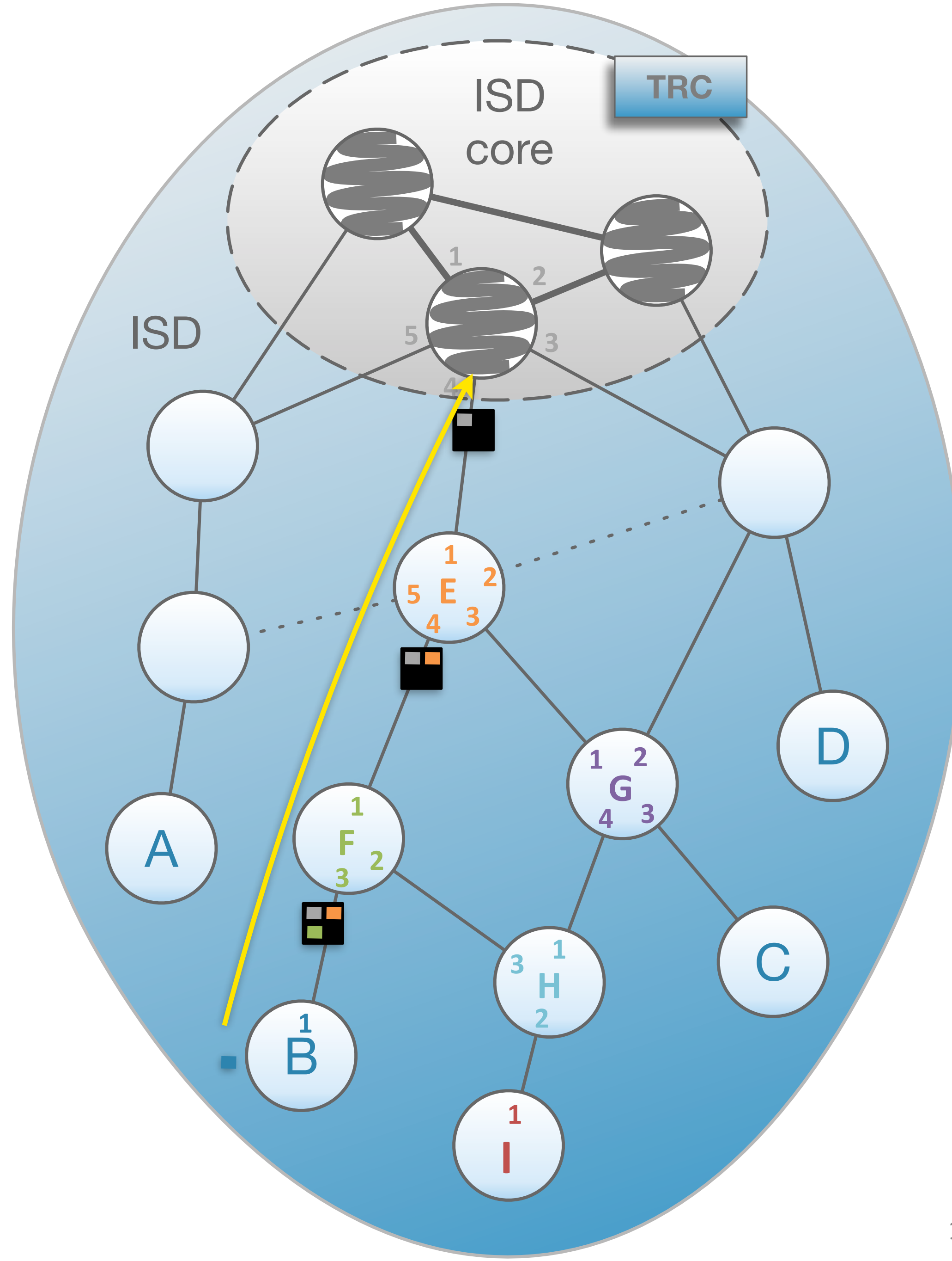


SCION Routing (Control Plane)

Routing Phases:
(1) Path Exploration
(2) Path Registration
(3) Path Resolution

Beaconing

PCB
Core: Out: 4
AS E: In: 1, Out: 4
AS F: In: 1, Out: 3
AS B: In: 1



- Path Construction Beacons (PCB) are Sequence of signed Hop Fields
- Hop Fields (HF) carry the routing information for one AS

AS X:
In: y, Out: z

SCION Routing (Control Plane)

Routing Phases:

(1) Path Exploration

(2) Path Registration

(3) Path Resolution

Beaconing

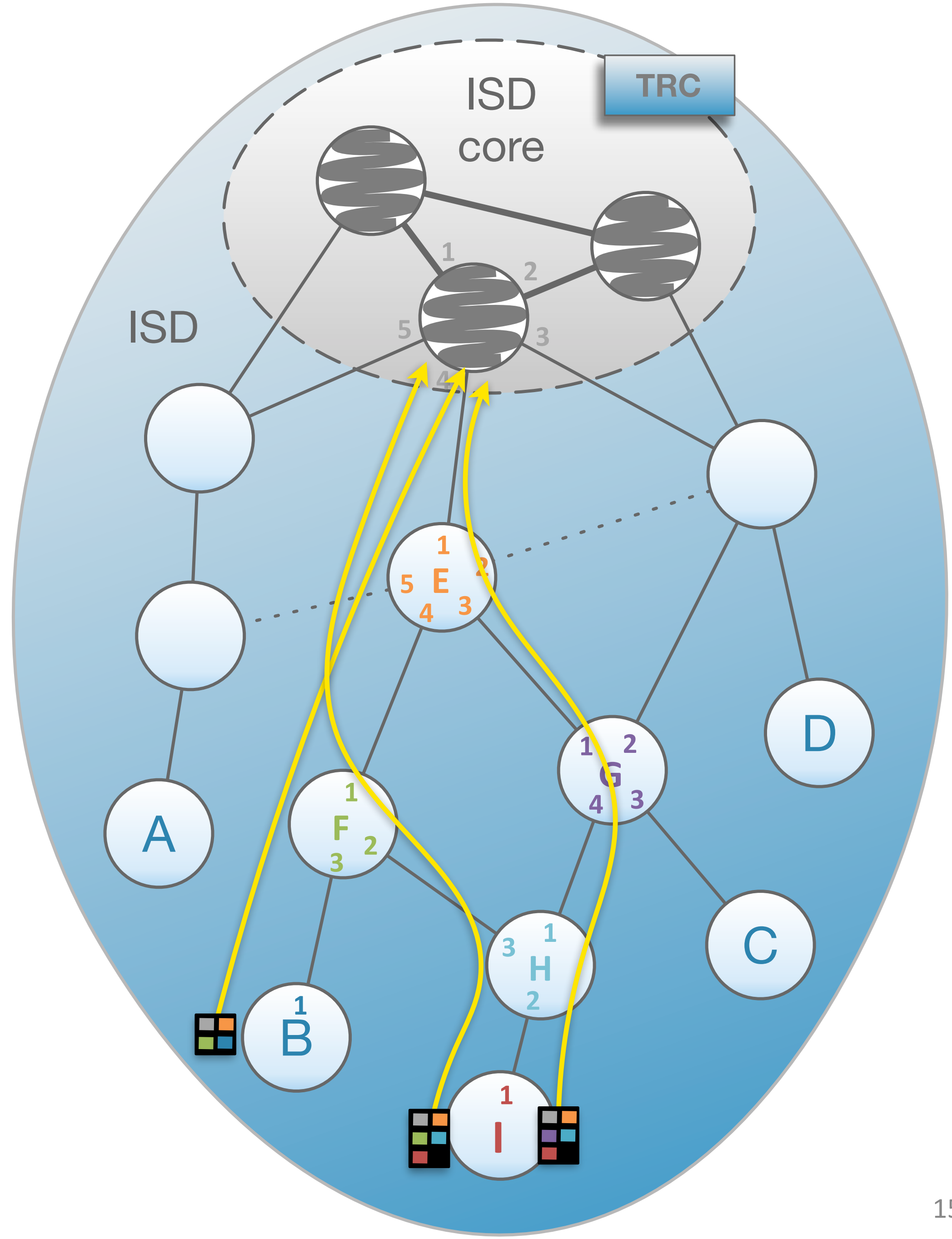
PCB

Core: Out: 4

AS E:
In: 1, Out: 4

AS F:
In: 1, Out: 3

AS B: In: 1



- Path Construction Beacons (PCB) are Sequence of signed Hop Fields
- Hop Fields (HF) carry the routing information for one AS

AS X:
In: y, Out: z

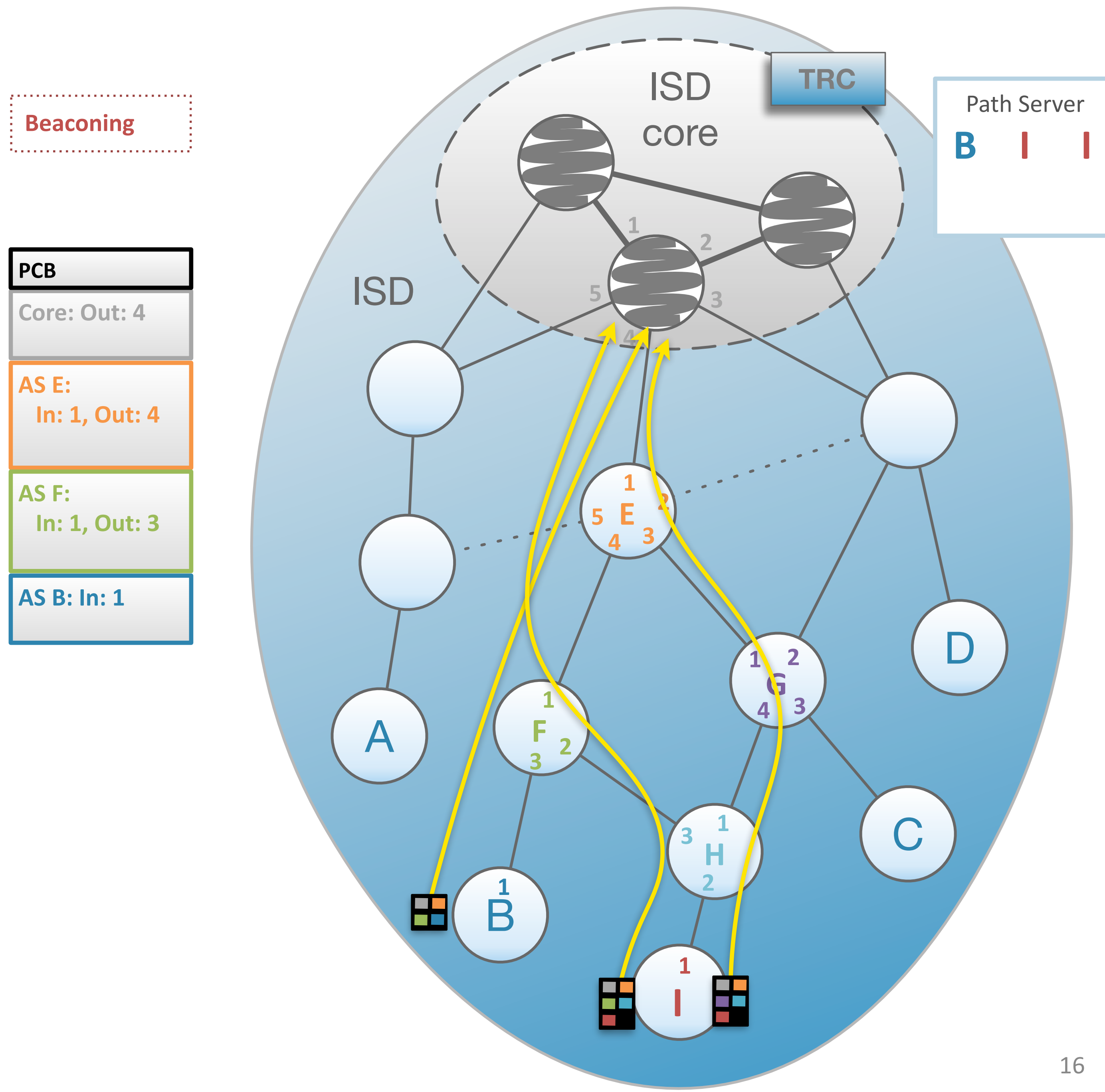
SCION Routing (Control Plane)

