verifiedSCION: Verified Secure Routing

Peter Müller

Joint work with the verifiedSCION Team at ETH

ETH zürich

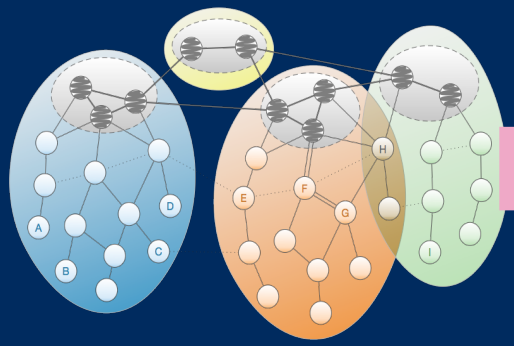
Security and Correctness

- Protocol-level properties
  - **Path validity**: Constructed paths are valid and reflect the routing decisions by on-path ASes
  - **Path authorization**: Packets travel only along previously authorized paths
  - **Detectability**: An active attacker cannot hide their presence on the path

- Code-level properties
  - **Safety**: No run-time errors
  - **Correctness**: Routers and servers implement protocol correctly
  - **Progress**: Required I/O happens eventually
  - **Backdoor freedom**: Code does not leak information about crypto keys

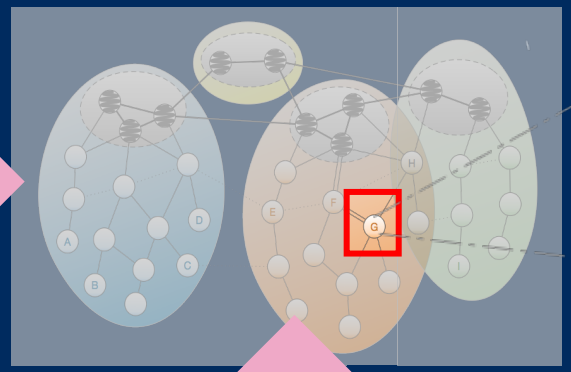
Formal end-to-end verification of security and correctness

