Assessing the RPKI Validator Ecosystem

Paul Henry Friedemann*, Nils Rodday*†, Gabi Dreo Rodosek*

*Research Institute CODE, Universität der Bundeswehr München,
†University of Twente

***Abstract*—In this work we compare seven currently available Resource Public Key Infrastructure (RPKI) validators regarding their ease of installation, performance, consistency in results, code quality, applicability, and feature-richness. We weigh the different characteristics and rank each validator based on the scores it received during our tests. We find Routinator to perform the best and recommend using this RPKI validator in a production environment. Additionally, we uncover inconsistencies in the validation results between different validators and investigate possible causes. Moreover, we also look at the RPKI repository infrastructure and discuss recent outages.**

***Index Terms*—BGP, RPKI, ROV, RPKI Validator**

I. Introduction

The Border Gateway Protocol (BGP) is the de-facto standard for inter-domain routing although it is known to be insecure [1], [2]. A major issue with BGP is that it is based on trust and as such lacks proof of address ownership. Any Autonomous System (AS) is allowed to announce any prefix range towards its peering partners, effectively controlling and attracting incoming traffic. However, this makes it very easy for AS operators with malicious intent to announce prefix ranges that have not been assigned to them by the Internet Assigned Numbers Authority (IANA). Moreover, even without a malicious intent, the sheer complexity of network management often leads to misconfigurations and therefore route leaks. An unintentional act of announcing prefixes ranges to peers and effectively hijacking prefixes. A prominent and recent example is the route leak by Vodafone India Ltd. in April 2021. Roughly 30.000 prefixes were announced by mistake, leading to a 13 fold increase of incoming traffic and unavailability of the original services [3].

Many features have been implemented as workarounds over the years that try to fix the aforementioned problems such as IRR objects, BGP filters, etc. Moreover, the scientific community proposed many extensions that try to tackle the problem: Secure BGP (S-BGP) [4], Secure Origin BGP (soBGP) [5], Interdomain Route Validation (IRV) [6] and Pretty secure BGP (psBGP) [7]. Due to their complexity or missing incentives for network operators, adoption never happened.

In order to establish proof of address ownership and solve the aforementioned problem, the Secure Inter-Domain Routing (SIDR) working group [8] of the Internet Engineering Task Force (IETF) has put effort into two security features: Border Gateway Protocol Security (BGPsec) [9] and RPKI [10]. BGPsec provides path validation and requires every AS en route to participate, while the RPKI provides origin authentication and has a lower entry barrier. RPKI deployment is picking up [11]–[13].

In order to validate RPKI Route Origin Authorization (ROA) objects within an AS, the operator will need to install at least one RPKI validator. There are currently several validators available: RPSTIR2 [14], OctoRPKI [15], Routinator 3000 [16], FORT-Validator [17], rpki-client [18], and rpki-prover [19]. The RPKI-Validator 3 has been discontinued [20]. Rcynic [21] seems to be discontinued as well as there have not been any updates since 2018. They come in different flavors, e.g. programming language, performance, stability, etc.

Since the RPKI is a fairly new technology which is continuously improved, there has been not much work on RPKI validators yet. Kristoff et al. [22] explore RPKI validators default parameters to obtain hints on their fetching intervals. In [23]–[25] one finds additional resources on how to install RPKI validators.

Contributions. This work focuses on the comparison of the different validators to give recommendations on which validator would be best to use in a production environment. To this end we:

1) Develop a metric to assess the different validators.
2) Compare the validators according to the predefined metric.
3) Give recommendations which validator is to be preferred in a production environment.
4) Discuss recent RPKI infrastructure outages.

The remainder of this paper is structured as follows: Section II introduces the basic functionality of the RPKI while Section III gives a brief overview of the candidates. Sections IV explains our experiment setup and elaborates on the metric that we used to rate the different validators. Our results are presented in Section V together with details on each assessed validator to point out their advantages and disadvantages. Section VI details RPKI infrastructure outages.

II. RPKI Architecture

The basic architecture of the RPKI is detailed in RFC 6480 [10]. However, many more RFCs have been published to detail different parts of the process. An overview can be found in [26].