Routing Phases:

(1) Path Exploration

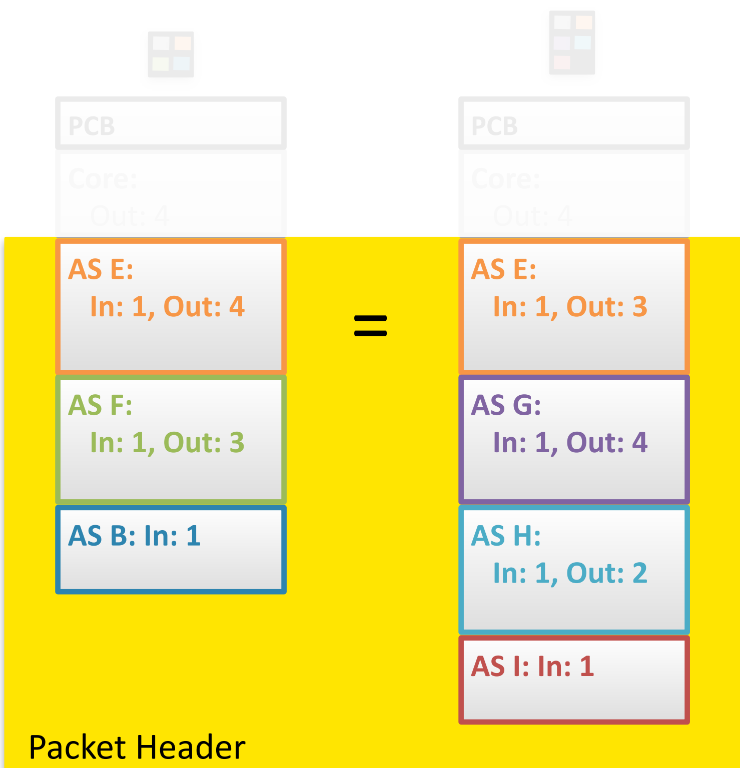
(2) Path Registration

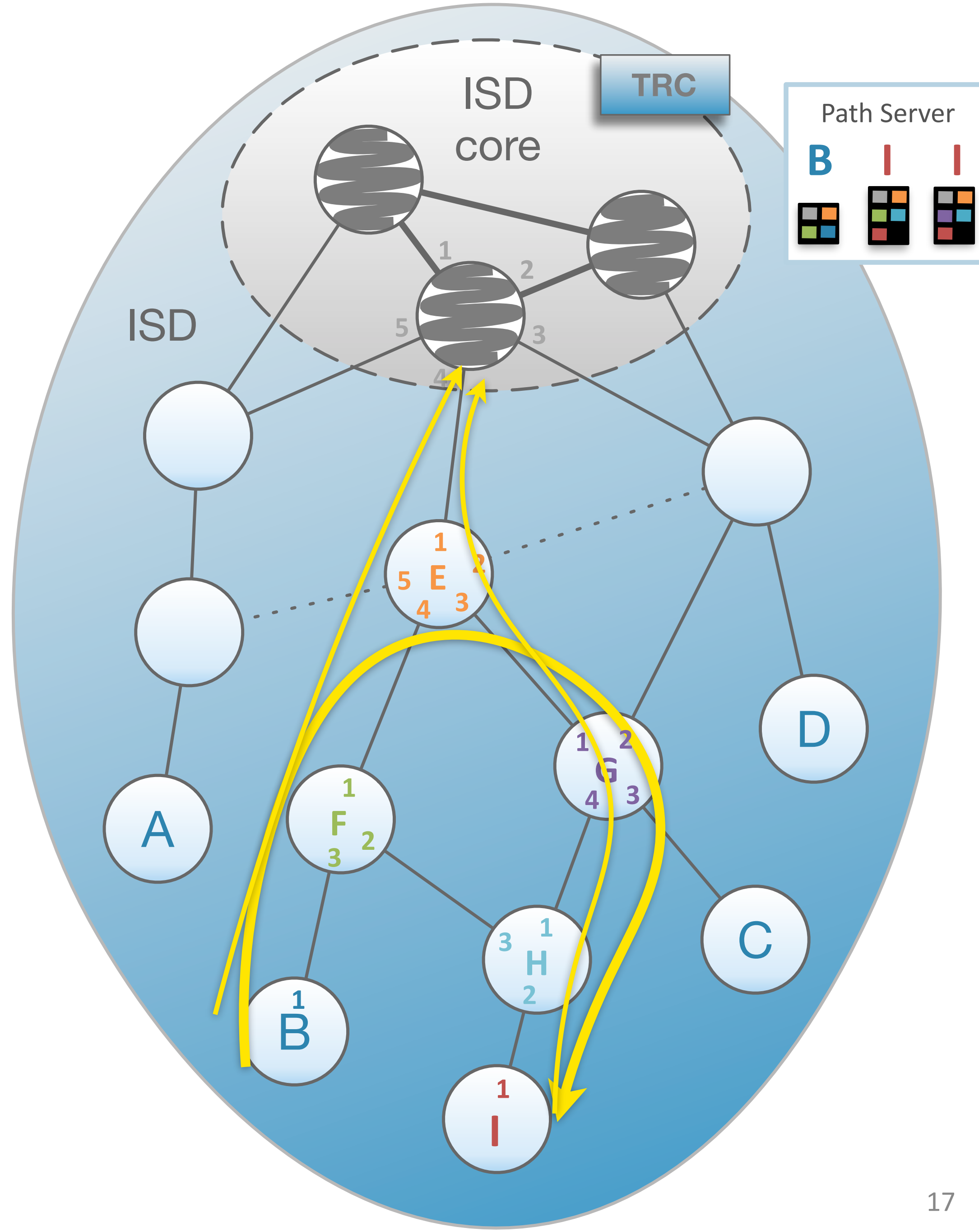
(3) Path Resolution



SCION Routing (Control Plane)

Routing Phases:
(1) Path Exploration
(2) Path Registration
(3) Path Resolution





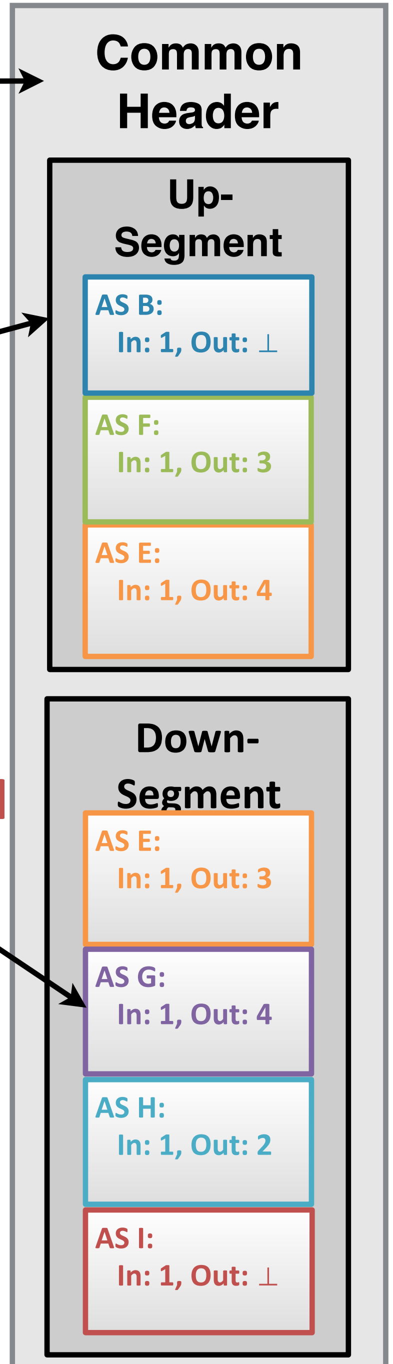
SCION Forwarding (Data Plane)

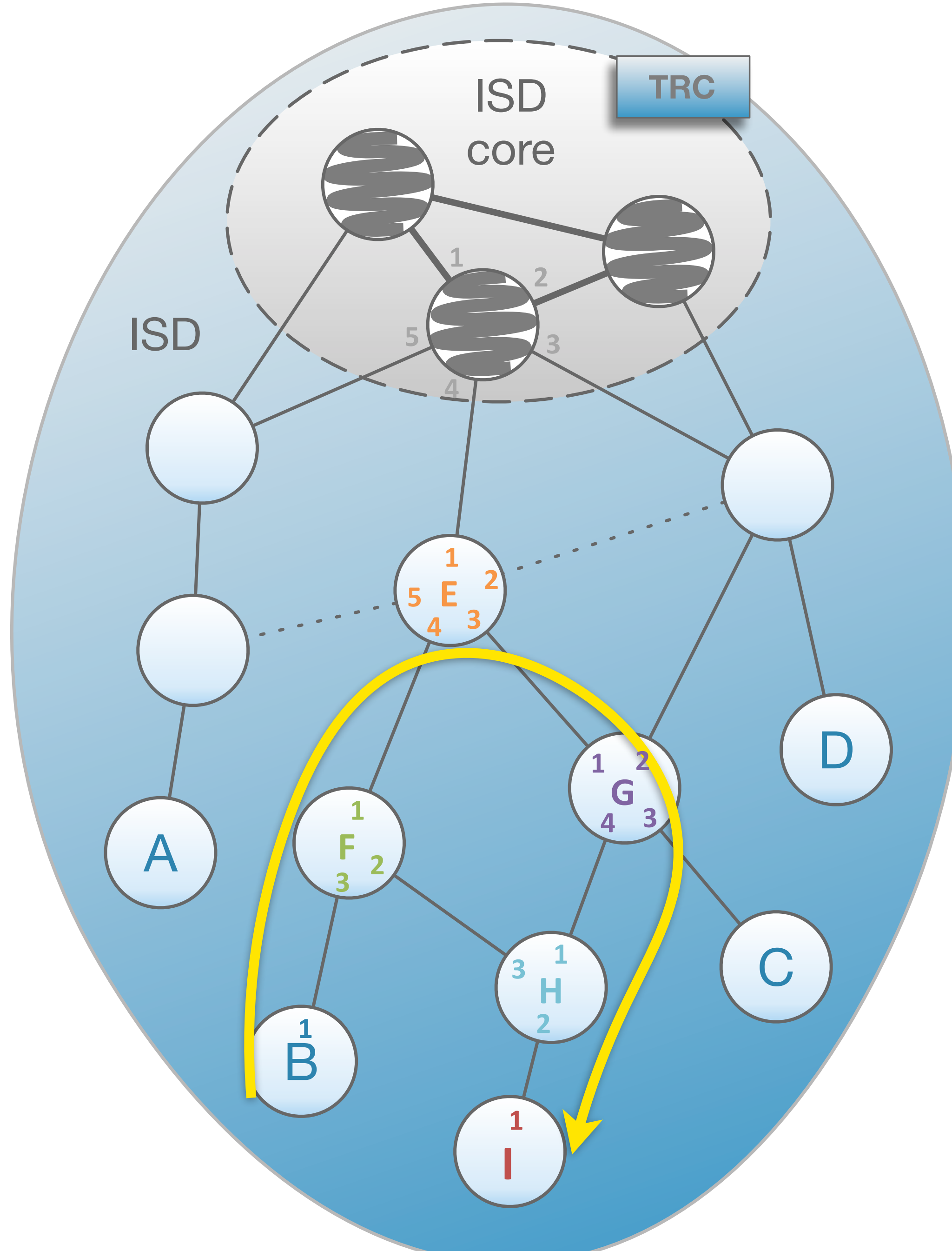
Packet header

Forwarding along:
- **Up-Segment**
- **Core-Segment**
- **Down-Segment**

Segments are sequences of Hop Field (HFs).

Hop Field contain routing information of one AS.





