Proof of Transit

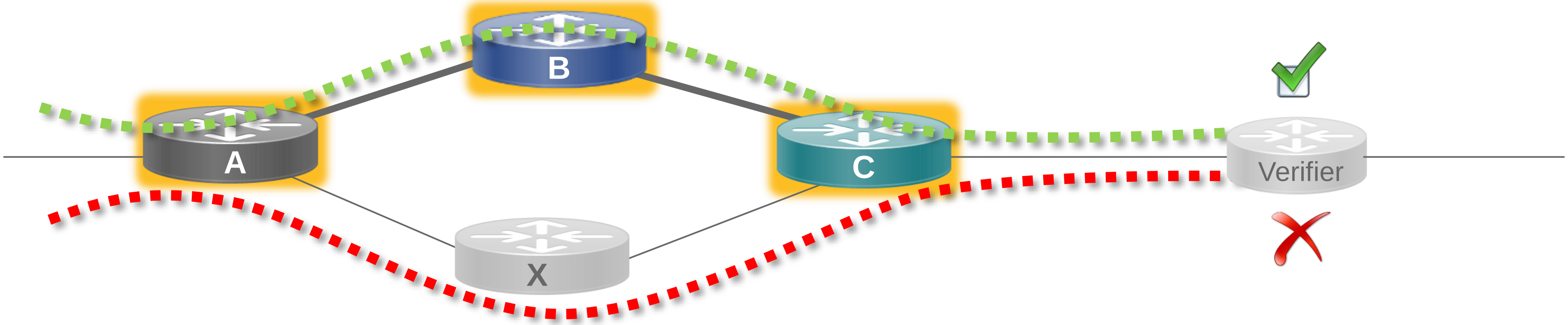
Frank Brockners, Shwetha Bhandari,
Sashank Dara, Carlos Pignataro (Cisco)
Hannes Gedler (rtbrick)
Steve Youell (JMPC)
John Leddy (Comcast)
David Mozes (Mellanox Technologies)
Tal Mizrahi (Marvell)

IETF 99 Prague – OPSEC WG; July 19th, 2017

draft-brockners-proof-of-transit-03.txt

Consider TE, Service Chaining, Policy Based Routing, etc...

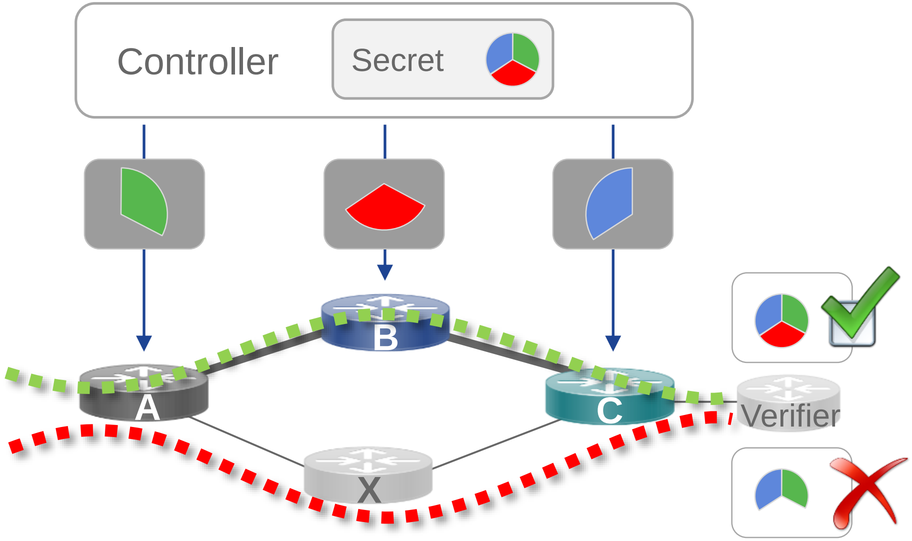
“How do you *prove* that traffic follows the selected path?”


A
B
C
X
Verifier

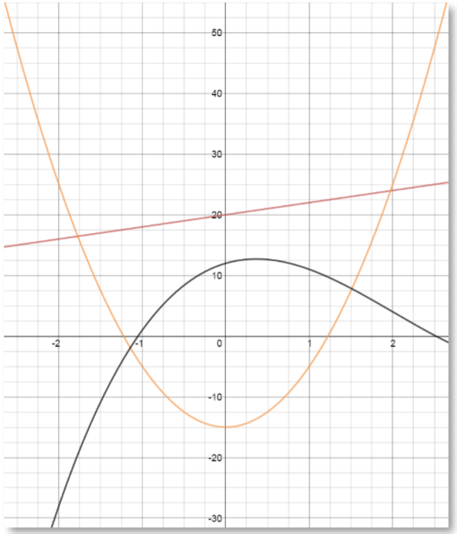
Ensuring Path and/or Service Chain Integrity
Approach

