Private and Verifiable Interdomain Routing Decisions

Mingchen Zhao*  Wenchao Zhou*  Alexander Gurney*
Andreas Haeberlen*  Micah Sherr+  Boon Thau Loo*

* University of Pennsylvania  + Georgetown University
Motivating Scenario
Motivating Scenario

- ASes are making "promises" about their routing behavior
  - Needed to implement various routing goals, e.g., meeting peer traffic ratios, avoiding use of expensive uplinks, ...
  - Neighbors may depend on these being implemented correctly
Motivating Scenario

- ASes are making "promises" about their routing behavior
  - Needed to implement various routing goals, e.g., meeting peer traffic ratios, avoiding use of expensive uplinks, ...
  - Neighbors may depend on these being implemented correctly

I will always give you my shortest route to Google!
Motivating Scenario

- ASes are making "promises" about their routing behavior
  - Needed to implement various routing goals, e.g., meeting peer traffic ratios, avoiding use of expensive uplinks, ...
  - Neighbors may depend on these being implemented correctly
Problem: Detecting broken promises
Problem: Detecting broken promises
Problem: Detecting broken promises
Problem: Detecting broken promises

How do I know whether Bob is keeping his promise?
Problem: Detecting broken promises

Goal: Verify whether the promise is kept

How do I know whether Bob is keeping his promise?
Can auditing help?

Example: NetReview (NSDI 2009)
Can auditing help?

Example: NetReview (NSDI 2009)
Challenge: Privacy

I do not want to reveal all my routes to Alice!
Can we have our cake and eat it too?

S-BGP, soBGP, psBGP, NetReview, ...
Goals

• **Security**: If Bob breaks his promise, Alice will detect it.

• **Privacy**: Verification does not reveal more information than BGP.

• **Evidence**: If Bob breaks his promise, Alice can prove it.

• **Accuracy**: If Bob does not break his promise, nobody can prove he did.
Approach: Collaborative Verification

Charlie → Bob → Alice → Doris → Eliot
Approach: Collaborative Verification

- Idea: break the verification into small pieces
Approach: Collaborative Verification

- Idea: break the verification into small pieces
- Assign each piece to someone who can verify it with only local knowledge
- Successful verification of all pieces implies that the promise has been kept
Outline

• Motivation
  • Goal: Verify promises about routing decisions
  • Challenge: Privacy

• The SPIDeR system

• Evaluation

• Summary
Modeling promises

4 hop Euro peer

3 hop Asia Customer

3 hop Euro peer

5 hop Asia Provider

4 hop Africa Customer

3 hop Africa Provider
Modeling promises

3 hop route: Africa Provider

4 hop route: Euro peer

5 hop route: Euro Provider

Indifference Class
Modeling promises

4 hop Euro peer

3 hop Asia Customer

5 hop Euro Provider

4 hop Africa Customer

3 hop Africa Provider
Modeling promises

Customer route

4 hop
Africa
Customer

Peer route

3 hop
Euro
peer

Provider route

3 hop
Africa
Provider

Indifference Class
Our study of 88 real AS policies shows that most of the popular promises can be modeled in this way.
Background: Merkle Hash Tree

- Merkle Tree

Commitment

Hash

Values

Hash

b₁

Hash

b₂

Hash

b₃

Hash

b₄

Path to the root

Proof that the second value is $b₂$

Reveals nothing about $b₁$, $b₃$, $b₄$!
SPIDeR: single prefix case

Bob

Alice
Charlie
Doris
Eliot

Merkle hash tree

Bit k set to 1: "I have a route that is at most k hops long"
SPIDeR: single prefix case

Bit k set to 1: "I have a route that is at most k hops long"
Bit k set to 1: "I have a route that is at most k hops long"
SPIDeR: single prefix case

Bit k set to 1: "I have a route that is at most k hops long"
SPIDeR: single prefix case

Bit k set to 1: "I have a route that is at most k hops long"
SPIDeR: single prefix case

Bit k set to 1: "I have a route that is at most k hops long"
Single prefix case

a) No bit below 2 is set; this is the shortest route!
b) All bits above 2 are set; Bob didn't lie to the others!

Bit k set to 1: "I have a route that is at most k hops long"
SPIReR: single prefix case

Bit k set to 1: "I have a route that is at most k hops long"
Single prefix case

Bit 2 is set; there is a better route.

(3+1) hop

Bob

3 hop

5 hop

2 hop

Alice

Merkle hash tree

Bit k set to 1: "I have a route that is at most k hops long"
SPIDeR: single prefix case

- If Bob picks the correct route, no neighbor learns anything new!
- If Bob picks the wrong route, at least one neighbor can detect it!

Bit k set to 1: "I have a route that is at most k hops long"
Making SPIDeR practical

• So far: We can verify promises about a single prefix and a single decision
  • We have a protocol
  • It meets all four goals
  • We proved the correctness (in a TR)
  • Guarantees hold even if an AS is malicious

• Practical issues
  • Multiple prefixes, temporal privacy, loose synchronization, logging system, withdrawals, incremental deployment
Multi-Prefix: Additional Challenges

• Running one-prefix protocol for each prefix owned by Bob?
  • It releases private information about which prefixes Bob has.

• Running single-prefix protocols for all possible prefixes?
  • It is not efficient. There are $2^{33} - 1$ possible different prefixes.
Modified Ternary Tree

- Efficiency: run one instance to verify all the prefixes
- Privacy: verifiers cannot learn anything about other prefixes.
Outline

- Motivation
- The SPIDeR system
  - Single prefix
  - Practical Challenges
- Evaluation
  - Functionality check
  - Microbenchmarks
  - Overhead
- Summary
Evaluation: Microbenchmarks

• An important metric is how fast we can make hash trees.
  • How quickly can we capture transient routing configuration problems?

• Experiment: generate a tree for a full BGP routing table on Dell PowerEdge 860.
  • Result: 17.4s (with three cores)
  • Scales almost linearly with the number of cores
Evaluation: Experimental Overhead

- Small AS topology with Quagga routers
- Injected a RouteViews trace
- AS 5’s SPIDeR ran on a single machine
Evaluation: Overhead

• Computation
  • 2.4 GHz core: 81.3% utilized
  • Commodity workstation is sufficient

• Bandwidth
  • Signatures etc.: 20.8kbps
  • Verifying 1% of commitments per minute: 3.0Mbps
  • On the order of a single DSL upstream link

• Storage
  • Keeping 1 year’s worth of logs: 145.7GB
  • Fits on a commodity hard drive
Evaluation: Overhead

• Computation
  • 2.4 GHz core: 81.3% utilized
  • Commodity workstation is sufficient

• Storage
  • Keeping 1 year’s worth of logs: 145.7GB
  • Fits on a commodity hard drive

A small AS could run SPIDeR on a single machine
  • On the order of a single DSL upstream link
Summary

• Goal: Verify promises about interdomain routing decisions
• Problem: Offer both security and privacy
• Solution: Collaborative verification
• Implemented in the SPIDeR system
  • Provable security and privacy guarantees
  • Efficient enough to run on a single commodity workstation

More information: http://snp.cis.upenn.edu/