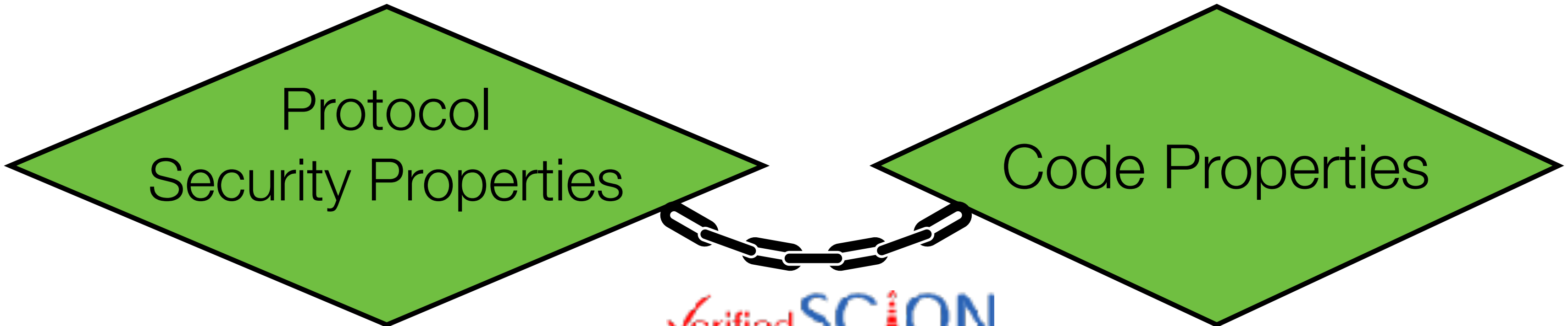
Verification

High-level, omitting formal details

Can We Verify Scion?

- Control and data plane guarantees
- Functional **correctness** of actual code
  - Suitable for high-assurance business cases
  - Ensures that routers are backdoor-free
- **Scion routers are simple and stateless**
  - This is the key to their (feasible) verification
  - Not possible for current Internet with highly complex routers and giant code bases of millions of lines

Correctness and Security
SCION Approach



VerifiedSCION

Verification of **protocols models** at the **network level**

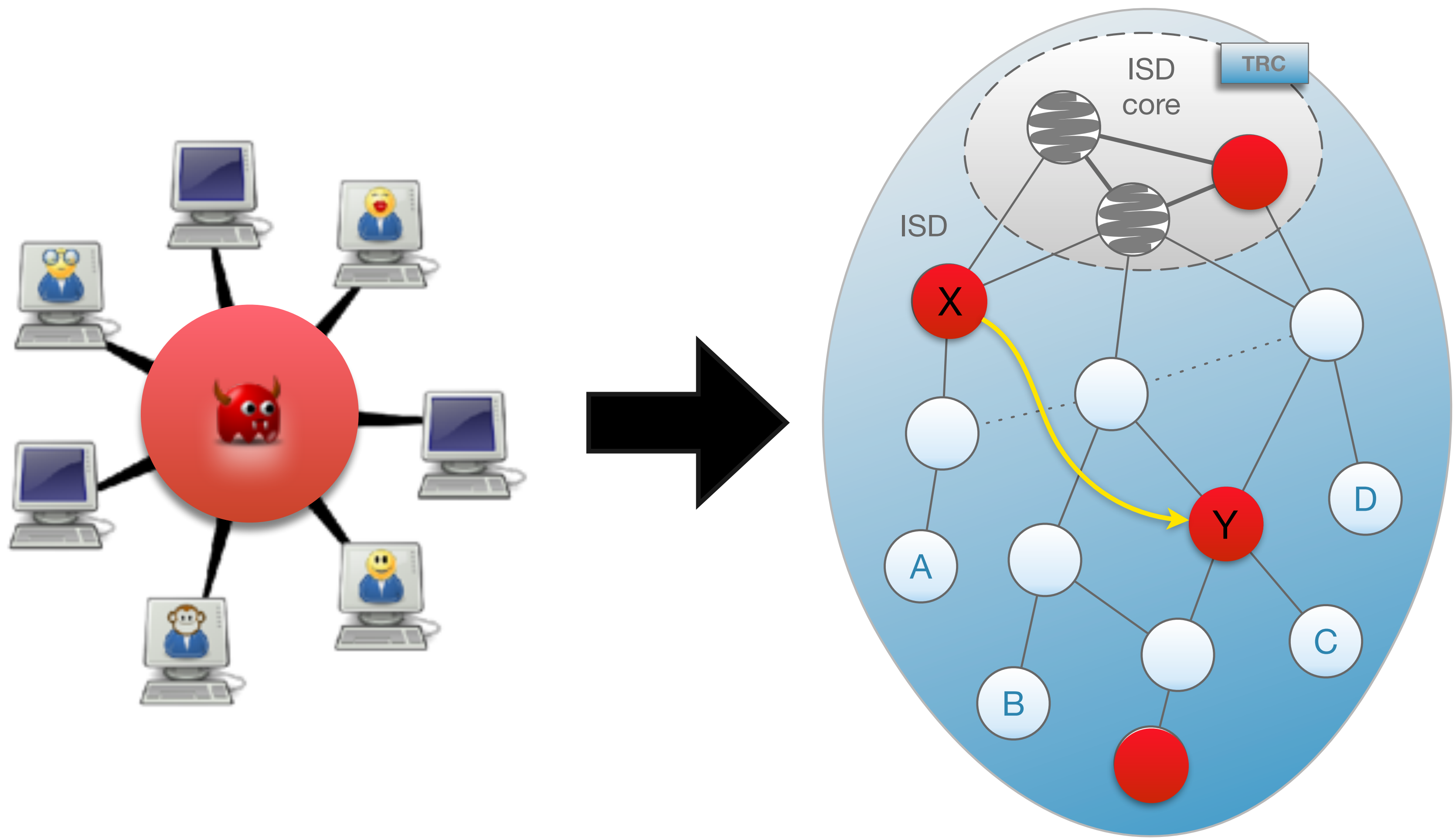
- **Assuming** that each SCION component **behaves as specified**
- **Technique:** stepwise refinement, preserves invariants, using Isabelle/HOL.

Verification of the **components** at the **code level**

- **Guaranteeing** that each SCION component **behaves as specified.**
- **Technique:** Hoare-style pre-/post-condition reasoning, Viper with Python front-end.

Concrete Attacker Model

We use a **localized, colluding** Dolev-Yao attacker model



Attacker controls the
entire network

Attacker controls a
subset of ASes

SCION Protocol Security Properties

Control plane properties

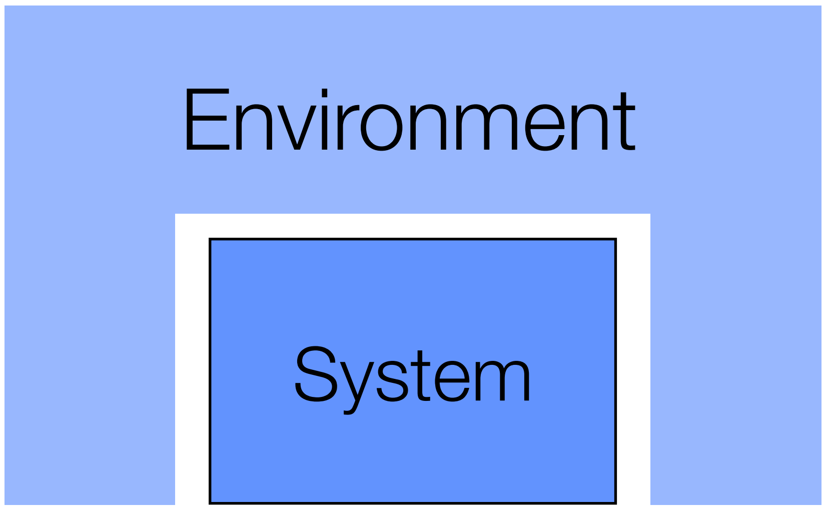
- **Beacon validity:** Sequence of ASes in a beacon corresponds to a path in the network (modulo wormholes).

Data plane properties

- **Path authorization:** Packets only forwarded along previously authorized paths.
- **Weak detectability:** An active attacker cannot hide his presence on the path.

Our initial focus is on data plane / router code verification.