- Meta-data added to all user traffic
  - Based on “Share of a secret”
  - Provisioned by controller over secure channel to hops where “proof of transit” is required
  - Updated at every hop where proof of transit is required
- Verifier checks whether collected meta-data allows retrieval of secret

➡ “Proof of Transit”: Path verified



Solution Approach 1: Leveraging Shamir's Secret Sharing Polynomials 101



$f2(x) = 10x^2 - 15$

- Parabola: Min 3 points

$f1(x) = 2x + 20$

- Line: Min 2 points

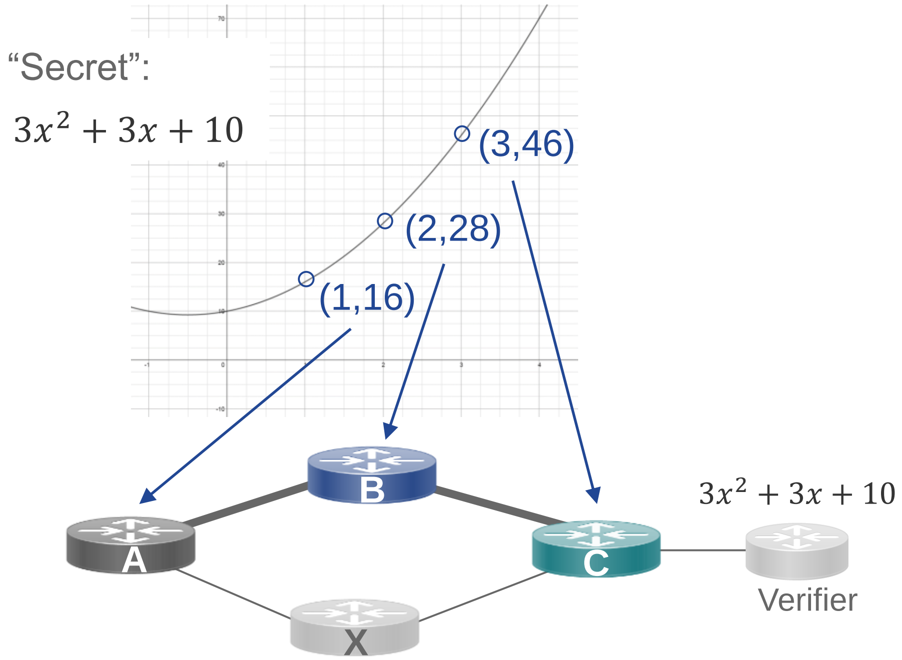
$f3(x) = x^3 - 6x^2 + 4x - 12$

- Cubic function: Min 4 points

General: It takes k+1 points to defines a polynomial of degree k.

Solution Approach 1: Leverage Shamir's Secret Sharing
"A polynomial as secret"

- Each service is given a point on the curve
- When the packet travels through each service it collects these points
- A verifier can reconstruct the curve using the collected points
- Operations done over a finite field (mod prime) to protect against differential analysis



Operationalizing the Solution Approach 1

- Leverage two polynomials:
  - POLY-1 secret, constant: Each hop gets a point on POLY-1
    Only the verifier knows POLY-1
  - POLY-2 public, random and per packet.
    Each hop generates a point on POLY-2 each time a packet crosses it.
- Each service function calculates (Point on POLY-1 + Point on POLY-2) to get (Point on POLY-3) and passes it to verifier by adding it to each packet.
- The verifier constructs POLY-3 from the points given by all the services and cross checks whether POLY-3 = POLY-1 + POLY-2
- Computationally efficient: 2 additions, 1 multiplication, mod prime per hop

POLY-1
Secret – Constant

+

POLY-2
Public – Per Packet

=

POLY-3
Secret – Per Packet

Operationalizing the Solution Approach 1

- Leverage two polynomials:
  - POLY-1 secret, constant: Each hop gets a point on POLY-1
    Only the verifier knows POLY-1
  - POLY-2 public, random and per packet.
    Each hop generates a point on POLY-2 each time a packet crosses it.
- Each service function calculates (Point on POLY-1 + Point on POLY-2) to get (Point on POLY-3) and passes it to verifier by adding it to each packet.
- The verifier constructs POLY-3 from the points given by all the services and cross checks whether POLY-3 = POLY-1 + POLY-2
- Computationally efficient:
  2 additons, 1 multiplication, mod prime per hop

