An Implementation of Verifiable Routing on PolKA

Henrique Coutinho Layber†, Roberta Lima Gomes†, Magnos Martinello†, Vitor B. Bonella†, Everson S. Borges‡, Rafael Guimarães†‡

†Department of Informatics, Federal University of Espírito Santo
‡Department of Informatics, Federal Institute of Education Science and Technology of Espírito Santo

Abstract

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere malum nobis opinemur. Quod idem licet transferre in voluptatem, ut postea variari voluptas distinguique possit, augeri amplificarique non possit. At etiam Athenis, ut e patre audiebam facete et urbane Stoicos irridente, statua est in quo a nobis philosophia defensa et collaudata est, cum id, quod maxime placeat, facere possimus, omnis voluptas assumenda est, omnis dolor repellendus. Temporibus autem quibusdam et.

Keywords

Verifiable Routing; Path Verification; Proof-of-transit; In-networking Programming

I Introduction

Ever since Source Routing (SR) was proposed, there has been a need to ensure that packets traverse the network along the paths selected by the source, not only for security reasons but also to ensure that the network is functioning correctly and correctly configured. This is particularly important in the context of Software-Defined Networking (SDN), where the control plane can select paths based on a variety of criteria.

In this paper, we propose a new P4[1] implementation for a new protocol layer for PolKA[2], able to do verify the actual route used for a packet. It is available on GitHub[1]. This is achieved by using a composition of hash functions on stateless core switches, each using a key to generate a digest that can be checked by the controller which knows the secrets. The controller can then verify that the packet traversed the network along the path selected by the source, ensuring that the network is functioning correctly.

II Related Works

This is an extension of PolKA, a protocol that uses stateless *Residue Number System*-based Source Routing scheme[2].

This work is just part of a complete system, PathSec[3]. PathSec also deals with accessibility, auditability, and other aspects of a fully-featured Proof of Transit (PoT) network. This works only relates to the verifiability aspect of PathSec.

[1] https://github.com/Henriquelay/polka-halfsiphash/tree/remake/mininet/polka-example

III Problem Definition

Using simpler terms, in PolKA, the route up to a protocol boundary (usually, the SDN border) is defined in s[4]. s calculates and sets the packet header with enough information for each node to calculate the next hop. Calculating each hop is done using Chinese Remainder Theorem (CRT) and the Residue Number System (RNS)[5], and is out of the scope of this paper. All paths are assumed to be both valid and correct.

Let $G = (V, E)$ be a graph representing the network topology, where V is the set of nodes (switches) and E is the set of edges (connections). Let e be the source node (SDN ingress edge) and d be the destination node (SDN egress edge). Let path P be a sequence of nodes:

$$P = (e, s_1, s_2, .., s_n, d)$$

where
P Path from e to d.
s_i Core switch i in the path.
n Number of core switches in the path.
e Ingress edge (source).
d Egress edge (destination).

The main problem we are trying to solve is to have a way to ensure if the packets are following the path defined by the source. Notably, the solution need not to list the switches traversed, but only to verify if the packet has passed through the correct path.

A solution should be able to identify if:
1. The packet has passed through the correct switches.
2. The packet has passed through the correct order of switches.
3. The packet has not passed through any switch that is not in the path.

More formally, given a sequence of switches P_e generated on the ingress e, and a sequence of switches actually traversed P_j, a solution should identify if $P_e = P_j$.

Figure 1 shows the most used topology used in the experiments.

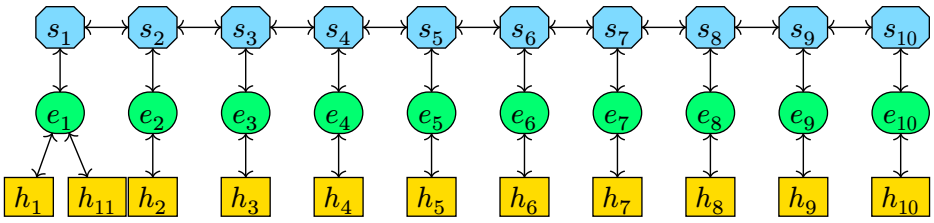

Figure 1: Topology setup

IV Solution Proposal

Since the system is stateless, using function composition is a good way to propagate errors. Function composition preserves the order-sensitive property of the path, since $f \circ g \neq g \circ f$ in a general case. Each node will execute a single function of this composition, using the previous node's output as input. By using injective functions in two variables, we can use one of the variables to have any uniquely per-node value, ensuring that the function is unique for each switch, ensuring $f_{s_1}(x) \neq f_{s_2}(x)$ for nodes s_1 and s_2. In this way:

$$f_{s_1} \circ f_{s_2} \circ f_{s_3} = f\Big(\mathrm{id}_{s_3}, f\Big(\mathrm{id}_{s_2}, f\Big(\mathrm{id}_{s_1}, x\Big)\Big)\Big)$$

f_{s_i} Function for switch s_i.

id_{s_i} Unique identifier for switch s_i.

The function f is a hash function, and the unique identifier id_{s_i} is the switch's ID. TODO exit port is also added

IV.1 Assumption

Controller that knows all IDs, and the hash function used.

IV.2 Implementation

It was implemented {{THIS WAY}}. It was validated with {{THIS}}.

This detects {{X Y Z}}

V Limitations

- Replay attack is undetectable if timing is not considered.

VI Future Work

- Rotating key for switches for detecting replay attacks (holy shit this is hard)
- Include (entrance or exit) port in hash

Bibliography

[1] P. Bosshart *et al.*, "P4: programming protocol-independent packet processors," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, pp. 87–95, Jul. 2014, doi: 10.1145/2656877.2656890.

[2] C. Dominicini *et al.*, "PolKA: Polynomial Key-based Architecture for Source Routing in Network Fabrics," in *2020 6th IEEE Conference on Network Softwarization (NetSoft)*, 2020, pp. 326–334. doi: 10.1109/NetSoft48620.2020.9165501.

[3] M. Martinello *et al.*, "PathSec: Path-Aware Secure Routing with Native Path Verification and Auditability." 2024.

[4] E. S. Borges *et al.*, "PoT-PolKA: Let the Edge Control the Proof-of-Transit in Path-Aware Networks," *IEEE Transactions on Network and Service Management*, vol. 21, no. 4, pp. 3681–3691, 2024, doi: 10.1109/TNSM.2024.3389457.

[5] C. Dominicini *et al.*, "Deploying PolKA Source Routing in P4 Switches : (Invited Paper)," in *2021 International Conference on Optical Network Design and Modeling (ONDM)*, 2021, pp. 1–3. doi: 10.23919/ONDM51796.2021.9492363.