System & Environment



Attacker

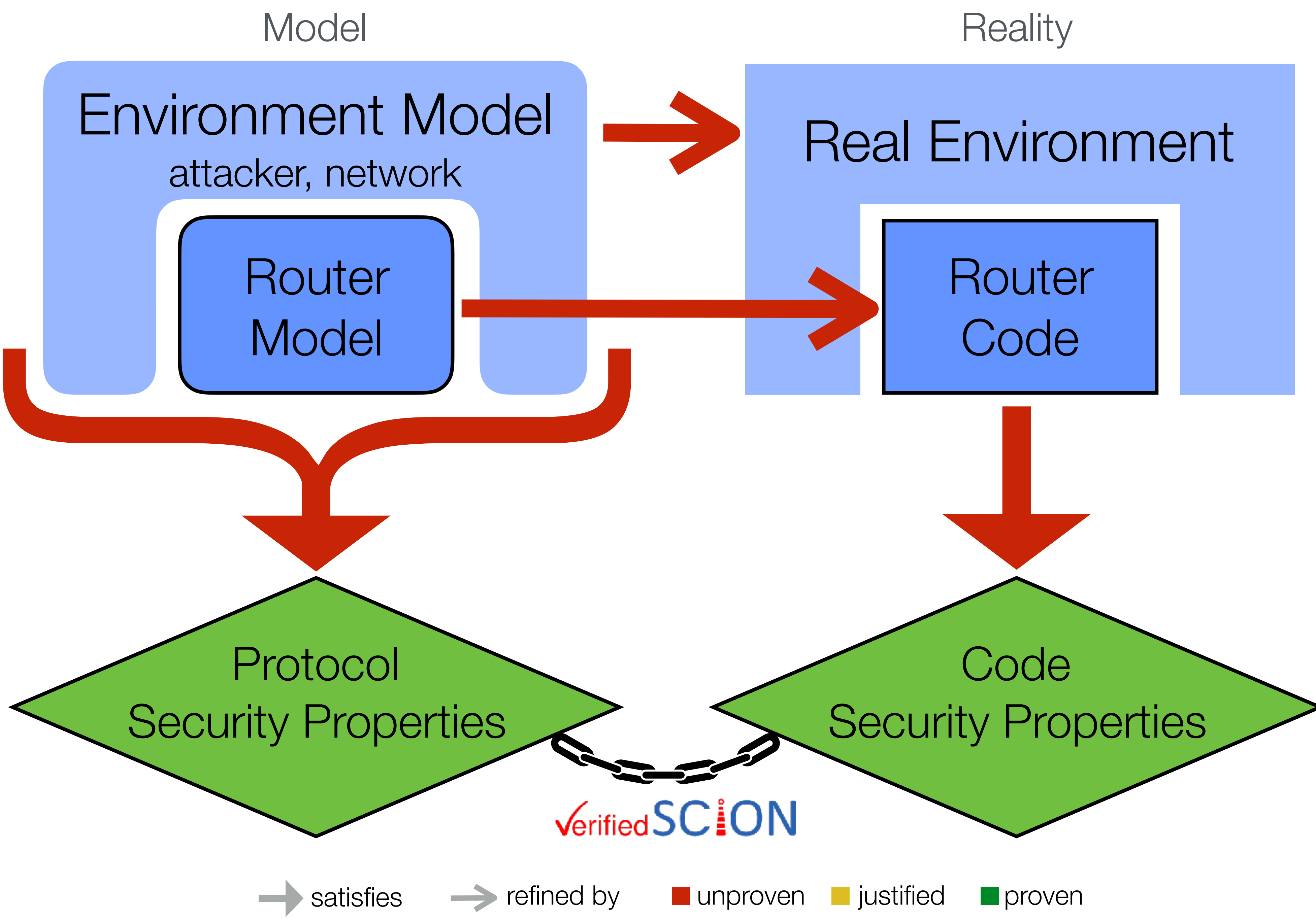
Network

End hosts

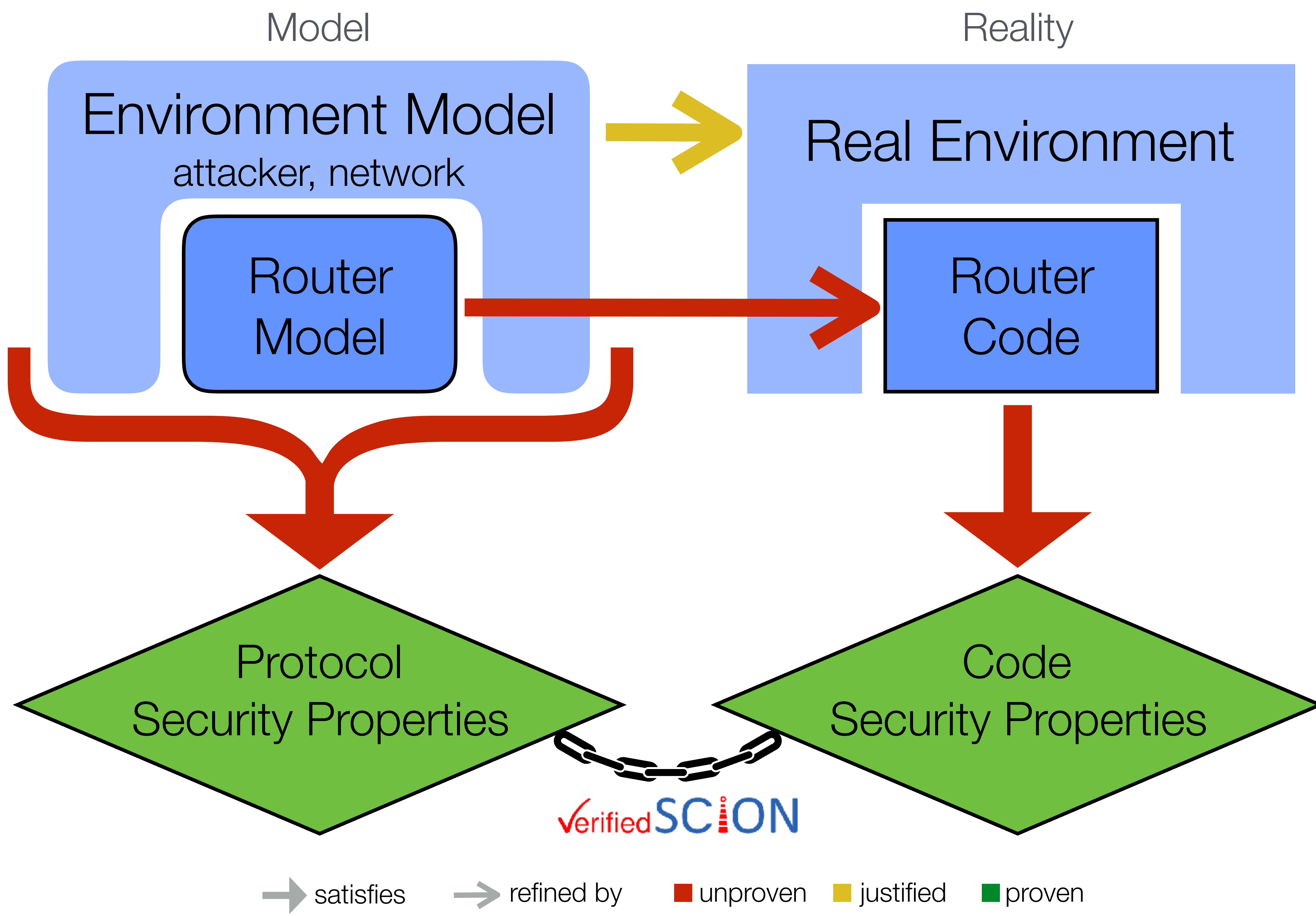
OS & Libraries

Border Router

SCION Router Verification Overview

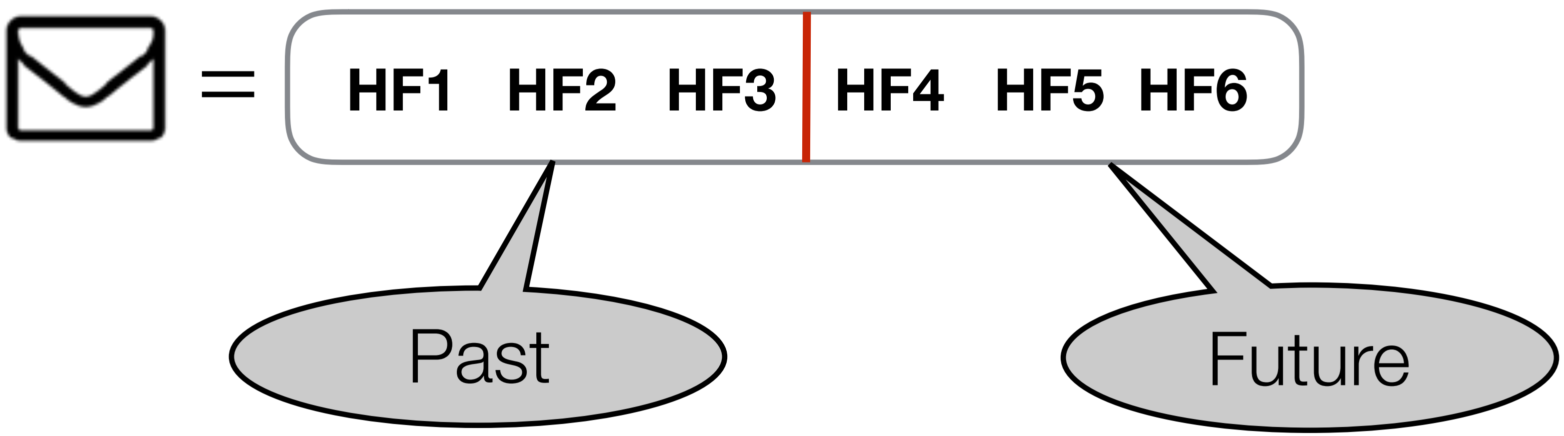


SCION Router Verification Overview



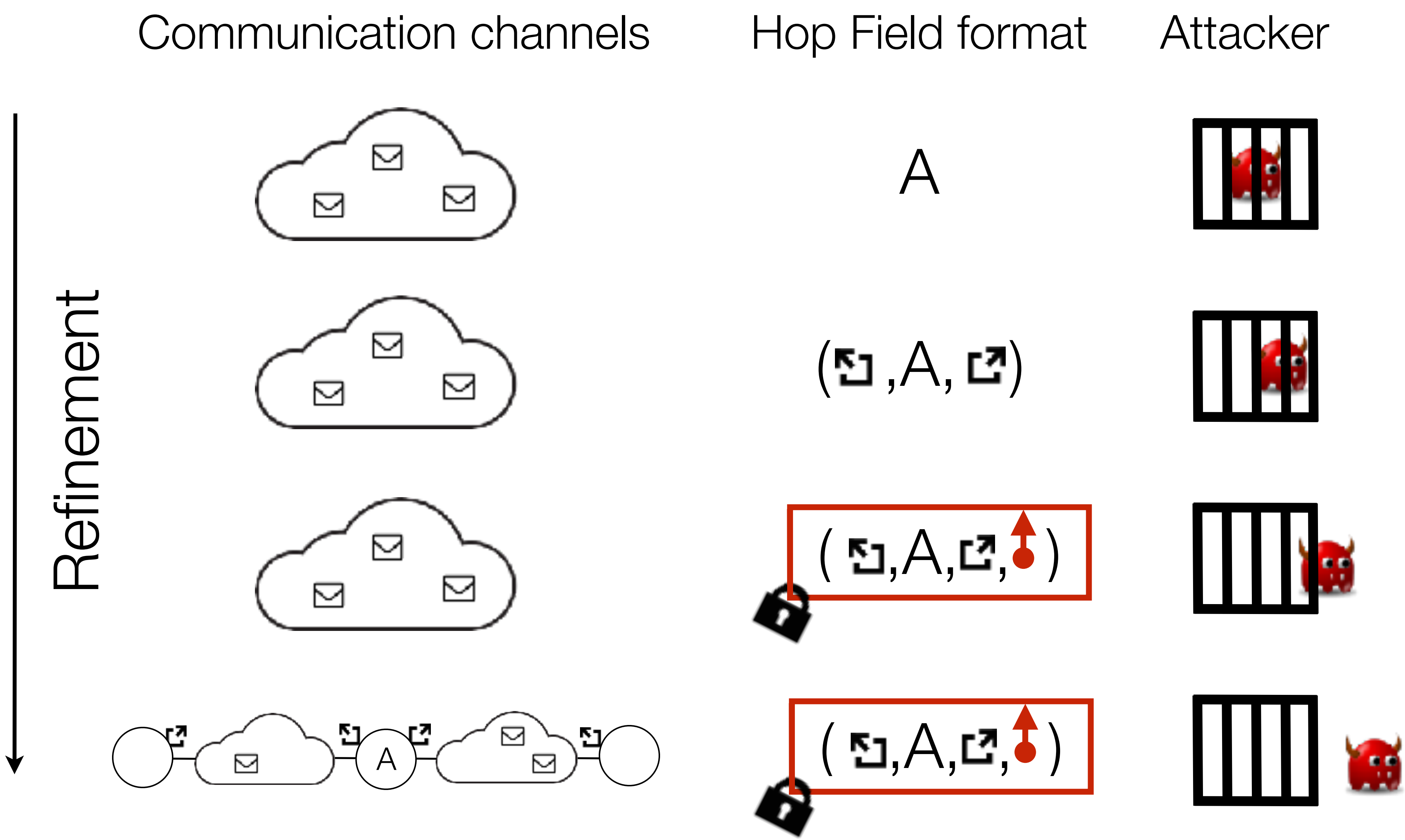
Abstract Packet Format

The Path is the Packet



- A path is a sequence of **Hop Fields (HF).**
- Each Hop Field contains routing information for one AS.
- Path is separated into Past and Future parts that indicate the packet's position in network.

Refinement Overview



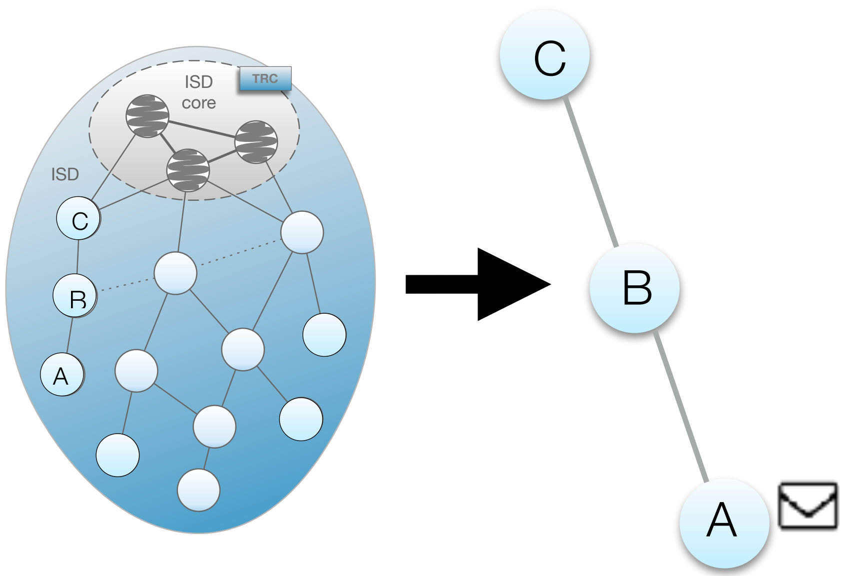
Idea: strengthen attacker while increasing protection of paths.

