On Measuring RPKI Relying Parties

John Kristoff
jkrist3@uic.edu
University of Illinois at Chicago
USA

Randy Bush
randy@psg.com
IIJ and Arrcus
Japan and USA

Chris Kanich
ckanich@uic.edu
University of Illinois at Chicago
USA

George Michaelson
ggm@apnic.net
APNIC
Australia

Amreesh Phokeer
amreesh@afrinic.net
AFRINIC
Mauritius

Thomas C. Schmidt
t.schmidt@haw-hamburg.de
HAW Hamburg
Germany

Matthias Wählisch
m.waehlisch@fu-berlin.de
Freie Universität Berlin
Germany

ABSTRACT

In this paper, we introduce a framework to observe RPKI relying parties (*i.e.,* those that fetch RPKI data from the distributed repository) and present insights into this ecosystem for the first time. Our longitudinal study of data gathered from three RPKI certification authorities (AFRINIC, APNIC, and our own CA) identifies different deployment models of relying parties and (surprisingly) prevalent inconsistent fetching behavior that affects Internet routing robustness. Our results reveal nearly 90% of relying parties are unable to connect to delegated publication points under certain conditions, which leads to erroneous invalidation of IP prefixes and likely widespread loss of network reachability.

CCS CONCEPTS

• **Networks** → **Public Internet**; *Routing protocols*; **Security protocols**; **Network measurement**; • **Security and privacy** → *Security protocols.*

KEYWORDS

Internet, Routing, Security

ACM Reference Format:
John Kristoff, Randy Bush, Chris Kanich, George Michaelson, Amreesh Phokeer, Thomas C. Schmidt, and Matthias Wählisch. 2020. On Measuring RPKI Relying Parties. In *ACM Internet Measurement Conference (IMC '20), October 27–29, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3419394.3423622

1 INTRODUCTION

The Resource Public Key Infrastructure (RPKI) [20] is an architecture to support improved security for the BGP [30] routing system on the Internet. For the first time, cryptographically secured objects such as a chain of X.509 certificates in the RPKI, can be used to authenticate BGP route announcements to defend against problems such as prefix re-origination route leaks [32].

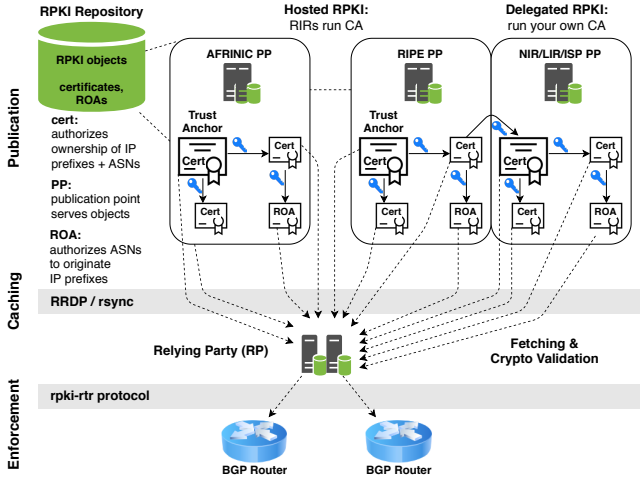


Figure 1: The RPKI ecosystem

Despite initial skepticism [15] the deployment of RPKI is underway. Several large transit providers (*e.g.,* AT&T [23], NTT [35], and Telia [4, 22]), Internet Exchange Points (*e.g.,* AMS-IX [1]), mid and small size ISPs (*e.g.,* Fiber Telecom [12]) as well as content providers (*e.g.,* Cloudflare [21]) evaluate and reject invalid routes in production based on RPKI information. RPKI as currently deployed protects against BGP misconfiguration and basic attacks of illegitimate origination of IP address space. Future RPKI-based mechanisms such as ASPA [2, 3] and BGPsec [13] will provide stronger protection against accidents and malicious attacks. For a validation system, to consider protected Internet resources (*i.e.,* IP prefixes and AS numbers) completely, it will be critical to fetch *all* RPKI objects.