Issuance of ROAs. Resources (blocks of IP addresses) are assigned by the IANA to the five Regional Internet Registries

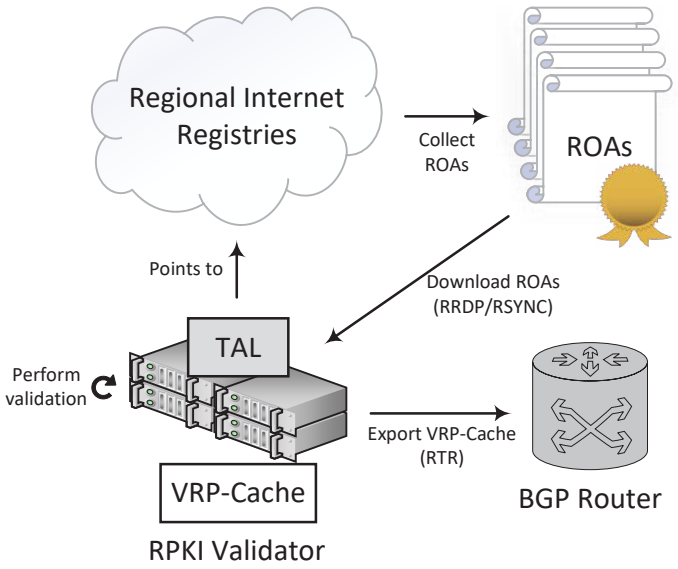

Figure 1. Simplified validation process. The RPKI validator fetches the ROAs and cryptographically validates them. It exports the VRP-cache to the BGP routers for decision making.

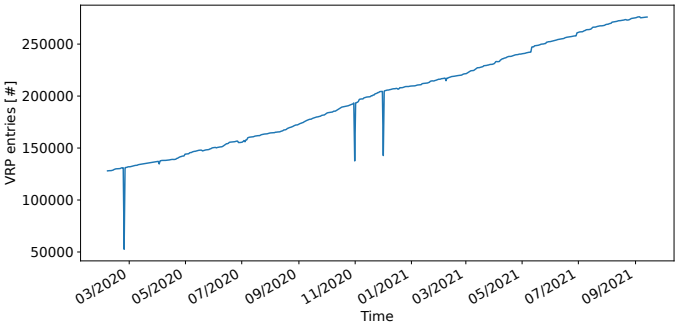

Figure 2. VRP entries from March 2020 - September 2021. We observe a steady increase of VRP entries. The drops relate to outages in the validator software producing the underlying data [30].

(RIRs). They run the RPKI Trust Anchors (TAs) and delegate the received resources via the issuance of certificates to Local Internet Registries (LIRs) which in turn can forward them to Internet Service Providers (ISPs). The process can be repeated amongst ASes until an AS that wishes to announce these resources has obtained them. Typically, the leaf AS will issue a ROA, which cryptographically binds these resources to an AS number, together with a max-length attribute.

Validation of ROAs. Once ROAs are published by the owners of that address space, any AS participating in the inter-domain routing infrastructure can fetch and validate them to secure its routing decisions. Figure 1 illustrates the workflow of an RPKI validation process. A validator performs a validation by starting to locate the trust anchor via the Trust Anchor Locator (TAL). ROAs are published at different publication points and the validator collects all available objects. Rsync [27] or RPKI Repository Delta Protocol (RRDP) [28] can be used for that purpose. Once obtained, validation is performed by checking the cryptographic validity of each ROA. This process can be performed out-of-band, which has an advantage over the in-band processing of BGPSec as it does not put additional strain on the BGP routers which make routing decisions. An additional advantage is that not every AS in the AS path for the specific BGP announcement is required to participate for the RPKI to function properly [29]. The BGP router receives the Validated ROA Payload (VRP) and every announcement can be categorized as valid, invalid, or not found. Any participating BGP router is able to detect BGP hijacks based on the exported information and routing decisions can be made accordingly.

Recent publications show that RPKI deployment is constantly picking up [11], [31], [32]. Figure 2 confirms these findings by showing an increase in the amount of VRP entries that can be found as validator output from March 2020 - September 2021. More ROAs are created over time and therefore more VRP entries are generated once validation has been performed.

III. Validators

In this section, we are going to provide basic information about the chosen validators. All of them are publicly hosted on GitHub and released under an open source license, such as Berkeley Software Distribution (BSD), Massachusetts Institute of Technology (MIT), or Internet Systems Consortium (ISC). Figure 3 illustrates the publication dates of the respective validators. The first validator appeared in 2011 while the second one was developed in 2015. From 2018 onwards RPKI development picked up and more validators were added to the ecosystem. Since the RPKI Validator was discontinued by RIPE NCC in July 2021, we skip additional information for this validator, but still report on the results for completeness.

RPSTIR2. The first version of RPSTIR [33] was written in C and published by BBN Technologies in 2015. According to the analysis of commits, the maintenance and further development of the project was transferred to ZDNS around 2017. A new version, RPSTIR2 was released in 2020 and written in Go, which implements the majority of the RPKI standards [14]. It is used in the backend of RPKIVIZ [34], an RPKI visualization tool [35].