: Message set   : Neighbor ASes   : MAC   : Fields protected by MAC

Simplified Scenario (Initially)

Packet traversal along a single up-segment

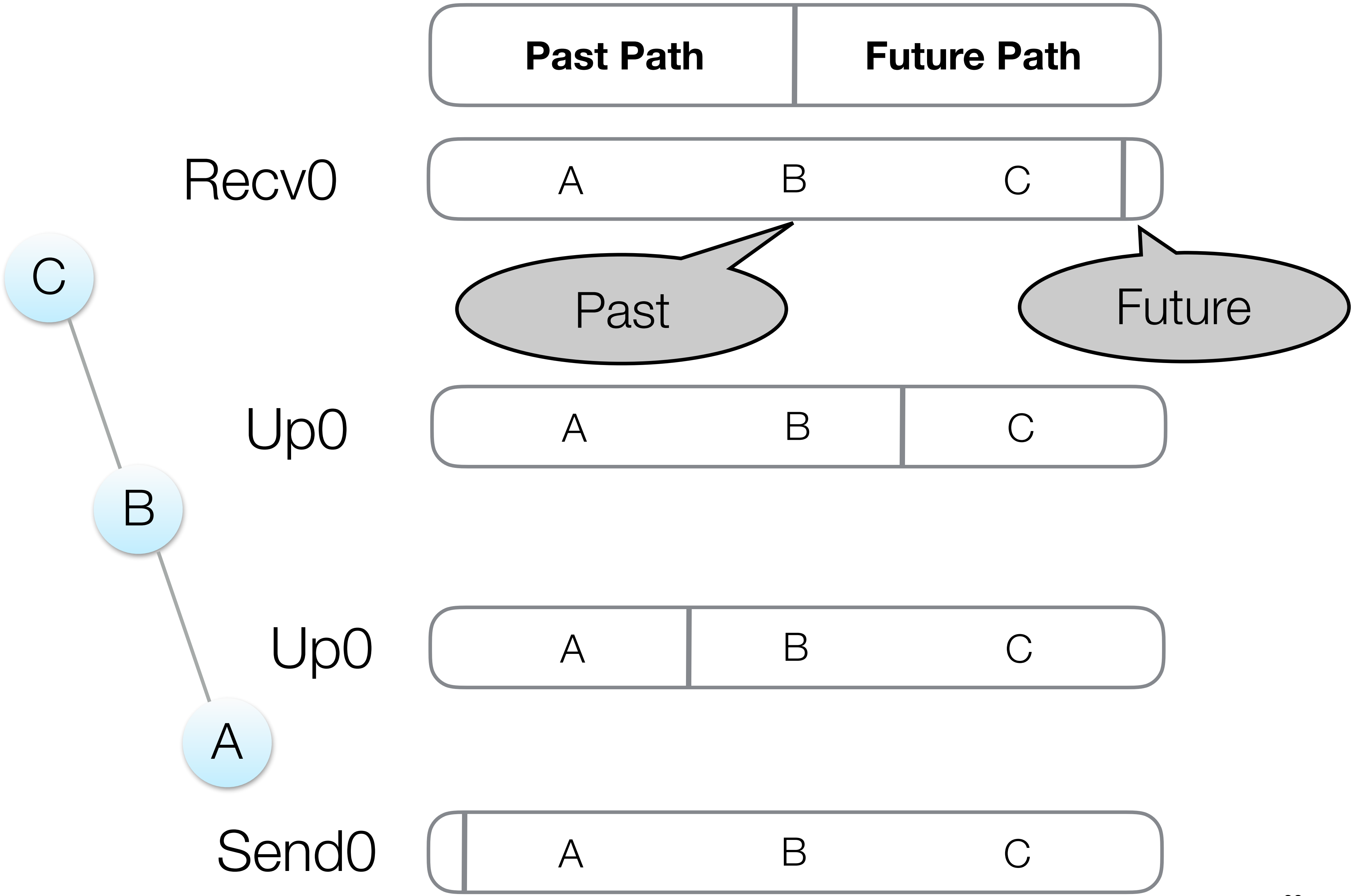
- A set of authorized-paths from path server is given as parameter
- Simplified setting
  - Ignore core- and down-segments
  - No peering or core links
  - Single ISD
  - No changes in link status (up/down)



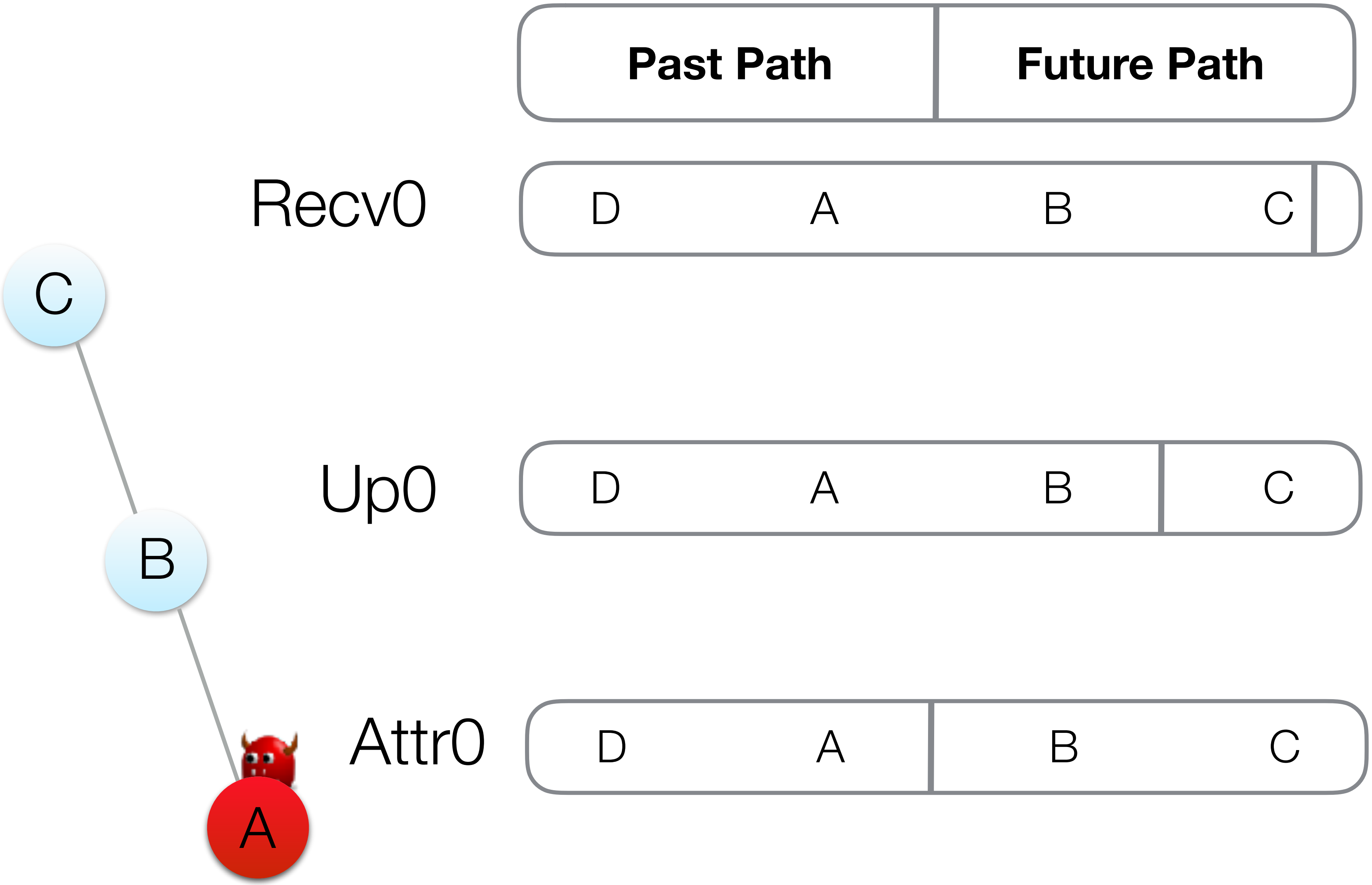
Verification is still challenging enough!

Data Plane Model 0

Example of one Packet along a simple Path

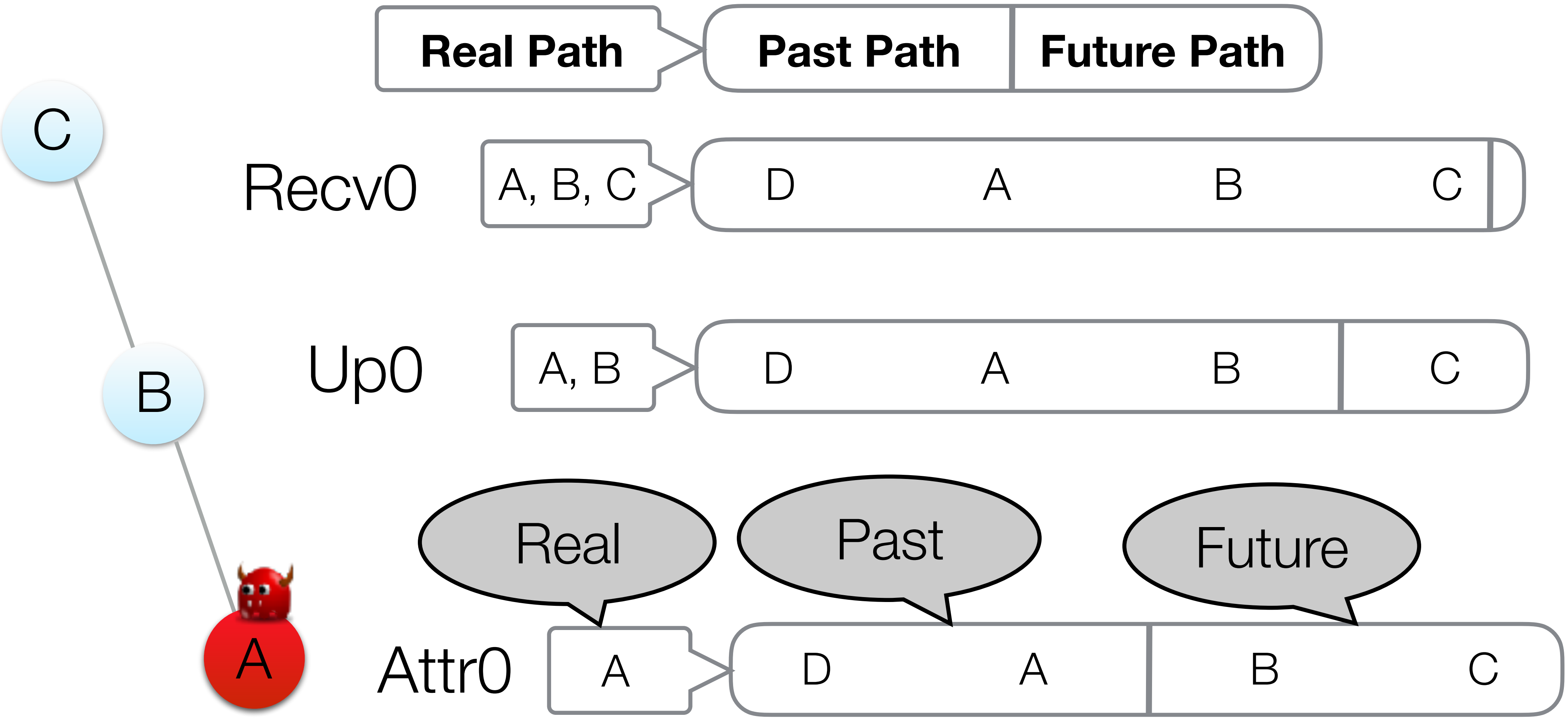


Data Plane Model 0



Problem: Past Path is unreliable

Data Plane Model 0

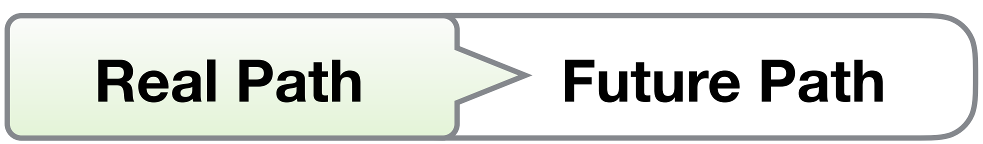


- Add new component, **real path**, to message
- History variable recording **actually path traversed** so far (Not part of actual implementation)

Formalized Properties of Model 0

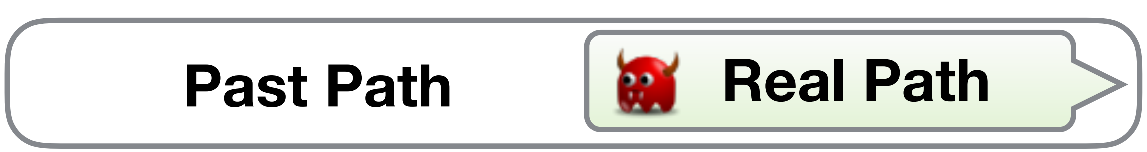
Assumption (control plane) Assume a set of **authorized-paths** resulting from beaconing process.

- **Path authorization:** Packets are forwarded only along previously authorized paths.