RPKI in a nutshell. Much like the web-based PKI and certificate system, RPKI objects are created and signed by a *Certification Authority (CA).* All RPKI objects (*e.g., route origin authorization (ROA)*) are disseminated as files in a distributed repository of *publication point (PP)* servers. Analogous to a DNS authoritative server and DNS

IMC '20, October 27–29, 2020, Virtual Event, USA

ACM ISBN 978-1-4503-8138-3/20/10…$15.00
https://doi.org/10.1145/3419394.3423622

resolvers, a PP makes RPKI data available to *relying parties (RPs)*. In contrast to DNS resolvers, which fetch data on demand and have a partial view of the DNS, these RPs must periodically fetch all authoritative data and maintain a complete view. RPs use `rsync` [34] or RRDP [5] for data retrieval, then cryptographically validate received RPKI objects, cache the results, and relay data such as valid prefix-to-origin AS bindings to BGP routers for use in the route decision-making process (see Figure 1).

In the RPKI, the trust anchors (TAs) are each of the five independent Regional Internet Registries (RIRs): ARIN, APNIC, AFRINIC, LACNIC, and RIPE. The most common deployment model is the *hosted RPKI*, in which these RIRs maintain the RPKI infrastructure and offer RPKI as a service to their members. To allow for a fully distributed system, each owner of Internet resources may opt to run its own CA by deploying the *delegated RPKI* model. Ultimately, each RP validates all signed objects in the RPKI hierarchy, starting from each TA and following certificate paths to obtain a complete, crypto-verified view of the RPKI hierarchy.

As RPKI is increasingly used to validate and enforce Internet route announcements, the connection between PPs and the RPs is considered to be critical Internet infrastructure. Understanding how these entities interact with each other, and whether all objects propagate to a sufficiently large portion of the Internet, will be instrumental to determining whether the RPKI works as intended and can be relied upon.

In this paper, we present an investigation of the current operation, completeness, and consistency of RPs and RP software, in the RPKI as it is currently designed and deployed.

What might go wrong? There are several reasons that lead to an incomplete or outdated view of RPKI data. (*i*) A RP does not reach a publication point server; (*ii*) a RP fetches data infrequently; (*iii*) a RP does not follow the technical specifications.

State of the art. Current research that analyzed the deployment of RPKI focuses on two aspects. (*i*) The creation of ROA objects and the validation outcome of BGP announcements [17, 36, 37]. (*ii*) The use of RPKI-based route filtering [14, 31, 33]. It is an open research topic how relying parties behave and whether they have a complete view of the RPKI. Having a better understanding of the relying parties in the wild also provides insight into which networks potentially consider RPKI for origin validation.

Contributions. In this work, we present a first look at RP access behavior and RPKI synchronization robustness. We make the following contributions:

(1) We introduce a reproducible measurement framework for evaluating RPKI RP synchronization behavior, timeliness, and completeness.
(2) We characterize weaknesses in RPKI data propagation to RPs.
(3) We survey RPKI relying party software, access protocols, and synchronization patterns with publication points.
(4) We identify a fundamental mismatch in RP software behavior and and protocol design expectations through a series of controlled publication point server experiments.

2 MEASUREMENT FRAMEWORK

In this section, we introduce our measurement framework to better understand the deployment and operation of RPKI relying parties.

Table 1: Default refresh intervals for common RP software.

RP software	RRDP	rsync
FORT Validator [28]	1 hour	1 hour
rpki.net rcynic [18]	1 hour	1 hour
OpenBSD rpki-client [29]	not implemented	1 hour
OctoRPKI [10]	20 minutes	20 minutes
Routinator [19]	10 minutes	10 minutes
RIPE NCC Validator 3 [26]	2 minutes	10 minutes
RIPE NCC Validator 2 [25]	1 minute	10 minutes

Our framework is designed to allow for full reproducibility. It implements active as well as passive measurement methods based on the following three core building blocks: (*i*) controlled CAs and publication points, (*ii*) controlled relying parties, and (*iii*) controlled RPKI objects (*i.e.,* ROA Beacons).

