Securing Internet Routing: Machine-Checked Proofs for ASPA Route-Leak Detection and a False-Positive Flaw in the ASRA Extension

Nobuo Otoi
BA Cybersecurity, Tokyo, Japan
otoi@ba-cybersecurity.com

October 2026 (version 1)

Abstract

Route leaks—BGP announcements that send Internet traffic through networks that should not carry it—have repeatedly caused large outages and traffic interception. ASPA (Autonomous System Provider Authorization) lets BGP speakers detect route leaks by checking an AS_PATH against RPKI-signed lists of provider ASes. The verification procedure of draft-ietf-sidrops-aspa-verification has reached working-group consensus, and its correctness properties are stated in prose (Section 5.2 and Appendix B of the draft). We transcribe Sections 5.3–5.6 and prove in Lean 4, for paths of any length, assuming that business relationships are seen consistently from both ends and that every registered ASPA lists exactly the providers of its AS: the procedure never marks a valley-free path Invalid; a Valid outcome implies the path is valley-free; Property 1 of Appendix B holds; and under full deployment the procedure never returns Unknown, so that Valid and Invalid coincide exactly with valley-free and route leak. We then analyse ASRA (Autonomous System Relationship Authorization), a proposed extension that adds fake-link detection and is stated to "fully preserve the route leak detection capabilities of ASPA". Step 4 of its downstream enhancement, in both Algorithm A and Algorithm B, marks valley-free paths Invalid: when the origin has no ASPA and the apex of the path is a lateral-peer hop, a single AS with correct ASPA and ASRA objects suffices. The smallest instance has four ASes, and in exhaustive enumeration of four-AS topologies every false positive comes from Step 4. Starting the Step 4 range at $max_up + 1$ instead of $min_up + 1$ removes the flaw; we prove in Lean that the modified step fires only on route leaks. We also record a consequence of ASPA for AS-path poisoning. Proofs and scripts are released as artifacts.

Keywords: network security; Internet routing security; BGP; route leaks; RPKI; ASPA; ASRA; formal verification; Lean 4; IETF standardization

1 Introduction

A route leak is a BGP announcement that violates the export policies implied by business relationships, usually formalised by the valley-free model of Gao and Rexford [11, 8]. ASPA [2, 1] lets a customer AS publish, as an RPKI object, the set of its provider ASes. From these objects a BGP speaker bounds the up-ramp and the down-ramp of a received AS_PATH; if the two ramps cannot meet, the route is a leak.

The draft states its guarantees in prose: the AS_PATH is valid if the combined ramp lengths reach the path length, and leaked or malformed otherwise (Section 5.2); Appendix B lists detection properties under partial deployment. An earlier formulation of the downstream algorithm was analysed by its designers, who stated two theorems and proved them by enumeration for small cases with an induction step to be described later [4]. The current formulation (Sections 5.4–5.6)

has no published proof. Because the procedure will run in router code across the Internet, and because extensions are being designed on top of it, we give machine-checked arguments.

Contributions.

1. A literal, executable transcription of Sections 5.3–5.6 of draft-ietf-sidrops-aspa-verification, cross-checked against a Gao–Rexford ground truth on all labelled topologies of up to four ASes, including non-transparent route servers (835,224 combinations of path, receiver and deployment set at $n = 4$), and against all 25 rows of the worked examples referenced by the draft [3].
2. Lean 4 proofs, for paths of any length, of no false positive, soundness of VALID, Property 1 of Appendix B, and exactness under full deployment (Section 4). The development uses core Lean only, contains no `sorry`, and does not depend on `Classical.choice`.
3. A false-positive flaw in Step 4 of the ASRA downstream enhancement [5], with a four-AS counterexample (Section 5).
4. A modified Step 4 and a Lean proof that it fires only on route leaks (Section 5.3).
5. Two observations, on AS-path poisoning and on the transparent route-server exception (Section 6).