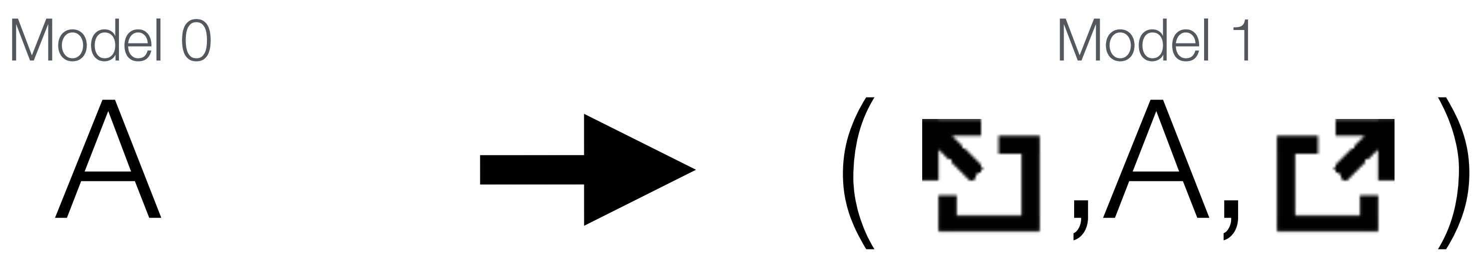
∈ **authorized-paths**

- **Weak detectability** An attacker cannot hide his presence on the path; follows from the following suffix property:



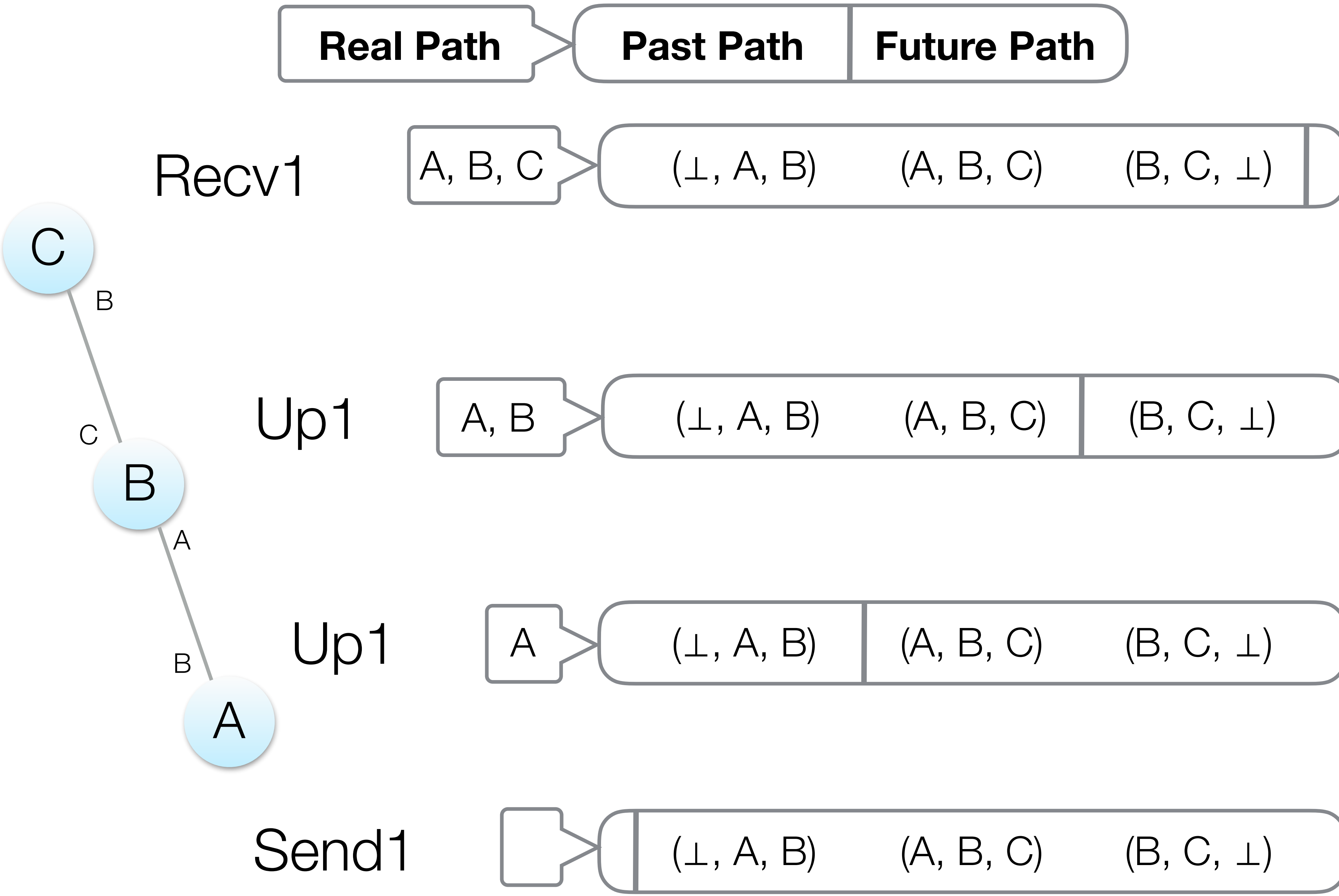
Data Plane Model 1

Hop Field format is refined:



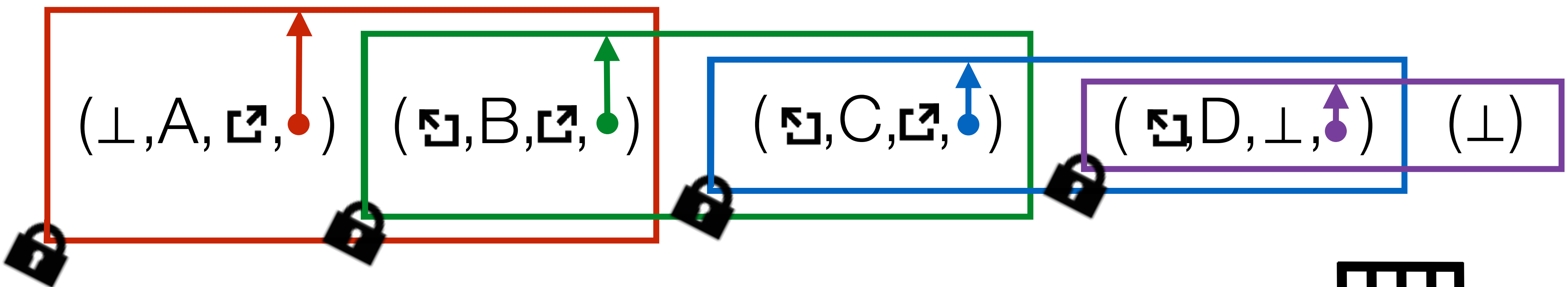
Added: references to previous and next AS

Data Plane Model 1



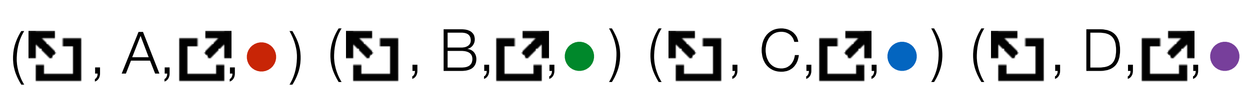
Data Plane Model 2: "Chaining" of MACs

Hop Field format is further refined by adding a MAC

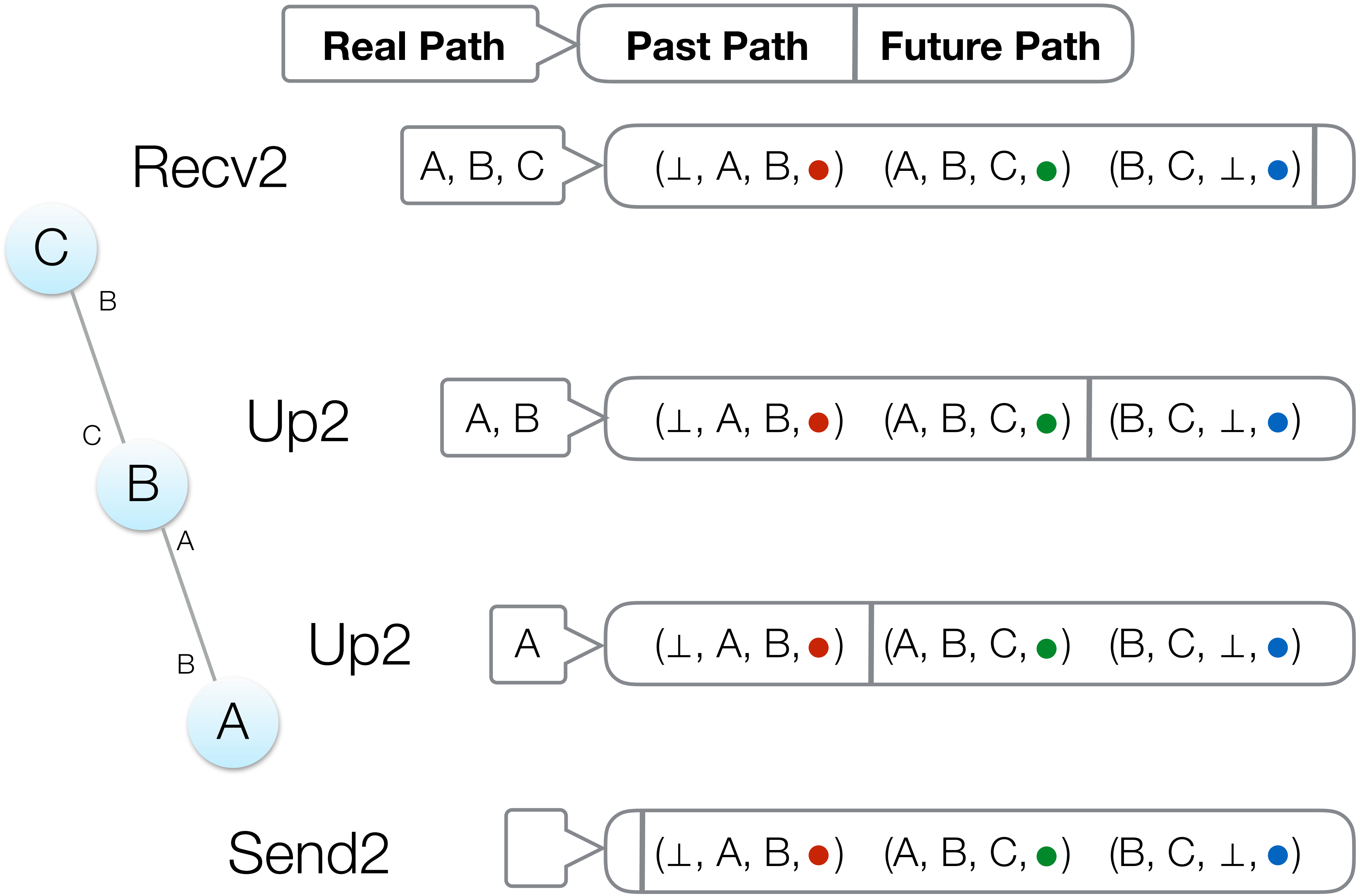


- MAC at *A* is produced with a *key(A)* known only to *A*
- MAC includes data and MAC of subsequent Hop Field (needed for verification)

Simplified representation:



Data Plane Model 2

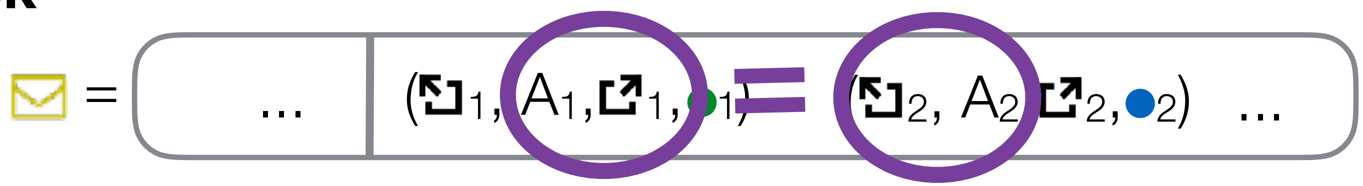


Up-Event in Model 2

Guard

In select in 

Check



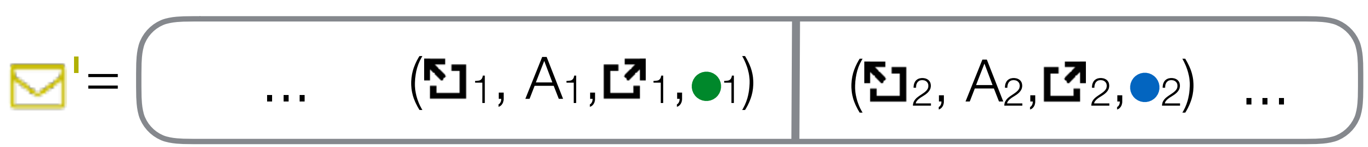
$\wedge$ ●$_1$ = valid MAC using *key*(A$_1$)