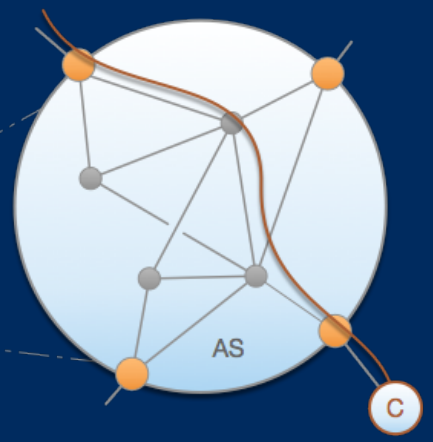
Mathematical model of entire network



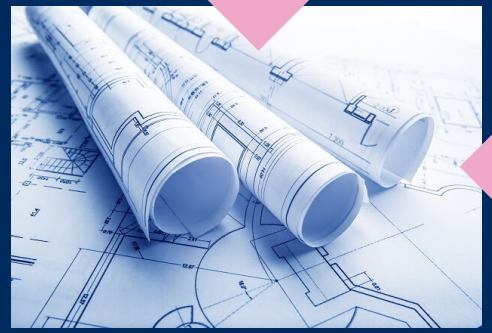
Refinement

Mathematical model of border router





Equivalence



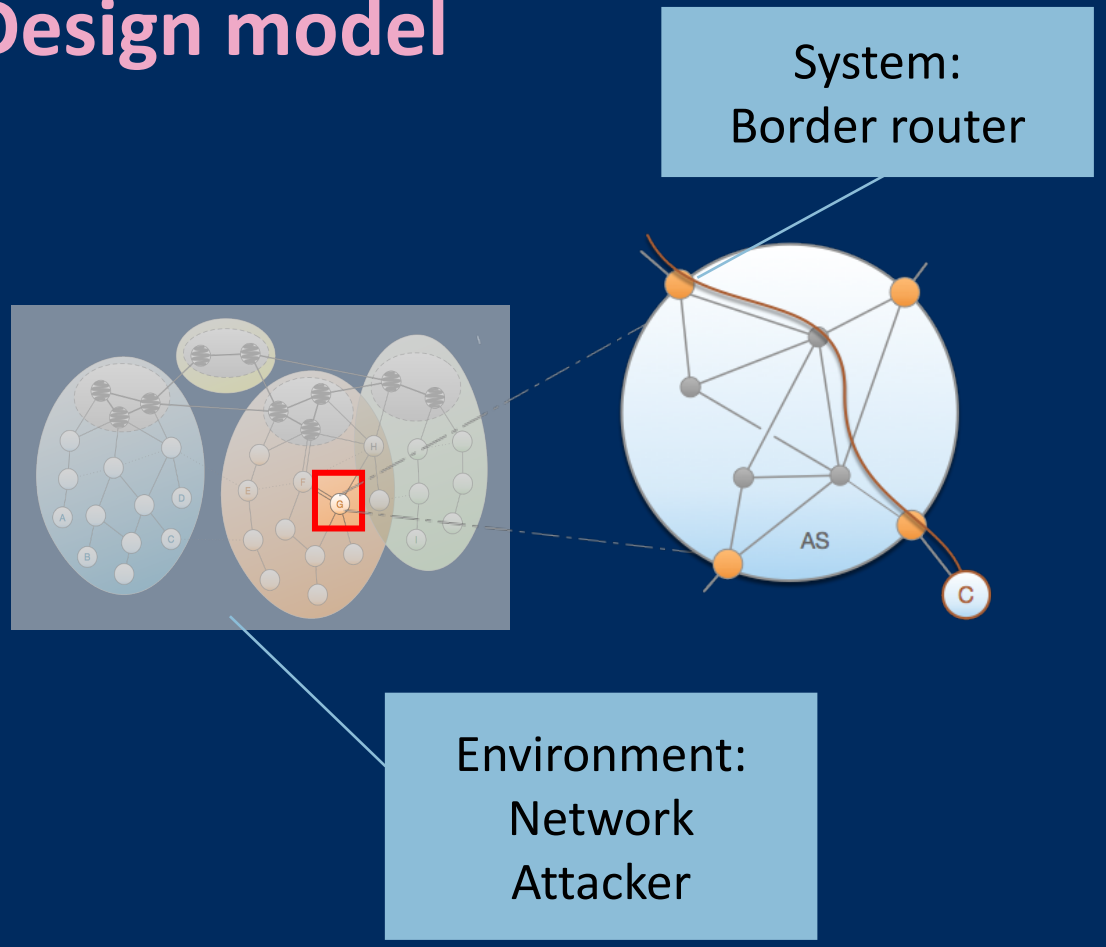
Verification

Router specification

Router implementation

Protocol Verification

Design model



Stepwise refinement

- Prove properties of most abstract model
- Each refinement
  - Incorporates additional system requirements
  - Preserves properties of more-abstract system
- Strategy: strengthen attacker while increasing security features

Program Verification

Specification:
What is the intended behavior?

Program:
How is the behavior achieved?

Verified properties

- No run-time errors
- Termination
- Functional properties
- I/O behavior
- Progress
- Backdoor freedom

Status and Milestones

Key results

- Theory & technology
  - Program verification techniques
  - Integration of protocol and program verification
- Proof of concept
  - Verification of packet forwarding
  - Verification of path authorization and detectability
  - Verification of parts of the Python prototype

Upcoming milestones

- Q4/19
  - Basic Go verifier
- Q2/20
  - Formal model of control plane
  - Formal model of bandwidth reservation
  - Verification of packet forwarding
- Q4/20
  - Full-fledged Go verifier

Conclusion

- IP implementations are complex and large
  - They inevitably have both design and code-level bugs
  - Some of these bugs can be exploited by attackers
- The design of Scion enables formal verification of protocol and code
- Verification provides unprecedented guarantees to ISPs and end users
  - Functional correctness
  - Availability
  - Security, in particular, backdoor freedom