2.1 Building Blocks

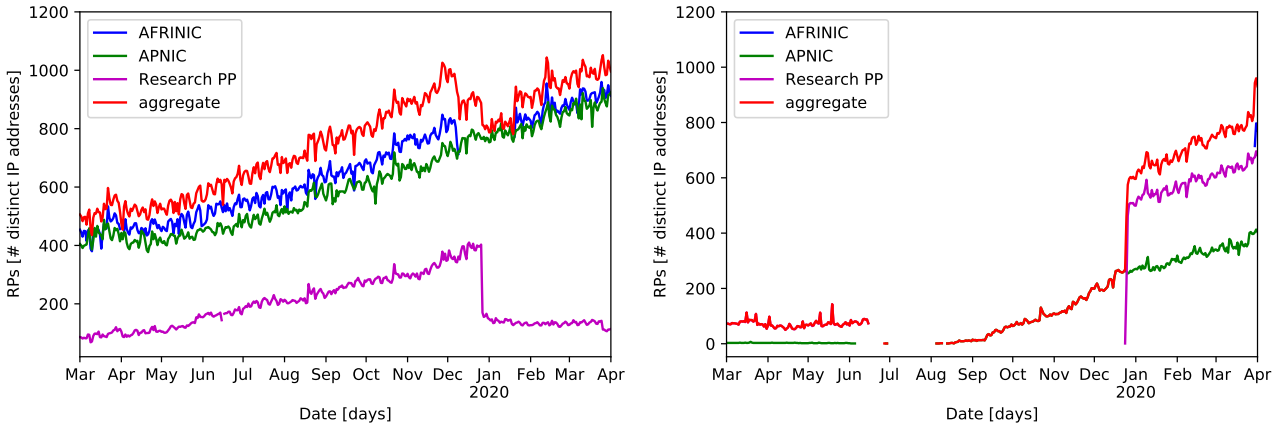
Controlled CA and PP server. Global RP behavior in the wild is best observed by PP server operators. To be independent of third party data, we leverage the hierarchical, distributed, and delegated design of the RPKI CA publication system. We operate a child, two grandchildren CAs and three PPs under one of the RIR CAs. This provides us with the necessary *vertical* view from RP, through the trust anchor, and towards multiple levels of delegated CAs. To implement a *horizontal* view, we plan to deploy delegated repositories under additional CAs in the future.

Our PPs provide the ability to evaluate `RRDP` and `rsync` access methods as well as IPv4 and IPv6 independently in order to study a variety of RP deployment scenarios and behavioral anomalies. We record time stamps, IP addresses, originating ASNs, and reverse DNS records of the accessing RPs. When RRDP is used, we capture the HTTP `User-Agent` to help the underlying software.

Controlled RP Cache Server. We operate multiple topologically distributed RPs, each running different software implementations fetching data recursively from all public trust anchors. These vantage points provide insights into the availability and performance between any RP and PP pair. We will publicly document IP addresses and configuration of fetching parameters to serve as ground truth not only for us but also for CA operators. Even if CA operators that run monitoring do not log access from our RPs, they will be able to investigate reachability issues.

ROA Beacons. We introduce ROA Beacons. Similar to BGP Beacons [24] they change their configuration based on well-defined, publicly available schedules to monitor propagation delays. We operate two types of ROA Beacons, those that are published and revoked periodically, and those that periodically alternate between a different assignment of a prefix to origin ASs.

ROA Beacons serve multiple purposes in detail. External parties can verify whether their RPs and routers maintain recent data. A PP operator can check when RPs fetch the changes. In this paper, we used ROA Beacons to verify ground truth (*i.e.,* whether RPs fetch updated data). Leveraging RPKI Beacons for complementary measurement studies will be part of our future work.



(a) rsync (b) RRDP

Figure 2: Synchronization activities from distinct RPs to fetch RPKI objects from monitored PPs via different protocols.

2.2 Establishing Ground Truth

We have run our CA/PP and ROA Beacon setup since 2018, while we have had controlled RPs running since December 2019. Our RPs were configured to use a combination of access methods and IP protocols for our initial measurement framework validation. We verified the clocks on each end of the connection. All had accurate and consistent notions of time using NTP. Our RPs were topologically diverse and verified to consistently reach all PPs with latency of at most 1 second. Using default RP software settings we verified the expected synchronization refresh intervals shown in Table 1 with what we observed from the RPs to the PPs.

Having confirmed our visibility and the ability to accurately detect interval frequency patterns, these directly observed vantage points establish our ground truth and baseline. We can then fit RP software default timers and observed interval patterns onto our entire pool of PP access logs to fingerprint uncontrolled RP cache servers and identify anomalous operating patterns.