$\wedge$ ●$_2$ = valid MAC using *key*(A$_2$)

$\wedge$ ↗$_1$ = A$_2$ $\wedge$ ↖$_2$ = A$_1$

Action

Out put ✉' in  where



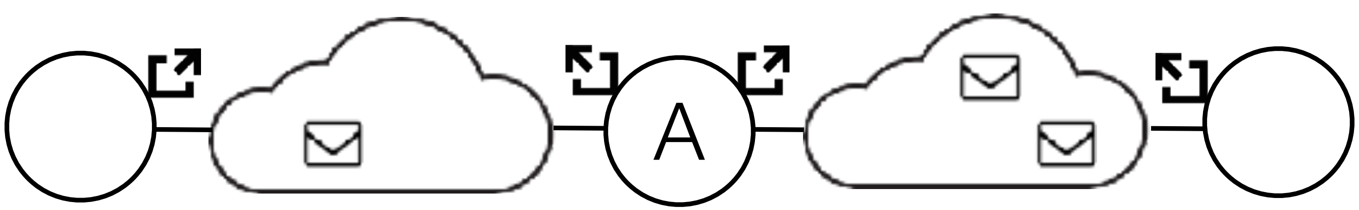
Refining Model 2

Model 2

Global Message Set

Refinement

Model 3

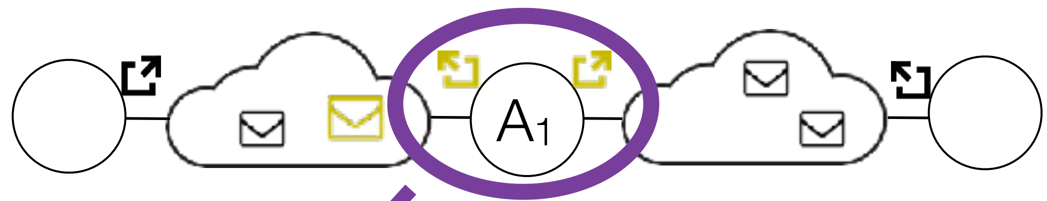


Inter-AS Message Sets

Up-Event in Model 3

Guard

In select from



Check

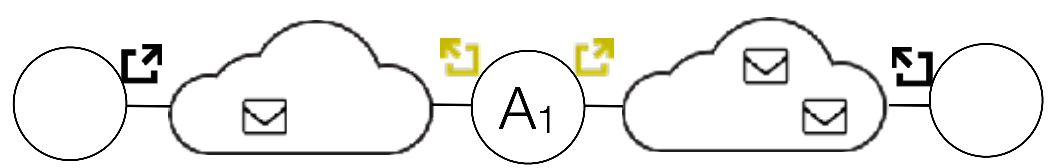
= | ... | $_1$, A_1, $_1$ ●$_1$) ... |

$\wedge$ ●$_1$ = valid MAC using *key*(A_1)

$\wedge$ $_1$ = $\wedge$ $_1$ =

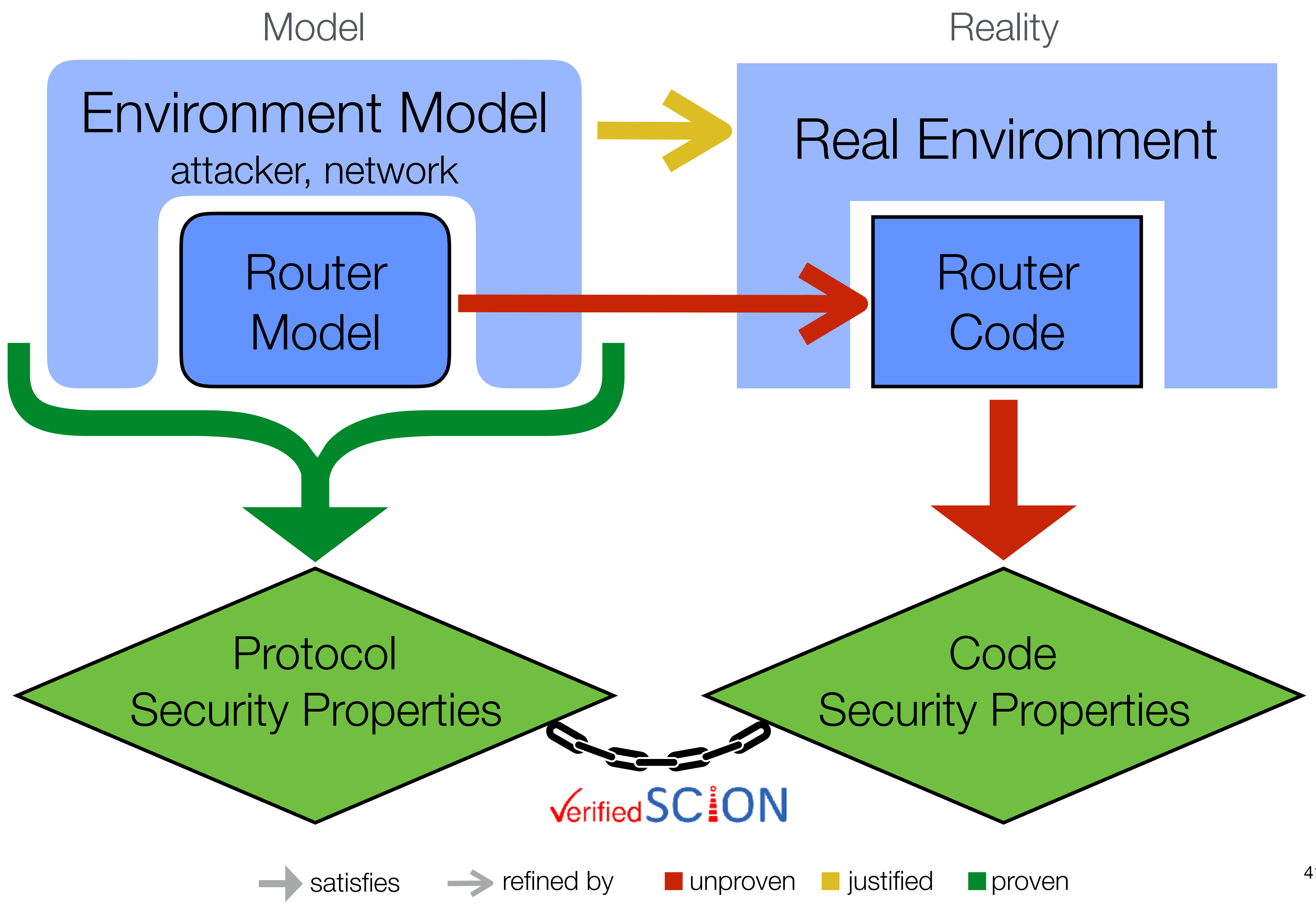
Action

Out put ' in

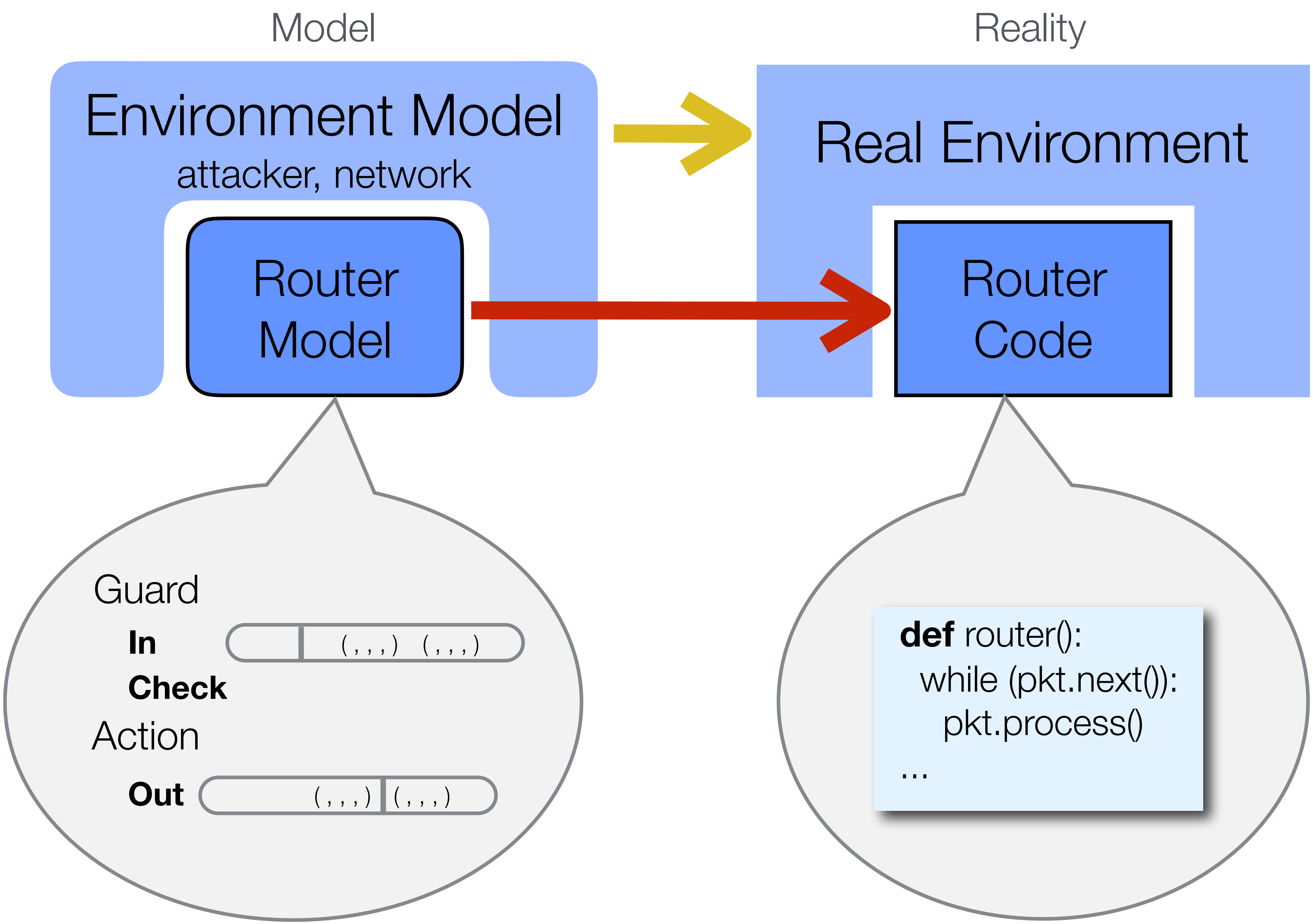


'= | ... ($_1$, A_1, $_1$, ●$_1$) | ... |

SCION Router Verification Overview

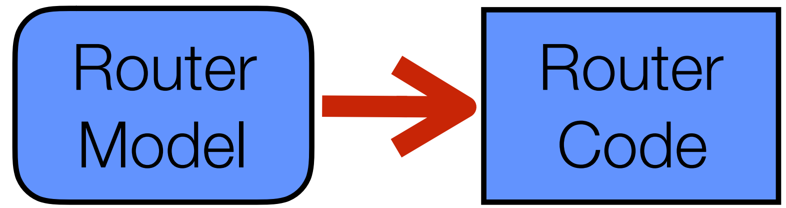


Router Model vs. Code

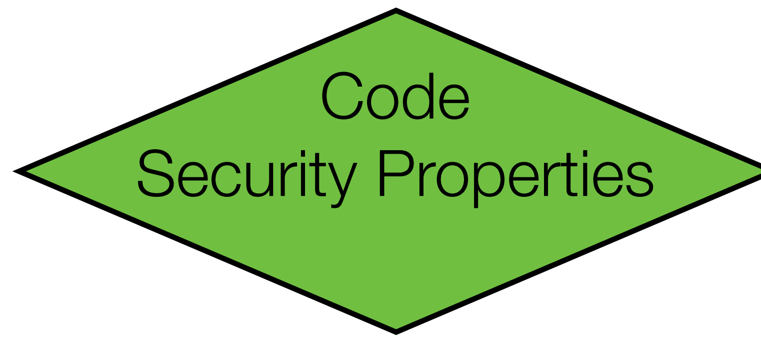


Code-Level Verification

- Main goal: prove **functional correctness**.
  - Code refines the protocol.