OctoRPKI. Cloudflare has developed its own RPKI validator software, OctoRPKI [15]. It is written in Go and was released in 2019. This project also covers the RPKI to Router (RTR) server GoRTR [36].

Routinator 3000. Routinator [37] is written in Rust and was released by NLnet Labs in 2019. It supports in its release 0.10.1 a standalone version of the user interface. NLnet Labs also contributes to the RPKI ecosystem with other developments such as Krill, a software product that allows for running a RPKI Certificate Authority (CA) [38]. Support is provided via Discord [39] and a mailinglist [40]. Additionally to the integrated RTR server, NLnet Labs provides a standalone version to serve as a proxy for larger networks [41].

FORT-Validator. The FORT project [42], a secure routing initiative from the RIR Latin America and Caribbean Network

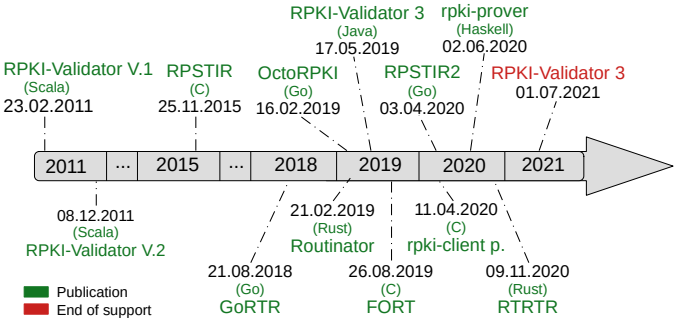

Figure 3. Timeline showing major milestones of validators and separate RTR servers.

Information Centre (LACNIC) and NIC.MX from Mexico, developed its RPKI validator in C and released it 2019.

rpki-client. The rpki-client is maintained by the OpenBSD project [43]. It was released in June 2019 made available for other operating systems in November 2020. This version was taken as a reference point for our evaluations. The validator focuses on usability with simple source code that provides the basic functionality of an RPKI validator.

rpki-prover. The rpki-prover [19] is a software project of Mikhail Puzanov, who was the main developer of RIPE NCC's RPKI Validator. rpki-prover is under development since June 2019 and written in Haskell. It was relaeased in February 2020.

IV. Methodology

Experiment setup. As simultaneous execution of seven validators is crucial for our evaluations, we had to choose between a virtualized environment or running the validator software directly on dedicated hardware. While virtualization seems appropriate for splitting resources and achieving simultaneous execution, we cannot control the scheduling mechanism of the host providing the virtualization and were therefore concerned that each VM would not have the exact same resources available. To avoid this issue we decided to install the validators on dedicated hardware.

Our evaluation platform consists of seven Raspberry Pis 4B with 4GB of RAM. All of them are connected to an uplink serving 90 Mbit/s via a Cisco Catalyst 2960-S switch. Raspberry Pis are relatively cheap and the Operating System (OS) is easy to replicate with microSD cards. In order to mitigate the bottleneck of the microSD card compared to the speed of common hard drives or Solid State Discs (SSDs) we chose high performance microSDXC cards with 160MB/s read and 60MB/S write speed. As a result, all measurements are executed under the same conditions. Raspberry Pi OS lite from 2020-08-20, the former Raspbian, is used as OS [44]. Since for some validators 64-bit OSs are explicitly recommended, we followed this guideline. A common base installation was created on one device and then distributed to all others before the validators were installed individually. Additionally, we used a virtualized platform on a powerful server equipped with 112 cores and 700GB of RAM for long-term tests.

Evaluation criteria. In order to be able to evaluate the validators as objectively as possible, they are evaluated on the basis of predefined evaluation criteria. An overview of these, sorted into six categories, can be found in Table I of the evaluation results. Each criterion is given a score from one to five and weighed accordingly. The weights are reaching from one to three. The higher the validator's score at the end, the better it performs in the tests. Assignment of scores will always remain subjective to some extent in studies such as this. However, we took care to reduce the introduced bias as much as possible by using domain-knowledge and assigning scores in discussion with peers. Most of the evaluation criteria used are straight-forward and therefore not discussed in more detail, but some require additional explanations:

For *CPU utilization/time*, the ratio of runtime to CPU usage was calculated in order to make different runs with different instances comparable. A run includes all parts of the validation process, e.g. obtaining the required ROAs from the publication points as well as performing the cryptographic operation to validate each ROA for its correctness. We normalized the average CPU utilization to 15 minutes as a fixed period for the duration of the validation with Equation 1. The lower the resulting CPU_{norm}, the more efficiently the software works.