2 ASPA verification

Let the COMPRESSED_AS_PATH be $\mathrm{AS}(N), \ldots, \mathrm{AS}(1)$, where $\mathrm{AS}(1)$ is the origin, $\mathrm{AS}(N)$ is the neighbour of the verifying AS, and $\mathrm{AS}(N+1)$ is the verifying AS. Let $\mathrm{SPAS}(x)$ be the union of the provider sets in the valid ASPAs of x, defined when x has at least one valid ASPA. The provider authorization function (Section 5.3) is

$$\mathsf{authorized}(x, y) = \begin{cases} \mathsf{nA} & \text{if } x \text{ has no valid ASPA,} \\ \mathsf{P} & \text{if } y \in \mathrm{SPAS}(x), \\ \mathsf{nP} & \text{otherwise.} \end{cases}$$

Writing $a_i = \mathsf{authorized}(\mathrm{AS}(i), \mathrm{AS}(i+1))$ and $b_j = \mathsf{authorized}(\mathrm{AS}(j), \mathrm{AS}(j-1))$, the bounds of Section 5.4 are

$$\begin{aligned} \mathit{max_up} &= \min(\{\, I \in [1, N-1] : a_I = \mathsf{nP} \,\} \cup \{N\}), \\ \mathit{min_up} &= \min(\{\, I \in [1, N-1] : a_I \neq \mathsf{P} \,\} \cup \{N\}), \\ \mathit{max_down} &= N - J_{\mathsf{nP}} + 1, \qquad \mathit{min_down} = N - J_{\neq \mathsf{P}} + 1, \end{aligned}$$

where $J_{\mathsf{nP}} = \max\{\, J \in [2, N] : b_J = \mathsf{nP} \,\}$ and $J_{\neq \mathsf{P}} = \max\{\, J \in [2, N] : b_J \neq \mathsf{P} \,\}$, and $\mathit{max_down} = N$ (resp. $\mathit{min_down} = N$) when the set is empty. For a route received from a provider (downstream, Section 5.6), after the prerequisite steps 1–3,

$$\mathrm{out}_{\downarrow} = \begin{cases} \textsc{Invalid} & \text{if } \mathit{max_up} + \mathit{max_down} < N, \\ \textsc{Unknown} & \text{else if } \mathit{min_up} + \mathit{min_down} < N, \\ \textsc{Valid} & \text{otherwise.} \end{cases}$$

For a route received from a customer, a lateral peer or a route server, and for a route received by a route server (upstream, Section 5.5), the same rule applies with $\mathit{max_down} = \mathit{min_down} = 0$; we write $\mathrm{out}_{\uparrow}$.

3 Model

Hops. A path of N ASes has $n = N - 1$ hops. Hop i ($0 \le i < n$) goes from $\mathrm{AS}(i+1)$ to $\mathrm{AS}(i+2)$ and is the triple $h_i = (\tau_i, \alpha_i, \beta_i)$, where $\tau_i \in \{\mathsf{up}, \mathsf{down}, \mathsf{peer}\}$ is its true type (up: the receiver is

the sender's provider; down: a customer; peer: a lateral peer), $\alpha_i = \mathsf{authorized}(\mathrm{AS}(i{+}1), \mathrm{AS}(i{+}2))$ and $\beta_i = \mathsf{authorized}(\mathrm{AS}(i+2), \mathrm{AS}(i+1))$.

Bounds as prefix lengths. For a predicate P and a list ℓ, let $\mathrm{lead}_P(\ell)$ be the length of the longest prefix of ℓ all of whose elements satisfy P, and let $\overleftarrow{h} = (h_{n-1}, \ldots, h_0)$. Then

$$\begin{aligned} max_up &= 1 + \mathrm{lead}_{\alpha \neq \mathsf{nP}}(h), & min_up &= 1 + \mathrm{lead}_{\alpha = \mathsf{P}}(h), \\ max_down &= 1 + \mathrm{lead}_{\beta \neq \mathsf{nP}}(\overleftarrow{h}), & min_down &= 1 + \mathrm{lead}_{\beta = \mathsf{P}}(\overleftarrow{h}). \end{aligned}$$

We checked by hand that these agree with the index-based definitions of Section 2, including the empty-set cases; the literal Python transcription agrees with them on every enumerated case; and both the Python transcription and the Lean definitions reproduce every row of the worked examples [3] (9 upstream, 10 downstream, 6 complex-relationship rows), on all reported bounds and on the outcome. In Lean each row is checked by `decide`.

Definition 1 (Valley-free)**.** A sequence of hop types $\tau_0, \ldots, \tau_{m-1}$ is *valley-free*, written $\mathrm{VF}(\tau)$, if

$$\forall\, i < j < m: \quad \tau_i \neq \mathsf{up} \implies \tau_j = \mathsf{down},$$

that is, it matches $\mathsf{up}^*\, \mathsf{peer}^?\, \mathsf{down}^*$. The ground truth for a received route is VF of the full sequence including the hop into the verifying AS, which is down for downstream verification and up or peer for upstream verification.