POLY-1
Secret – Constant

+

POLY-2
Public – Per Packet

=

POLY-3
Secret – Per Packet

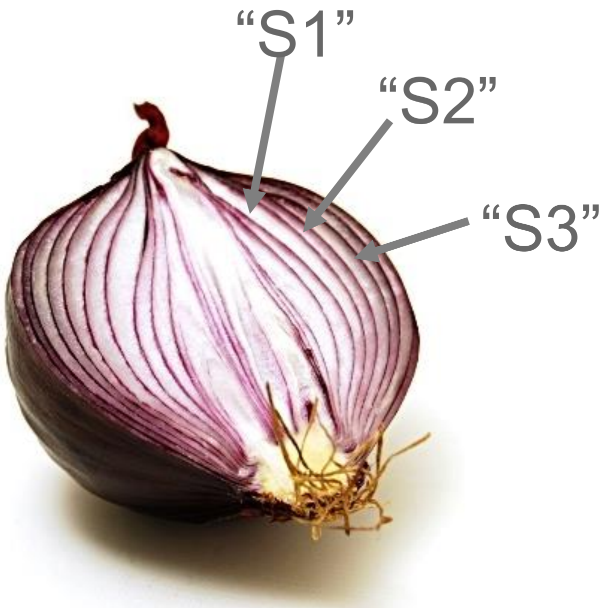
Meta Data for Service/Path Verification

- Verification secret is the independent coefficient of POLY-1
  - Computation/retrieval through a cumulative computation at every hop ("cumulative")
- For POLY-2 the independent coefficient is carried within the packet (typically a combination of timestamp and random number)
  - n bits can service a maximum of 2^n packets
- Verification secret and POLY-2 coefficient ("random") are of the same size
  - Secret size is bound by prime number

Transfer Rate	RND/ Secret Size	Max # of packets (assuming 64 byte packets)	Time that "random" lasts at maximum
1 Gbps	**64**	$2^{64} \approx 2 * 10^{19}$	approx. 10^{13} seconds, approx. 310000 years
10 Gbps	**64**	$2^{64} \approx 2 * 10^{19}$	approx. 10^{12} seconds, approx. 31000 years
100 Gbps	**64**	$2^{64} \approx 2 * 10^{19}$	approx. 10^{11} seconds, approx. 3100 years
10 Gbps	56	$2^{56} \approx 7 * 10^{16}$	approx. 10^{9} seconds, approx. 120 years
10 Gbps	48	$2^{48} \approx 2 * 10^{14}$	approx. 10^{7} seconds, approx. 5.5 months
10 Gbps	40	$2^{40} \approx 1 * 10^{12}$	approx. $5 * 10^{4}$ seconds, approx. 15 hours
1 Gbps	32	$2^{32} \approx 4 * 10^{9}$	2200 seconds, 36 minutes
10 Gbps	32	$2^{32} \approx 4 * 10^{9}$	220 seconds, 3.5 minutes
100 GBps	32	$2^{32} \approx 4 * 10^{9}$	22 seconds

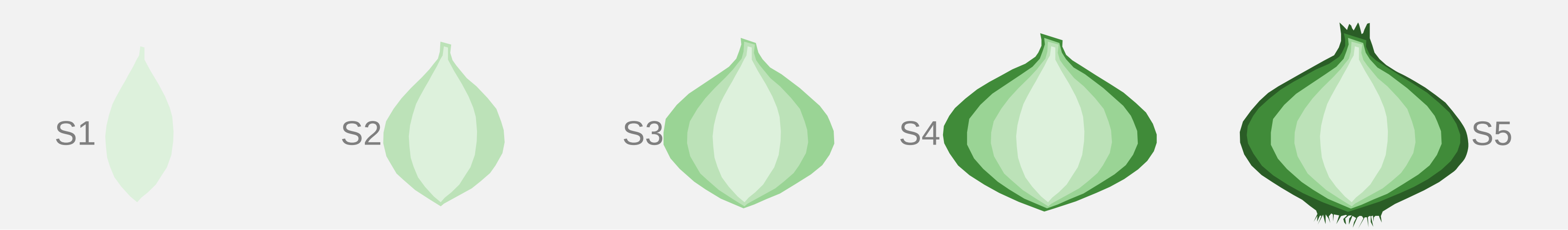
Solution Approach 2: Nested Crypto: “Compose an Onion”

- Approach
  - A service is described by a set of secrets, where each secret is associated with a service function. Service functions encrypt portions of the meta-data as part of their packet processing.
  - Only the verifying node has access to all secrets. The verifying nodes re-encrypts the meta-data to validate whether the packet correctly traversed the service chain.
- Notes
  - Nested encryption allows to check the order in which the nodes where traversed
  - To be used only when hardware assisted encryption is available. i.e. AES-NI instructions or equivalent. Otherwise this could be very costly operation to verify at line speed.