$$CPU_{norm} = \frac{CPU_{average} * duration}{15min} \quad (1)$$

Since the RPKI is a security add-on for BGP, it is crucial that all validators perform reliably the same operations and provide the same results at the end of the validation process. We compare the resulting VRP-cache of each validator and check for differences, labeled as *Differences between validators*. Since there is no proper ground-truth available as to how many VRP-entries should be generated, we compared all validators and discuss outliers in the following section. Moreover, we check *Cache vs. Fetch Differences*. All validators fetch the entire repository during the first run and only fetch deltas compared to the existing data during the following runs, as a measure of traffic reduction. In this category we compare the consistency of results during both validation processes with the same validator. One validation run from a clean slate and the other one using a cache and continuous polling. For a production environment, it is worth mentioning that this criterion does not have major implications for an AS during long-term Route Origin Validation (ROV) execution. They mainly affect the beginning of the use of a validator with an empty cache. After the first validation runs, the effects are no longer seen in practice.

V. Results

During our evaluation Routinator performs the best, with an overall score of 250/270 points. RPSTIR2 ranks by far as the lowest candidate with a score of 104/270 points. Since there

Table I
COMPARISON OF ALL VALIDATORS. THE HIGHEST SCORES ACHIEVED IN EACH CASE ARE HIGHLIGHTED. OVERALL, ROUTINATOR RANKS THE HIGHEST.

	Weight	RPKI-Val.	RPSTIR2	OctoRPKI	Routinator	FORT-Val.	rpki-client	rpki-prover
Installation								
Quality of documentation	3	4	2	4	5	4	2	4
Installation possibilities	2	4	2	5	5	5	4	3
Installation steps quantity	2	5	1	4	4	5	4	5
Installation duration	1	3	2	5	5	5	3	5
Dependencies	1	3	1	4	5	4	2	5
ARM-Installation	2	5	1	4	5	5	5	1
Intermediate ranking	55	46	17	47	**53**	51	37	40
Performance								
Validation runtime	3	4	1	2	5	4	4	5
CPU utilization/time	3	2	1	5	5	3	3	5
Network utilization	3	5	3	5	5	1	3	5
Max. RAM consumption	3	3	1	2	4	5	5	4
Average RAM consumption	2	3	2	3	3	5	5	3
Max. system memory increase	1	3	1	5	4	3	5	4
System drive stress	1	5	1	5	2	3	3	5
Intermediate ranking	80	56	24	58	69	55	63	**72**
Validation Results								
Differences between validators	3	5	1	5	5	5	5	3
Cache vs Fetch Differences	2	5	2	5	3	5	4	5
Intermediate ranking	25	**25**	7	**25**	21	**25**	23	19
Code								
Update freq. + code changes	1	4	2	3	5	4	5	4
Support	1	3	2	3	5	4	5	5
LoC (Complexity)	1	2	3	5	4	2	3	5
Intermediate ranking	15	9	7	11	**14**	10	13	**14**
Applicability								
Update effort	3	3	5	4	5	4	4	5
Configurability	2	3	2	3	5	5	2	2
Provision of summaries	1	5	2	3	5	1	3	4
Intermediate ranking	30	20	21	21	**30**	23	19	23
Functionality								
MAN-Page and help	3	1	2	3	5	5	4	3
SLURM support	2	5	2	5	5	5	1	1
Logging capabilities	2	3	2	3	5	5	3	2
User interface	2	5	1	1	4	1	1	3
Single and server execution	2	2	2	5	5	5	1	1
RTR server integration	2	4	4	4	5	5	1	5
Intermediate ranking	65	41	28	45	**63**	57	26	33
Overall ranking	270	197	104	207	**250**	221	181	201
Rank		5	7	3	**1**	2	6	4

is also quite a gap towards all other candidates, we observe a significant quality difference between RPSTIR2 and the remaining validators. A summary of the results for all evaluation criteria can be found in Table I. The following paragraphs highlight individual findings for the different validators.

RPSTIR2. The validator achieves the lowest score with 104/270 points and obtains rank no. 7.

Firstly, this validator achieves a low score as it has quite a complicated and error-prone installation process. There do not exist precompiled binaries, which makes it harder for an operator to use this validator. Instead, RPSTIR2 and a specifically required Openssl version have to be build from source. The build process was erroneous due to dependency issues, which got resolved by the developers after direct interaction via email. MySQL 8 has to be installed separately. Configuration requires trial and error, as the documentation is not mature enough yet. The build process is broken again since May 2021.