3 RESULTS

In this section, we study the real-world access behavior of RPKI RP deployments from our vertical and horizontal measurement framework. We analyze insight from our own controlled PPs in the delegation tree along with contributed data from two RIR-operated PPs.[1] This allows us to spot anomalies, identify trends, uncover problems, and propose enhancements to improve the system.

3.1 Completeness of RPKI View

We first compare the synchronization patterns among RPs as seen from different PPs to answer our initial question: Do RPs fetch data from each and every PP?

Figure 2 depicts the global RP population over time by counting distinct RP IPv4 or IPv6 addresses seen each day, distinguished by different PP and fetching protocols.[2] The number of RPs is clearly on the rise. `rsync` activities at our PP dropped once RRDP was activated at the end of 2019, because modern RP software prefers RRDP over `rsync`. RIR PPs, however, continue to see both protocols, because the Trust Anchor Locator (TAL) specifies a `rsync` URI and RPs are advised to use this URI to retrieve the object referenced at every refresh interval [16]. This is why the AFRINIC PP did not see a similar drop in `rsync` when it activated RRDP in March, 2020.

While the access patterns are congruent among PPs, each PP sees different numbers of RPs, suggesting that not all RPs have a complete set of RPKI repository data. Particularly noteworthy is that our PP appears to see roughly 20% fewer RPs than the RIR PP parents. Surprisingly, a consistent set of RPs appear unable or unwilling to retrieve RPKI data from child PPs. We find over 80% of the missing RPs are using `rsync` to the RIR PPs the same day. We found that some RPs would eventually return to our child PP on another day using RRDP, but access was often intermittent. Furthermore, about 60% of these missing RPs have synchronization interval patterns matching the RIPEv2 or RIPEv3 validator. A partial explanation for this phenomenon is revealed in an access method packet filter experiment we conducted and detailed in § 3.4.

3.2 Type of Networks Hosting RPs

There has been relatively little discussion or guidance provided to operators on RP deployment strategies. The `rpki-rtr` specification [8] suggests three deployment models, where RPKI cache data is maintained: (*i*) by an upstream provider or third party, (*ii*) in one or more local cache systems, or (*iii*) throughout a distributed set of cache systems within each major network region. Our own experience and initial anecdotal evidence suggested that most operators setup one or two distinct RPs for their entire network, while a small handful of operators implemented their own unique, and often times complex, design strategies encompassing aspects of the three basic deployment scenarios.

[1] AFRINIC and APNIC PP access logs, not publicly available.

[2] There have been moments of transient data loss, but we don't believe these affect overall conclusions.

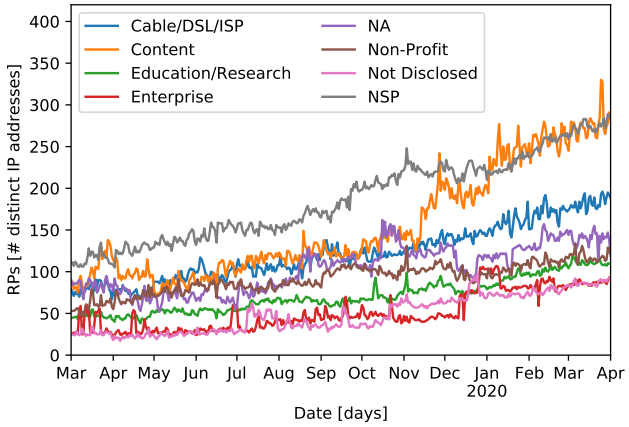


Figure 3: RPs deployed over time by classification of originating network.

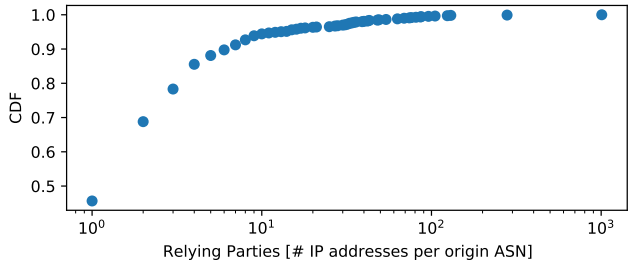


Figure 4: The number of RPs seen per ASN in the first four months of 2020.