Lemma 1 (`vf_snoc_down`, `vf_snoc_nondown`)**.** $\mathrm{VF}(\tau \cdot \mathsf{down}) \iff \mathrm{VF}(\tau)$*, and for* $t \neq \mathsf{down}$*,* $\mathrm{VF}(\tau \cdot t) \iff \forall i,\ \tau_i = \mathsf{up}$.

Assumptions.

(A1) Consistency: y is a provider of x if and only if x is a customer of y.

(A2) ASPA correctness: if x has a valid ASPA, then $y \in \mathrm{SPAS}(x)$ if and only if y is a provider of x.

From (A1) and (A2) every hop satisfies (Lean: `hop_correct`)

$$\tau_i = \mathsf{up} \Rightarrow \alpha_i \neq \mathsf{nP}, \qquad \alpha_i = \mathsf{P} \Rightarrow \tau_i = \mathsf{up}, \qquad \tau_i = \mathsf{down} \Rightarrow \beta_i \neq \mathsf{nP}, \qquad \beta_i = \mathsf{P} \Rightarrow \tau_i = \mathsf{down}. \tag{1}$$

Not modelled in Lean. Steps 1–3 of the procedures (empty path, neighbour-ASN check, AS_SET) are assumed to have passed. Route servers are covered only by the enumeration. Complex relationships are excluded by (A1). The theorems do not require consecutive ASes to be neighbours, so they hold in particular for every real path.

4 Machine-checked properties of ASPA

All results are in `Aspa.lean` (Lean 4.33.1, core only, about 550 lines including the 25 example checks). `#print axioms` reports at most `propext` and `Quot.sound` for every theorem. Throughout, every hop satisfies (1).

Theorem 1 (No false positive; `down_no_false_positive`, `up_no_false_positive`)**.** *If* $\mathrm{VF}(\tau)$ *then* $\mathrm{out}_\downarrow \neq$ Invalid*. If* $\tau_i = \mathsf{up}$ *for all* i*, then* $\mathrm{out}_\uparrow \neq$ Invalid*.*

Proof sketch. Let $k = \mathrm{lead}_{\alpha \neq \mathsf{nP}}(h)$. If $k = n$ then $max_up = N$. Otherwise $\alpha_k = \mathsf{nP}$, so $\tau_k \neq \mathsf{up}$ by (1), and $\mathrm{VF}(\tau)$ gives $\tau_j = \mathsf{down}$ for all $j > k$. Hence $\beta_j \neq \mathsf{nP}$ for $j > k$, so $max_down \geq n - k$, and $max_up + max_down \geq (k+1) + (n-k) = N$. □

Theorem 2 (VALID is sound; `down_valid_sound`, `up_valid_sound`)**.** *If* $\text{out}_\downarrow = \text{VALID}$ *then* $\text{VF}(\tau)$*. If* $\text{out}_\uparrow = \text{VALID}$ *then* $\tau_i = \mathsf{up}$ *for all* i*.*

Proof sketch. Let $k_1 = \text{lead}_{\alpha=\mathsf{P}}(h)$ and $k_2 = \text{lead}_{\beta=\mathsf{P}}(\overleftarrow{h})$. By (1), $\tau_i = \mathsf{up}$ for $i < k_1$ and $\tau_j = \mathsf{down}$ for $j \geq n - k_2$. VALID gives $k_1 + k_2 \geq n - 1$. If $i < j$ and $\tau_i \neq \mathsf{up}$, then $i \geq k_1$, so $j \geq k_1 + 1 \geq n - k_2$ and $\tau_j = \mathsf{down}$. □

Theorem 3 (Appendix B, Property 1; `up_property1`)**.** *If some hop* i *has* $\tau_i \neq \mathsf{up}$ *and* $\alpha_i \neq \mathsf{nA}$ *(its sender has an ASPA and sent the route to a customer or a lateral peer), then* $\text{out}_\uparrow = \text{INVALID}$*, whatever the deployment status of the other ASes.*

This is Property 1 with "AS B doing ASPA" read as AS B running the upstream procedure; only the ASPA of AS A is needed.