Service-Secrets are nested like layers of an onion

Solution Approach 2: “Compose the Onion”



1. A controller provisions all the nodes with their respective secret keys.
2. A controller provisions the verifier with all the secret keys of the nodes.
3. For each packet, the ingress node generates a random number RND and encrypts it with its secret key to generate CML value
4. Each subsequent node on the path encrypts CML with their respective secret key and passes it along
5. The verifier is also provisioned with the expected sequence of nodes in order to verify the order
6. The verifier receives the CML, RND values, re-encrypts the RND with keys in the same order as expected sequence to verify.

Proof of Transit: Meta-Data Transport Options

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Random                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Random (contd)                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Cumulative                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Cumulative (contd)                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

2nd Polynomial

Interative computation of secret

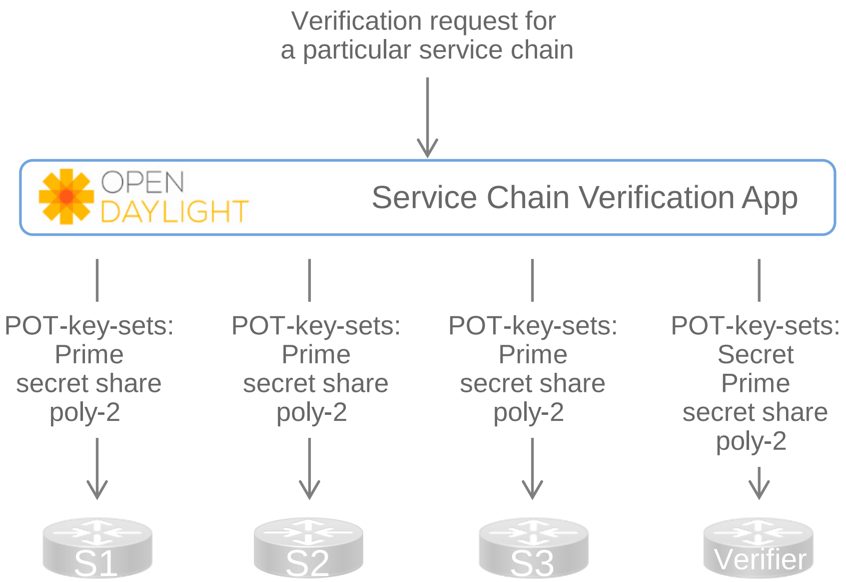
- Typical: 16* Bytes of Meta-Data
  - Random – Unique random number (e.g. Timestamp or combination of Timestamp and Sequence number)
  - Cumulative (algorithm dependent)

- Transport options for different protocols
  - Network Service Header
  - IPv6
  - Segment Routing
  - GRE
  - ...

*Note: Smaller numbers are feasible, but require a more frequent renewal of the polynomials/secrets.

Meta-Data Provisioning

- Meta-Data for Service Chain Verification (POT) provisioned through a controller (OpenDaylight App)
- Netconf/YANG based protocol
- Provisioned information from Controller to Service Function / Verifier
  - Service-Chain-Identifier
    (to be mapped to service chaining technology specific identifier by network element)
  - Service count (number of services in the chain)
  - 2 x POT-key-set (even and odd set)
    - Secret (in case of communication to the verifier)
    - Share of a secret, service index
    - 2nd polynomial coefficients
    - Prime number



Open Source Implementation

OpenDaylight Carbon Release: Controller App

- User guide: http://docs.opendaylight.org/en/latest/user-guide/service-function-chaining.html?highlight=sfc#sfc-proof-of-transit-user-guide
- Developer guide: http://docs.opendaylight.org/en/latest/developer-guide/service-function-chaining.html?highlight=sfc#sfc-proof-of-transit-developer-guide

FD.io/VPP: Data Path Implementation

- Documentation: https://docs.fd.io/vpp/17.04/ioam_ipv6_doc.html
- Code: https://gerrit.fd.io/r/gitweb?p=vpp.git;a=tree;f=src/plugins/ioam/lib-pot;h=fcaeb0f1923b7d0f7d8680d5811b6166b59c516f;hb=refs/heads/master

See also: https://github.com/CiscoDevNet/iOAM
More detailed presentation on POT:
https://www.slideshare.net/frankbrockners/proof-of-transit-securely-verifying-a-path-or-service-chain

Next Steps

- The authors appreciate additional comments and feedback on draft-brockners-proof-of-transit
- Will OPSEC WG consider working on Proof-Of-Transit?