Secondly, the validator raises questions regarding the basic

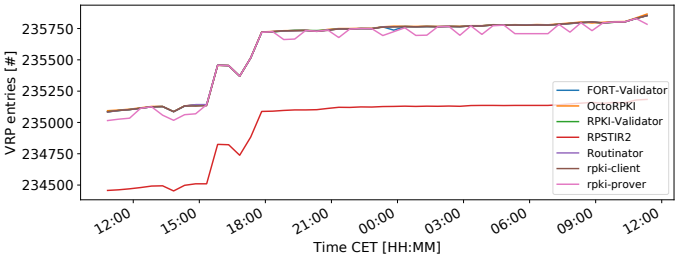

Figure 4. Comparison of VRP results of all validators in 24 hours. We observe that most validators obtain the same results while RPSTIR2 reports roughly 600 entries less.

functionality, the validation of ROA payload. Figure 4 shows that the progression curve is parallel to those of the other validators. However, a deviation of about 600 VRP entries can be seen. More detailed analysis showed that the difference in the data is over 1000 VRP entries (as the figure displays the accumulated amount and some entries might be missing while others are present in addition). The results of all other validators are roughly the same, with a few minor deviations. Our investigation into this issue showed that RPSTIR2 is using the same protocol of the RIR for the entire trust chain. If one publication point does not offer the previously established default (RRDP), a fall back to rsync is not performed, therefore ROAs will be missing and consequently less VRP entries will be generated. This is a major drawback. The software also creates different VRP entries when starting from scratch compared to an instance that has already performed several validations (*Catch vs. Fetch Differences*). We encountered 168 different VRP entries after about 20 hours and nearly 40 validations.

Thirdly, the validator has a remarkably low performance. This is particularly evident on Raspberry Pis, where validation takes between 25 and 90 minutes. Additionally, we also reproduced this finding on servers with state-of-the-art hardware. We suspect that the use of a relational database and the rapidly growing number of objects to be validated causes performance issues.

OctoRPKI. This software performs with 207/270 points exceedingly well in our evaluation. The installation is fast and can be done on a wide range of systems. Since Debian 11 was released, it can be installed from the native software repositories in the stable channel. The main criticism is that since December 2020 it is not clear where the project is headed, as the main developer left Cloudflare [45]. In August 2021 there has been activity on Github and a new version was released. However, it is important that the VRP entries created are consistent with other validators and are stable, which is the case for this validator.

Routinator 3000. Routinator is the highest scoring validator with 250/270 points. Due to the continuous support and implementation of new features by the dedicated team of NLnet Labs, the score of the last three categories is the highest. The installation is very simple and well documented. The performance is excellent, even on the Raspberry Pi with

low hardware specifications. It is considered a lightweight application with minimum 1GB RAM and 1GB free memory. The initial validation on the Raspberry Pi takes between three and six minutes. According to user feedback, there were brief problems with RAM consumption in version 0.9.0 [46]. The developers promptly informed the community [47] and solved the problem shortly after with version 0.10.0.

However, we observed problems when running the validator for the first time. In our tests, we found that the results were not quite the same as when the validator was run repeatedly. We attribute this to strict timeouts and too slow repositories. If a few validations have already been performed, the generation of the VRP entries is stable.

FORT-Validator. The validator achieves the second rank with 221/270 points. The installation and use is simple. FORT validator is already included in the Debian 10 repositories. In addition, other installation options are given. Overall, the performance is satisfactory. The only surprising thing was the high network load. Under the same conditions FORT validator created traffic of about 5968 MiB within 24 hours whereas Routinator only needed 376 MiB and rpki-client consumed 1043 MiB. The results are very stable and extensive code support is provided. A drawback is that according to the documentation, maintenance of the project will be limited to fixing critical bugs until the end of 2021 [17]. There will be no feature development until further notice. Usability is good. FORT validator offers all features with the exception of a web-based user interface.

rpki-client. The validator scores 181/270 points in the evaluation. This is mainly due to the fact that it provides a small optional feature set. It does not stand out in any other category with its ratings comparable to the other candidates. In terms of performance, rpki-client reaches the third place and its validation results are also reliable. The OpenBSD project announced to develop this validator in order to implement the core tasks of a validator reliably and easily. rpki-client must be extended with an external RTR server.

rpki-prover. The validator ranks on place no. 4 with 201/270 points. The performance of the validator is great. Since the first release in June 2021, the installation has also become much easier, as binaries are provided.