Theorem 4 (Full deployment; `down_full_no_unknown`, `up_full_no_unknown`, `down_full_exact`)**.** *If* $\alpha_i \neq \mathsf{nA}$ *and* $\beta_i \neq \mathsf{nA}$ *for all* i*, the outcome is never* UNKNOWN*. Consequently* $\text{out}_\downarrow = \text{VALID} \iff \text{VF}(\tau)$ *and* $\text{out}_\downarrow = \text{INVALID} \iff \neg\text{VF}(\tau)$*.*

`aspa_downstream` and `aspa_upstream` restate Theorems 1 and 2 for a topology and an ASPA table satisfying (A1) and (A2), with the ground truth evaluated on the full hop sequence including the hop into the verifying AS.

5 A false positive in the ASRA downstream enhancement

5.1 The step

draft-sriram-sidrops-asra-verification-05 [5] refines the downstream outcome using ASRA objects, which list an AS's customers and lateral peers. In Algorithm A, Step 2 returns early if $min_up = N$ or $min_up + min_down > N$, Step 3 checks for fake links, and Step 4 reads:

> Else, if for any i value from $(min_up + 1)$ to $(N - min_down)$, if AS(i) has ASPA and ASRA asserting that AS($i + 1$) is a lateral peer (and not a Provider or Customer), then change the outcome to Invalid.

Algorithm B uses the range $(min_up + 1)$ to $(N - 1)$ and has no second early-exit condition. The draft explains that Step 4 compensates for missing ASPA information about the next AS.

5.2 Counterexample

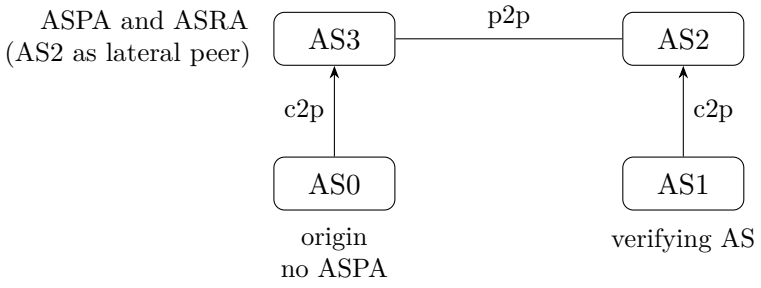


Figure 1: The four-AS counterexample. Arrows point from customer to provider.

In Figure 1, AS1 receives from its provider AS2 the path with AS(1) = AS0, AS(2) = AS3, AS(3) = AS2 ($N = 3$). Only AS3 registers objects: a correct ASPA (its providers, which include

neither AS0 nor AS2) and a correct ASRA listing AS2 as a lateral peer. The path is valley-free (up, peer, then down into AS1). The bounds are

$$\begin{aligned} min_up &= 1 && (\text{AS0 has no ASPA, so } \mathsf{authorized}(\text{AS0}, \text{AS3}) = \mathsf{nA}),\\ max_up &= 2 && (\mathsf{authorized}(\text{AS3}, \text{AS2}) = \mathsf{nP}),\\ min_down &= 1 && (\text{AS2 has no ASPA, so } \mathsf{authorized}(\text{AS2}, \text{AS3}) = \mathsf{nA}),\\ max_down &= 2 && (\mathsf{authorized}(\text{AS3}, \text{AS2}) = \mathsf{nP}), \end{aligned}$$

so $max_up + max_down = 4 \geq N$ and $min_up + min_down = 2 < N$: ASPA alone yields UNKNOWN. In both algorithms, Step 2 does not exit ($1 \neq 3$ and $1 + 1 \not> 3$); Step 3 detects no fake link (AS0 has no ASPA, and AS3's ASRA lists AS2); Step 4 examines $i = 2$, where AS(2) = AS3 asserts that AS(3) = AS2 is a lateral peer, and returns INVALID.

Cause. *min_up* stops at AS0 only because AS0 has no ASPA. The true up-ramp continues to AS3, the legitimate apex. Treating a lateral hop beyond *min_up* as a leak is justified only once the up-ramp is known to have ended, that is, at an nP pair. The configuration is common under partial deployment (a stub origin without ASPA, its early-adopting transit provider peering laterally, a customer cone below the peer), and a single adopting AS suffices.

Verification. We re-derived the counterexample from the specification text as above, for both algorithms. In Lean, `cex_valley_free`, `cex_aspa_unknown` and `cex_step4_fires` establish, by kernel evaluation and without axioms, that the path is valley-free, that ASPA alone yields UNKNOWN, and that the Step 4 condition holds over Algorithm B's range. Algorithm A's range is a sub-range ($N - min_down \leq N - 1$) and contains $i = 2$ here. Steps 2–3 are not modelled in Lean; that they pass is the hand derivation above, confirmed by the Python model. Exhaustively, over all labelled four-AS topologies, all simple paths received from a provider and all subsets of ASPA and ASRA adopters (ASRA adopters also have an ASPA; all objects correct), each algorithm yields 4,608 labelled false-positive cases, all from Step 4; Step 3 yields none, as expected, since with correct registrations every real link appears in an ASPA, an ASRA-C or an ASRA-LP.