Figure 3 shows the location of RPs based on the self-reported network classification as published in PeeringDB [27]. Recent upticks in RP growth are driven by three sectors: Cable/DSL/ISP, Content, and NSP (network service provider). Content networks have been responsible for a noticeable rise in RPs led by Facebook, which went from zero RPs in 2019 to nearly 70 in early 2020. While hosting networks are commonly used for RPs, such as Linode and DigitalOcean with approximately 50 and 25 RPs respectively in 2020, most networks appear to be deploying RPs within their own network boundaries near where their routers would reside.

To gain a sense of RP deployment by network, we graph the cumulative distribution function of distinct RP addresses per ASN seen in the first four months of 2020 in Figure 4. Over extended periods of time we found that the majority of RPs can be classified as server-class systems with stable source IP addresses. On average, 80% of RPs IP addresses will return each month and approximately 10% of all RPs have *rpki* in the DNS hostname. Even the few cases where RPs are not static server systems, this data provides a unique window into the natural roving population of RPs.

We were not surprised to find hosting providers with dozens of RPs over the course of monitoring period. A number of operators, researchers, or limited test RP installations appear on these networks for a variety of reasons. However, the network with the largest number of RPs, over 1000 distinct IP addresses, was a DNS infrastructure service provider with an elaborate RP design and deployment model. When we reached out to them to understand why so many RPs originate from their network, they explained that they deploy dozens of RPs throughout their network using a container-based model. These containers are transient nodes, with unstable addresses that may change frequently over short durations as network conditions change.

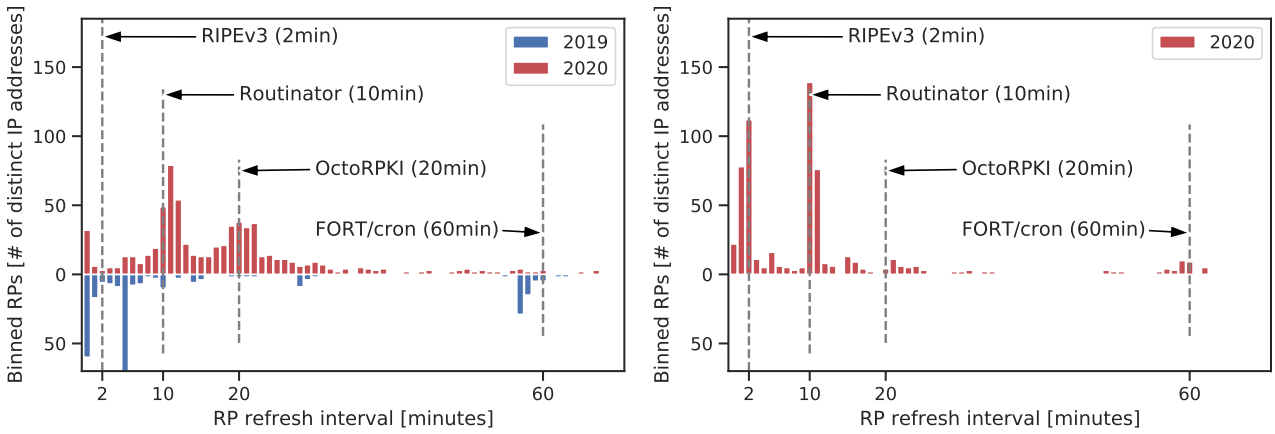
We also observed access via relay nodes of the Tor onion overlay, which not only obfuscates the IP address and location of the actual RP but also leads to distinct RP IP addresses per PP. One reason for this could be to conceal planned RPKI deployment, which we will study in future work.

3.3 Timeliness

Having considered the differences in access and view in the previous sections, we turn our attention on how current a RPKI view RPs are seeing. We refer to this quality as *timeliness*, which is a measure primarily derived from the configured refresh interval (*aka synchronization schedule*) of an RP. An RP must balance a refresh interval to ensure it obtains current data, but fetching too aggressively may place unnecessary strain on the PP infrastructure or waste RP cycles. The specifications are vague on RP refresh guidance, but suggest local RPs be "synchronized with each other at least every four to six hours." [6] An active IETF Internet-Draft [9] suggests rsync-based refresh intervals of once an hour and RRDP polling intervals of no more than every ten minutes. In practice, modern RP software defaults to one hour or less refresh intervals (refer to Table 1).