Some inconsistencies were observed during the *Differences between validators* category, as shown in Figure 4. At some instances the VRP entries are short of 60 compared to the other validators. These deviations are due to a lack of fallback to rsync and an unstable RRDP connection. The problem of missing fallback does also affect RPSTIR2 and is not solely present in this validator instance. A pull request on GitHub deals with this problem and might mitigate it [48]. Another point of criticism is the low number of extra functionality, which is planned to be improved by the developer in the future, according to the feature list on GitHub.

VI. Recent RPKI infrastructure outages

Since all validators rely on the availability of the RPKI infrastructure in order to fetch RPKI information, e.g. CA certificates, Certificate Revocation Lists (CRLs), Manifests, and ROAs, we look at some outages from the past and their impact to get an idea about the resiliency of the system.

Within 2020 several outages took place [49]. A simple disk quota limit led to the inability to write new RPKI objects to the rsync servers of RIPE NCC. It did not trigger monitoring systems and was therefore only discovered as the CRL expired, since it could not be renewed automatically [50]. To fix the problem, a full CA key-roll was necessary to update all objects. Another problem occurred based on the dependency of several involved systems in the publication process. An internal registry system at RIPE NCC holds resources that are eligible for certification and is queried by the RPKI software performing certification. If a resource is not available anymore, ROAs are removed accordingly. Normally this is a healthy process and ROAs are expected to be purged when a resource is removed. But if the registry system becomes unavailable for some reason, ROAs are removed assuming the underlying resources are not present anymore. This led to the removal of 2669 ROAs leaving the prefixes unprotected for the duration of the incident [51].

In the beginning of 2021, a problem occurred which led to inconsistent certificate states in which a child certificate claimed to cover more resources compared to its parent [52]. Depending on the version and make of the RPKI validator that was used to process these certificates, all RPKI certificates within the duration of the incident were rejected. This resulted into dismissal within the validating AS of all resources managed by RIPE NCC.

Running a reliable RPKI infrastructure is a learning process that requires removing unexpected barriers along the way. Therefore, other RIRs should use the lesson´s learned by RIPE NCC to improve stability and operation of their infrastructure to prevent similar mistakes. The transparency with which outages are openly discussed and dealt with also helps to increase trust in the underlying system.

VII. Conclusion

This work gave a brief summary of currently available RPKI validators and compared seven RPKI validators regarding different characteristics. We found Routinator to perform the best amongst them and recommend this validator for a production environment. We also uncovered inconsistencies in VRP cache entries, leading to different results of the validators that operators need to watch out for. Moreover, our work discussed recent RPKI infrastructure outages to illustrate the complexity of running such an infrastructure reliably.

Acknowledgment

This work was partly supported by the project CONCORDIA that has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 830927.

REFERENCES

[1] K. Butler, T. Farley, P. McDaniel, and J. Rexford, "A Survey of BGP Security Issues and Solutions," *Proceedings of the IEEE*, vol. 98, pp. 100 – 122, 02 2010.

[2] C. Testart, "Reviewing a Historical Internet Vulnerability: Why Isn't BGP More Secure and What Can We Do About it?" in *The 46th Research Conference on Communication, Information and Internet Policy 2018*, TPRC, Ed., August 2018.

[3] A. Siddiqui, "A major BGP route leak by AS55410," APNIC Blog, 2021. [Online]. Available: https://blog.apnic.net/2021/04/26/a-major-bgp-route-leak-by-as55410/

[4] S. Kent, C. Lynn, and K. Seo, "Secure border gateway protocol (S-BGP)," *IEEE Journal on Selected areas in Communications*, vol. 18, no. 4, pp. 582–592, 2000.

[5] J. Ng, "Extensions to bgp to support secure origin bgp (sobgp)," Working Draft, IETF Secretariat, Internet-Draft draft-ng-sobgp-bgp-extensions-02, April 2004. [Online]. Available: https://tools.ietf.org/html/draft-ng-sobgp-bgp-extensions-02

[6] G. Goodell, W. Aiello, T. Griffin, J. Ioannidis, P. D. McDaniel, and A. D. Rubin, "Working around BGP: An Incremental Approach to Improving Security and Accuracy in Interdomain Routing." in *NDSS*, vol. 23. Citeseer, 2003, p. 156.

[7] T. Wan, E. Kranakis, and P. C. van Oorschot, "Pretty Secure BGP, psBGP." in *The Network and Distributed System Security (NDSS) Symposium 2005*, 2005.