5.3 A modified Step 4

We propose to start the Step 4 range at $max_up + 1$ instead of $min_up + 1$.

Theorem 5 (`step4_fixed_sound`)**.** *Assume every hop satisfies* (1) *and every asserted lateral peer is a true lateral peer (`LatCorrect`). If the modified Step 4 fires, then* $\neg\mathrm{VF}(\tau)$.

Proof sketch. Let $k = \mathrm{lead}_{\alpha \neq \mathsf{nP}}(h)$, so $max_up = k + 1$, and suppose the step fires at hop i with $i > k$ and $\tau_i = \mathsf{peer}$. Then $\alpha_k = \mathsf{nP}$, hence $\tau_k \neq \mathsf{up}$, and VF would force $\tau_i = \mathsf{down}$. □

The theorem is proved for Algorithm B's range and hence holds for Algorithm A's sub-range. ASRA correctness enters only as the assumption `LatCorrect`; we do not derive it from an ASRA table.

On the same enumeration the modified step has no false positives and still flags 6,144 leaks that ASPA alone does not mark INVALID; the original Step 4 flags 15,360. Every one of the 9,216 cases no longer flagged has the same shape: the lateral hop is the first nP pair and the hop before it is nA. There the registered objects cannot distinguish a leak from the valley-free path of Figure 1, so flagging such cases necessarily produces false positives as well. We do not propose extending the up-ramp with the receiver's ASRA-C: Section 4.1 of the ASRA draft excludes that direction, since a forged ASRA-C would let a leaker evade detection.

5.4 Editorial

ASRA-05 cites Sections 5.1 (COMPRESSED_AS_PATH), 5.3 (ramp parameters), 5.4 (upstream) and 5.5 (downstream) of the ASPA verification draft; in revisions -28 and -29 these are Sections 5.2, 5.4, 5.5 and 5.6.

6 Further observations

AS-path poisoning. Suppose an origin O with an ASPA announces the poisoned path $[O, X, O]$. Since $\mathsf{authorized}(O, X) = \mathsf{nP}$, every AS that verifies the path computes Invalid: O's providers (upstream) and also ASes further downstream, for example a customer of P receiving $[O, X, O, P]$. The only exception in our model is when X is one of O's registered providers and has no ASPA itself, giving Unknown; if O has no ASPA the outcome is Unknown. Thus an ASPA-registered origin cannot poison arbitrary ASes. The draft does not state this consequence, and in the SIDROPS archive (7,290 messages as of 2026-10-28, searched through the IETF IMAP service) poisoning is mentioned only in passing and in other contexts. Since poisoning is used in measurement and traffic engineering, a sentence in the deployment considerations may be useful.

Transparent route-server exception. Section 5.1 of the draft notes both that a leaker may strip its own ASN and that routes from a transparent IX route server are a standard exception to the neighbour-ASN check. Together, a client that strips its own ASN and re-announces a provider-learned route through a transparent route server reaches other clients with a Valid outcome. This requires deliberate path manipulation; the ASRA-TIX registrations proposed in Section 3 of ASRA-05 target this case.

7 Related work

Sriram and Heitz [4] found that the downstream algorithm of revision -06 misclassified some valid paths as unknown, proposed a corrected algorithm based on a (K, L) ramp representation, and stated two theorems characterising the three outcomes, proved by enumeration for small cases with an induction step to be described later. Their formulation precedes the current max/min formulation; our proofs are mechanised, cover the current text, and also cover upstream verification and Appendix B. The SIDROPS list and draft-geng-sidrops-aspa-analysis [6] discuss ASPA's limitations informally, and Farina et al. applied ASPA verification to collected BGP data [7]. Route-leak detection and mitigation were defined in RFC 7908 [8] and [10], and BGP Roles with the OTC attribute in RFC 9234 [9]. On the formal-methods side, routing algebras [12, 13], the FSR toolkit [14] and a mechanised convergence proof for policy-rich distance-vector protocols [15] address convergence and policy safety rather than the ASPA procedure. To our knowledge no mechanised proof of ASPA's verification properties has been published. A search of the SIDROPS archive (all messages mentioning both ASRA and lateral peers, and all messages since June 2026 mentioning ASRA or false positives) found no report of the ASRA Step 4 issue.