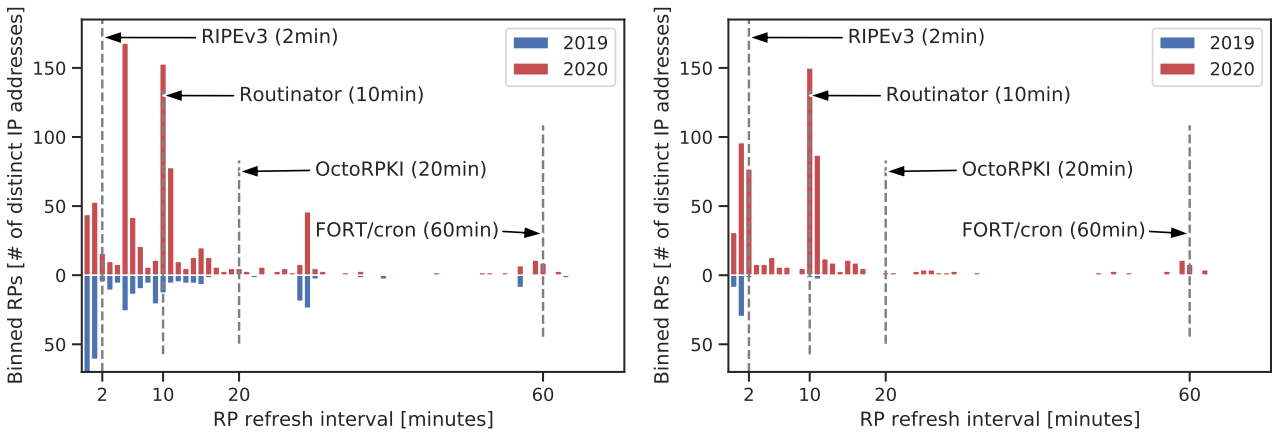
We examine inter-arrival connection times from each RP address to a PP in the wild. To avoid long term instability effects, we randomly choose a representative sample day. Most stable RPs will synchronize with a PP multiple times per day. Based on current practices and common recommendations we set a lower bound of 20 syncs per day. We reason that this helps ensure we consider only stable RP configurations and exclude short-lived, unstable RPs that may be used for testing or lab environments. However, on average up to 20% of RP IP addresses still fall below this threshold, which is significant. This would suggest a sizable population of RPs are lagging severely behind what is in the RPKI repository. Some RIRs update their repository as often as every few minutes if there are pending changes.

Our calculated interval from the minimum threshold connections are shown for all RIR PPs on March 4, 2020 and the year prior on the same date, see Figures 5 and 6. We notice RRDP exhibits reasonably predictable behavior compared to `rsync`, which varies from PP to PP and can be quite noisy. We believe this is in part due to the nature of `rsync` integration with RP software, which is a system call with less predictable completion times when refresh intervals tend to be longer using this access method. AFRINIC `rsync` frequency intervals are roughly what we might expect, while APNIC `rsync` seems to have peaks at unexpected intervals. When we evaluate corresponding user-agent strings in RRDP from the same IP addresses at APNIC, the majority of `rsync` clients at the unmarked 5-minute interval appear to be from RIPEv3 installations. We find no commonality of source IP address or origin AS among



(a) rsync (b) RRDP

Figure 5: Average connection interval at AFRINIC.