[8] IETF SIDR, "Secure Inter-Domain Routing (sidr) Concluded WG," 2006. [Online]. Available: https://datatracker.ietf.org/wg/sidr/about/

[9] M. Lepinski and K. Sriram, "BGPsec Protocol Specification," RFC 8205, Sep. 2017. [Online]. Available: https://rfc-editor.org/rfc/rfc8205.txt

[10] M. Lepinski and S. Kent, "An Infrastructure to Support Secure Internet Routing," RFC 6480, Feb. 2012. [Online]. Available: https://rfc-editor.org/rfc/rfc6480.txt

[11] T. Chung, E. Aben, T. Bruijnzeels, B. Chandrasekaran, D. Choffnes, D. Levin, B. M. Maggs, A. Mislove, R. v. Rijswijk-Deij, J. Rula, and N. Sullivan, "Rpki is coming of age: A longitudinal study of rpki deployment and invalid route origins," in *Proceedings of the Internet Measurement Conference*, ser. IMC '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 406–419. [Online]. Available: https://doi.org/10.1145/3355369.3355596

[12] A. Cohen, Y. Gilad, A. Herzberg, and M. Schapira, "Jumpstarting BGP security with path-end validation," in *Proceedings of the 2016 ACM SIGCOMM Conference*, 2016, pp. 342–355.

[13] J. Deger and F. Kargl, *Evaluation of the Deployment Status of RPKI and Route Filtering*. Universitätsbibliothek Tübingen, 2020.

[14] S. Qing and D. Ma, "RPSTIR2," GitHub, bgpsecurity, 2020. [Online]. Available: https://github.com/bgpsecurity/rpstir2

[15] L. Poinsignon, M. Chris, and J. Bampton, "OctoRPKI," GitHub, Cloudflare, 2019. [Online]. Available: https://github.com/cloudflare/cfrpki

[16] NLnet Labs, "Routinator Manual," 2021. [Online]. Available: https://routinator.docs.nlnetlabs.nl/en/stable/

[17] LACNIC and NIC.MX, "FORT Validator - Github Repository," 2021. [Online]. Available: https://nicmx.github.io/FORT-validator/

[18] K. Dzonsons, C. Jeker, J. Snijders, T. de Raadt, S. Benoit, and T. Buehler, "rpki-client," OpenBSD, 2021. [Online]. Available: https://www.rpki-client.org/

[19] M. Puzanov, "rpki-prover," GitHub, 2020. [Online]. Available: https://github.com/lolepezy/rpki-prover

[20] N. Trenaman, "Lifecycle of the RIPE NCC RPKI Validator," RIPE NCC, 20. Oktober 2020. [Online]. Available: https://labs.ripe.net/author/nathalie_nathalie/lifecycle-of-the-ripe-ncc-rpki-validator/

[21] R. Austein, R. Bush *et al.*, "Dragon Research Labs RPKI Toolkit," 2006. [Online]. Available: https://github.com/dragonresearch/rpki.net

[22] J. Kristoff, R. Bush, C. Kanich, G. Michaelson, A. Phokeer, T. C. Schmidt, and M. Wählisch, "On Measuring RPKI Relying Parties," in *Proceedings of the ACM Internet Measurement Conference*, 2020, pp. 484–491.

[23] J. Kristoff, "Installing RPKI Relying Party Software," 2020. [Online]. Available: https://dataplane.org/jtk/blog/2020/11/installing-rpki-rp-software/

[24] T. Phuntsho, "How to Install an RPKI Validator," RIPE Labs, 2019. [Online]. Available: https://labs.ripe.net/author/tashi_phuntsho_3/how-to-install-an-rpki-validator/

[25] G. Michaelson, "RIPE's RPKI Validator is being phased out, so what are the other options?" APNIC, 2021. [Online]. Available: https://blog.apnic.net/2021/02/17/ripes-rpki-validator-is-being-phased-out-so-what-are-the-other-options/

[26] Routing Security, "RPKI RFCs Graph," 2021. [Online]. Available: https://rpki-rfc.routingsecurity.net/

[27] A. Tridgell and P. Mackerras, "The rsync algorithm," 1998. [Online]. Available: https://rsync.samba.org/tech_report/

[28] T. Bruijnzeels, O. Muravskiy, B. Weber, and R. Austein, "The RPKI Repository Delta Protocol (RRDP)," RFC 8182, Jul. 2017. [Online]. Available: https://rfc-editor.org/rfc/rfc8182.txt

[29] Y. Gilad, A. Cohen, A. Herzberg, M. Schapira, and H. Shulman, "Are We There Yet? On RPKI's Deployment and Security." in *NDSS*, 2017.