- Other desirable properties **only on code level**:
  - **Safety**: Code does not raise runtime exceptions or have data races.
  - **Secure information flow**: Code does not leak any information about crypto keys.
  - **Liveness and deadlock freedom**



- Focus on the SCION code base.
  - Used libraries are given specifications, **assumed** to be correct.
  - Runtime, OS, ..., are **assumed** to be correct.

Program Verification

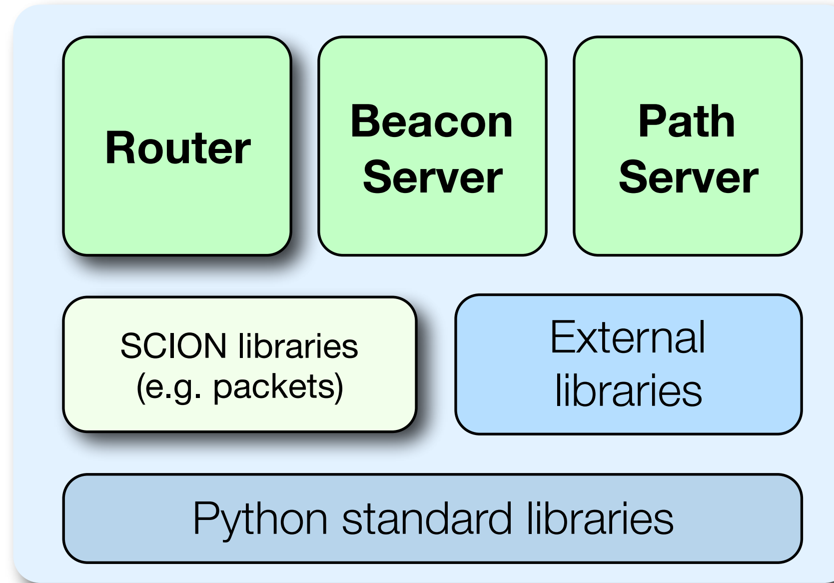
- Formal **specification** for each method
  - Pre- and postcondition, loop invariants

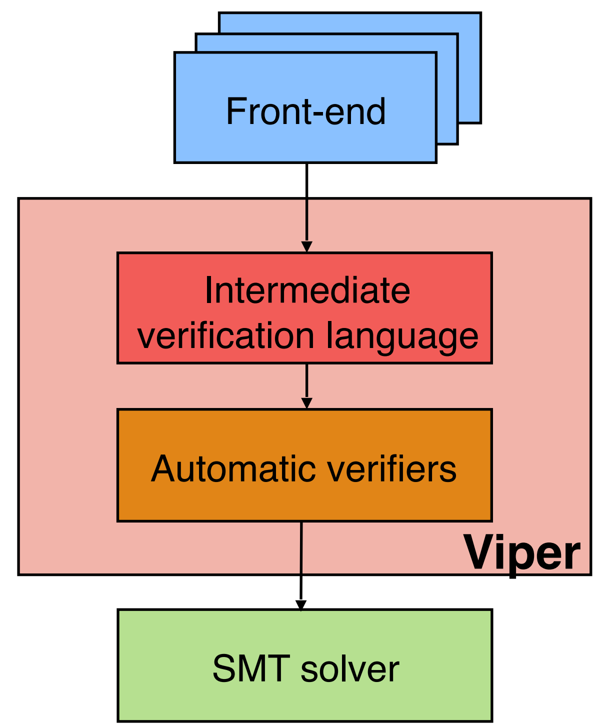
```
{n >

def sqrt(n):
  ...
  return result

{n =
```

- Formal **proof** that implementation satisfies specification.
  - Assuming **precondition** holds at the beginning, prove that **postcondition** holds after return (partial correctness).
  - For all possible inputs, schedules, callers, ...
  - Additional proof obligations for special properties, like progress

Code-based Verification

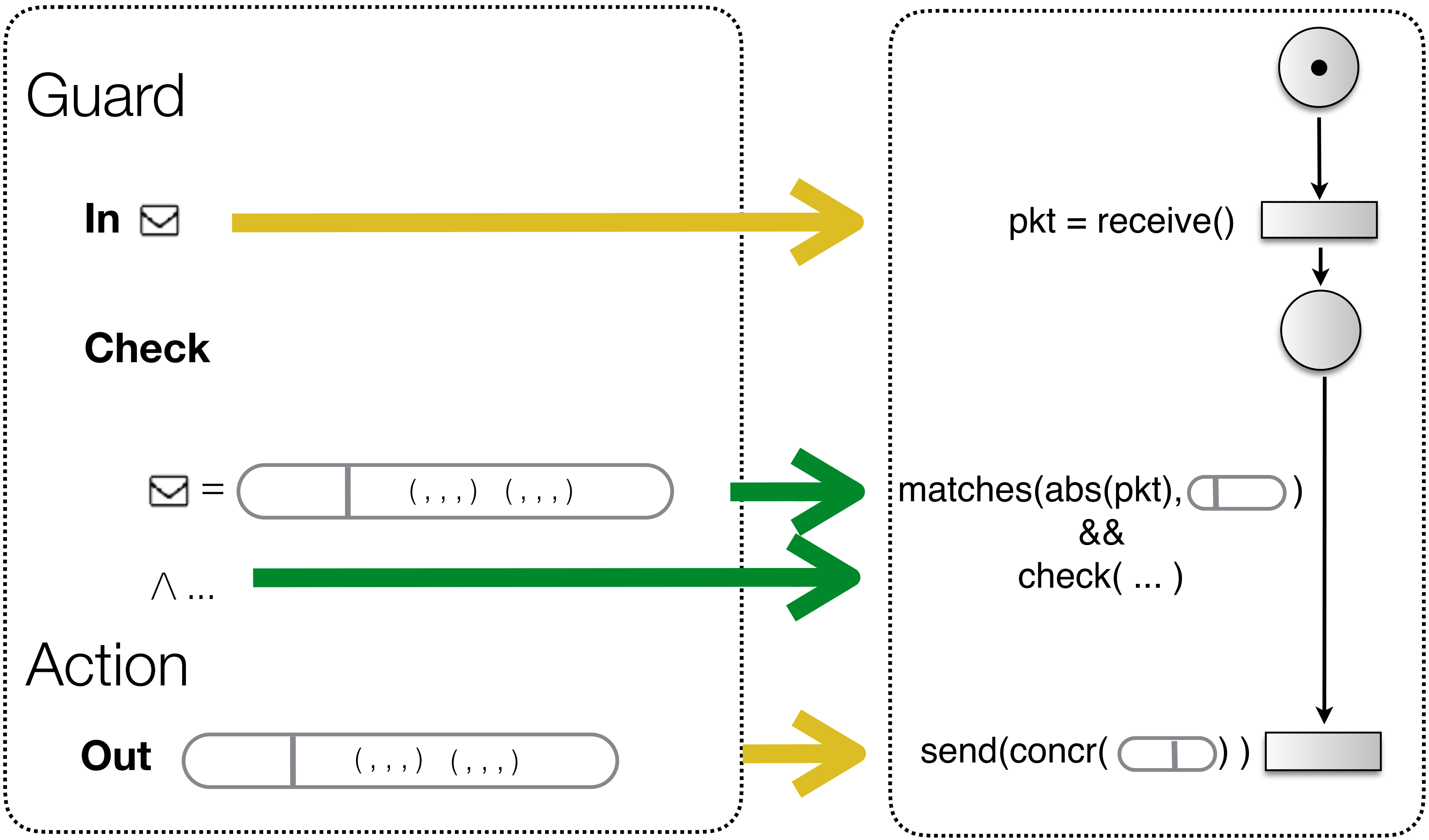
- Scion in Python 3
  - ~11k LOC
- Substantial subset of Python
  - Most standard OOP features
  - e.g. inheritance, exceptions, concurrency
- Focus on router first
- Use Viper Toolchain with Python front end





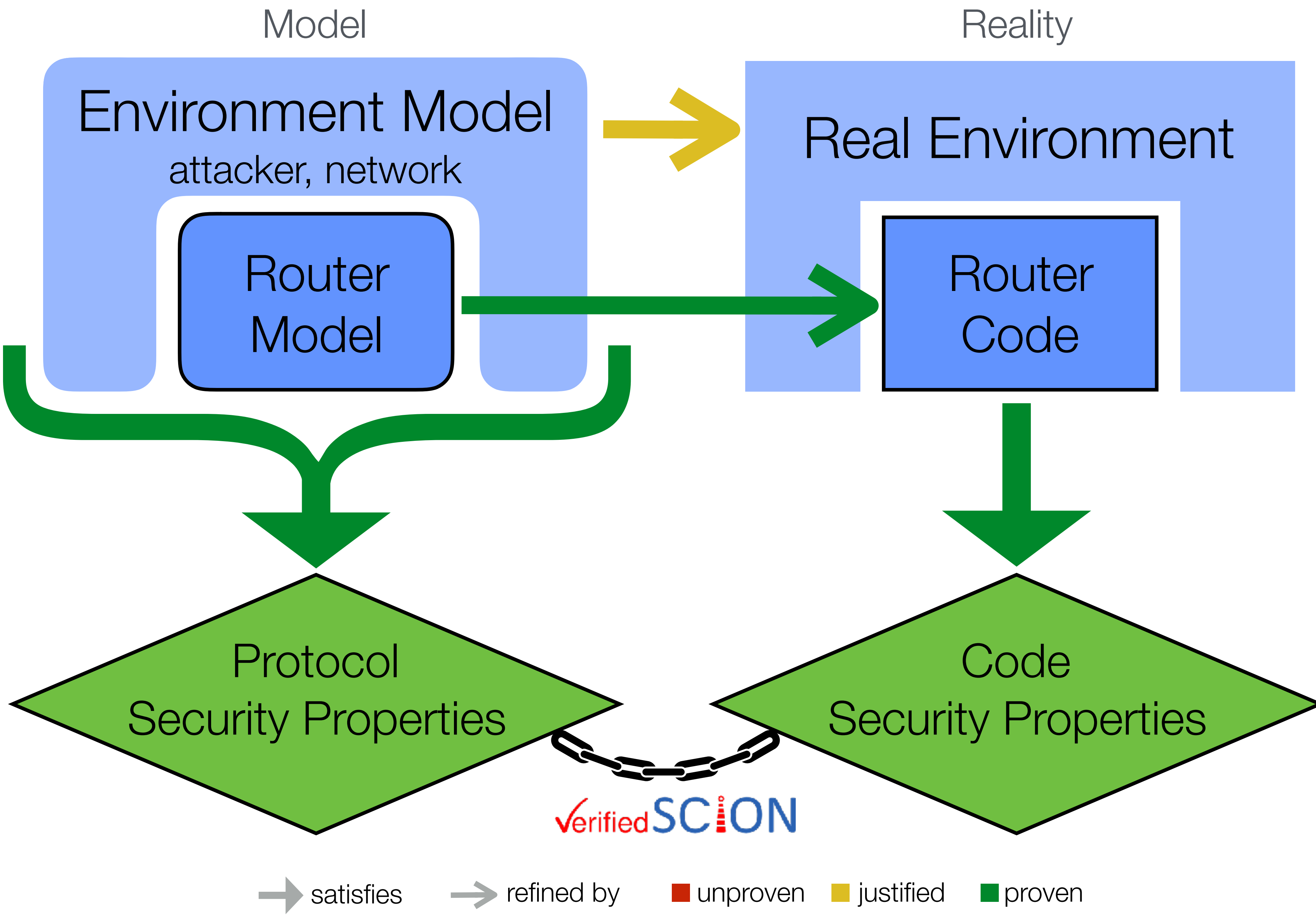
Linking it all up via Input-Output Specifications

(Code can be viewed as a transition system)



Based on: Pennickx, Jacobs, Piessens, "Sound, modular and compositional verification of the input/output behavior of programs", ESOP 2015.

SCION Router Verification Overview



Status

- Code verification tools built and prototyped
- First three levels of refinement completed
  - Improved understanding of protocols and properties
  - Uncovered numerous bugs and omissions
    - Revealed during modeling & formalization
    - Verified against implementation
- Next step: formally connect the two parts

Conclusions

- Internet, as designed, is insecure
- Scion architecture offers much stronger guarantees
- These can be put on a formal footing via

  refinement + code-level verification

- **Long term objective**: guaranteed back-door-free routers, made in Switzerland

anapaya systems

We are hiring: www.anapaya.net