8 Limitations

The ground truth is the standard valley-free model; complex relationships, sibling ASes and per-prefix policies are excluded. Route servers are verified only by bounded enumeration. The correspondence between the draft's prose and our definitions is checked by hand, by testing and against all 25 rows of the worked examples, but it is not proved. ASRA is an individual draft and may change; our analysis refers to revision -05 as published on 2026-07-06. Datatracker states were checked on 2026-10-28 (ASPA verification: "WG Consensus: Waiting for Write-Up").

Artifacts

`lean/Aspa.lean` (Lean 4 v4.33.1, core only; `lean Aspa.lean` prints only the `#print axioms` lines) and `model/` (Python 3: literal transcriptions `model.py`, `asra.py`; checkers `check.py`, `check_asra.py`, `fixcheck.py`, `lost.py`, `examples.py`). Repository: `https://github.com/otoinobuo/aspa-verify`. Archived with this paper at `https://doi.org/10.5281/zenodo.23734450`.

Acknowledgements

AI tools produced the proofs, code and text under the author's direction; the author chose the questions and claims and checked all results.

References

[1] A. Azimov, E. Bogomazov, R. Bush, K. Patel, J. Snijders, K. Sriram, "BGP AS_PATH Verification Based on Autonomous System Provider Authorization (ASPA) Objects", draft-ietf-sidrops-aspa-verification-29, October 2026.

[2] A. Azimov et al., "A Profile for Autonomous System Provider Authorization", draft-ietf-sidrops-aspa-profile.

[3] K. Sriram, O. Borchert, M. Matejka, "ASPA-based AS path verification examples", October 2026, `https://github.com/ksriram25/IETF/blob/main/ASPA_path_verification_examples.pdf`.

[4] K. Sriram, J. Heitz, "Insights into the accuracy of algorithms for ASPA based route leak detection", SIDROPS WG, IETF 110, March 2021, `https://github.com/ksriram25/IETF/blob/main/ASPA-alg-accuracy-v2.pdf`.

[5] K. Sriram et al., "Autonomous System Relationship Authorization (ASRA) as an Extension to ASPA for Enhanced AS Path Verification", draft-sriram-sidrops-asra-verification-05, July 2026.

[6] N. Geng et al., "An Analysis of ASPA-based AS_PATH Verification", draft-geng-sidrops-aspa-analysis-05, August 2026.

[7] M. Farina, M. Trevisan, A. Bartoli, "A First Look at ASPA: Deployment, Route Leaks, and AS Relationships", ACM Internet Measurement Conference (IMC), 2026, doi:10.1145/3777912.3839812.

[8] K. Sriram, D. Montgomery, D. McPherson, E. Osterweil, B. Dickson, "Problem Definition and Classification of BGP Route Leaks", RFC 7908, 2016.

[9] A. Azimov, E. Bogomazov, R. Bush, K. Patel, K. Sriram, "Route Leak Prevention and Detection Using Roles in UPDATE and OPEN Messages", RFC 9234, 2022.

[10] K. Sriram et al., "Methods for Detection and Mitigation of BGP Route Leaks", draft-ietf-idr-route-leak-detection-mitigation.

[11] L. Gao, J. Rexford, "Stable Internet routing without global coordination", IEEE/ACM Transactions on Networking 9(6), 2001, doi:10.1109/90.974523.

[12] J. L. Sobrinho, "An algebraic theory of dynamic network routing", IEEE/ACM Transactions on Networking 13(5), 2005, doi:10.1109/TNET.2005.857111.

[13] T. G. Griffin, J. L. Sobrinho, "Metarouting", ACM SIGCOMM, 2005, doi:10.1145/1090191.1080094.

[14] A. Wang, L. Jia, W. Zhou, Y. Ren, B. T. Loo, J. Rexford, V. Nigam, A. Scedrov, C. Talcott, "FSR: Formal Analysis and Implementation Toolkit for Safe Interdomain Routing", IEEE/ACM Transactions on Networking 20(6), 2012, doi:10.1109/TNET.2012.2187924.

[15] M. L. Daggitt, A. J. T. Gurney, T. G. Griffin, "Asynchronous convergence of policy-rich distributed Bellman-Ford routing protocols", ACM SIGCOMM, 2018, doi:10.1145/3230543.3230561.