[30] J. Snijders, "Deep Dive on Manifest Handling," RPKI mailinglist, 2020. [Online]. Available: https://lists.nlnetlabs.nl/pipermail/rpki/2020-December/000245.html

[31] N. Rodday, Í. Cunha, R. Bush, E. Katz-Bassett, G. D. Rodosek, T. C. Schmidt, and M. Wählisch, "Revisiting rpki route origin validation on the data plane," in *Proc. of Network Traffic Measurement and Analysis Conference (TMA). IFIP.*, 2021.

[32] N. Rodday, L. Kaltenbach, Í. Cunha, R. Bush, E. Katz-Bassett, G. D. Rodosek, T. C. Schmidt, and M. Wählisch, "On the deployment of default routes in inter-domain routing," in *Proceedings of the ACM SIGCOMM 2021 Workshop on Technologies, Applications, and Uses of a Responsible Internet*, 2021, pp. 14–20.

[33] D. Mandelberg and R. Hansen, "RPSTIR," GitHub, bgpsecurity, 2021. [Online]. Available: https://github.com/bgpsecurity/rpstir

[34] ZDNS, "RPKIVIZ," ZDNS, 2020. [Online]. Available: http://rpkiviz.zdns.cn/

[35] D. Ma, "RPKIVIZ: Visualizing the RPKI," APNIC, 2020. [Online]. Available: https://blog.apnic.net/2020/04/23/rpkiviz-visualizing-the-rpki/

[36] L. Poinsignon *et al.*, "GoRTR," GitHub, Cloudflare, 2021. [Online]. Available: https://github.com/cloudflare/gortr

[37] NLnet Labs, "Homepage," 2021. [Online]. Available: https://nlnetlabs.nl/

[38] M. Hoffmann and A. Band, "Krill," GitHub, 2021. [Online]. Available: https://github.com/NLnetLabs/krill

[39] RPKI Community, "RPKI Discord Channel," 2021. [Online]. Available: https://discord.gg/8dvKB5Ykhy

[40] NLnet Labs, "RPKI Mailinglist – Discussion on RPKI deployment and tools developed by NLnet Labs," NLnet Labs, 2021. [Online]. Available: https://lists.nlnetlabs.nl/mailman/listinfo/rpki

[41] M. Hoffmann, A. Band *et al.*, "RTRTR – An RPKI data proxy," GitHub, NLnetLabs, 2020. [Online]. Available: https://github.com/NLnetLabs/rtrtr

[42] LACNIC and NIC.MX, "FORT project," 2021. [Online]. Available: https://fortproject.net/en/home

[43] K. Dzonsons, C. Jeker, S. Benoit, J. Snijders, and R. Scheck, "rpki-client-portable," GitHub, 2020. [Online]. Available: https://github.com/rpki-client/rpki-client-portable

[44] Raspberry Pi Foundation. (2021) Raspberry Pi Downloads. [Online]. Available: https://downloads.raspberrypi.org/

[45] L. Poinsignon, "Status Tweet," Twitter, 29. Januar 2021. [Online]. Available: https://twitter.com/lpoinsig/status/1355199025100668929?s=21

[46] P. Friedemann, "RAM usage in 0.9.0," GitHub, 2021. [Online]. Available: https://github.com/lolepezy/rpki-prover/issues/41

[47] M. Hoffmann, "Increased memory consumption of Routinator 0.9.0," RPKI mailinglist, 2021. [Online]. Available: https://lists.nlnetlabs.nl/pipermail/rpki/2021-June/000289.html

[48] M. Puzanov, " Rrdp rsync fallback #57 ," GitHub, 2021. [Online]. Available: https://github.com/lolepezy/rpki-prover/pull/57

[49] N. Trenaman, "Lessons Learned on Improving RPKI," 2020. [Online]. Available: https://labs.ripe.net/author/nathalie_nathalie/lessons-learned-on-improving-rpki/

[50] ——, "RPKI Outage Post-Mortem - Disk Quota," 2020. [Online]. Available: https://www.ripe.net/ripe/mail/archives/routing-wg/2020-February/004015.html

[51] ——, "RPKI ROA Deletion: Post-mortem," 2020. [Online]. Available: https://www.ripe.net/ripe/mail/archives/routing-wg/2020-April/004072.html

[52] ——, "RPKI Outage Post-Mortem - Inconsistent Certificates," 2021. [Online]. Available: https://mailarchive.ietf.org/arch/msg/sidrops/mlFkEcI0DCLv0ZXLY3uZmM1x2do/