(a) rsync (b) RRDP

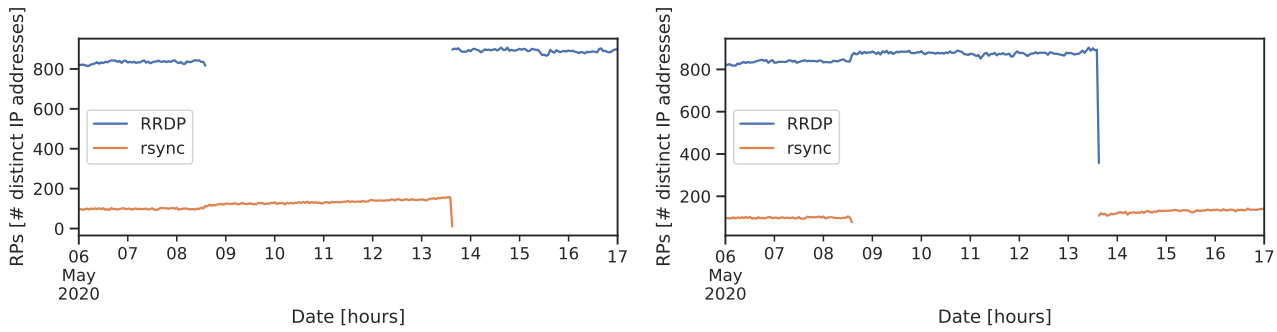
Figure 6: Average connection interval at APNIC.

these RPs. We suspect, but cannot verify that there exists or used to exist a build or common configuration for this software with that frequency interval. The small `rsync` peaks at 15 minutes consists primarily of rsync-only clients. Again we were unable to find any source material documenting this refresh rate from existing software.

In 2019, `rsync` was still the dominant access method used by RPs. We found `rsync` frequency intervals to only partially track modern RP software implementations with the implication that a number of RPs are running earlier generations of RP code and custom synchronization schedules.

When using RRDP, all well-known RP software implementations set an identifying user agent string in the HTTP connection. Since most RPs use a combination of RRDP and rsync, we leverage the RP IP address history from our vertical and horizontal views across a set of PPs. We are able to see RRDP fingerprints for roughly 95% for all RPs that connect using rsync this way.

The remaining small number of RPs we are unable to fingerprint by correlating RRDP user-agent strings from a corresponding IP address are rsync-only clients. There is one well known current implementation, rpki-client, and a couple of older ones (rcynic, rpstir) that support only rsync through cron with a recommended 1-hour synchronization refresh interval by default. [11, 18, 29] As as a result of our child PP configuration we discovered that all rsync-fetching RPs to our PP will include a trailing slash (`/`) of the replicated directory in the initial synchronization command with the exception of `rpki-client`. Since `rpki-client` is the only active implementation of an `rsync`-only RP this allows us to identify nearly all RP clients. As of this writing (Summer 2020), our PP sees approximately 40 distinct `rpki-client` RPs per day.



(a) Research Publication Point PP-*a*

(b) Research Publication Point PP-*b*

Figure 7: Impact of blocking synchronization protocols. On PP-*a* RRDP was first blocked on May 8 while `rsync` was still allowed, then `rsync` was blocked on May 13 while RRDP was allowed, and on PP-*b* vice versa. `rsync` fall-back rarely happened.

3.4 Filtering Experiment

Operating our own PP server enables us to shed light on unexpected RP behavior. When we evaluated the effect of disabling one of the access methods, (*i.e.,* the mandatory `rsync` vs. emerging RRDP) we were surprised by what we uncovered.

Most RP software have configuration settings to handle transient connection failures. PP operators can mitigate failures by deploying multiple PP instances such as with the use of DNS round robin or load balancing solutions. RPs will typically retry unreachable PPs at every synchronization interval.

Most PPs provide service interfaces for RRDP (a HTTPS-based service) as well as `rsync` (TCP port 873). On May 8, we blocked access from any RP to the RRDP service on one child PP-*a* and blocked `rsync` on a different child PP-*b*. On May 13, we swapped the protocol blocking on each server. We expected RPs that were using RRDP to fall back to `rsync` after a sufficient period of time. This would comply with current specs [20]. Conversely, we did not anticipate many RPs to migrate to RRDP from `rsync`; we assumed those RPs probably do not yet support RRDP or have been manually set to support only `rsync`. With our child CA we verified compliance with technical specifications, *i.e.,* how RPs would behave if one of the two access protocols suddenly became unavailable. This allows us to assess both robustness in general and potential transition scenarios in case `rsync` is deprecated. As shown in Figure 7, many RPs do not use implementations that comply with current specifications. The sudden decrease of RRDP sources at *a* on May 8 should show a corresponding rise in `rsync` when RPs discover RRDP has failed. Likewise we would expect the same at *b* on May 13. However, most RP software accounting for nearly 90% of all RPs do not fall back to `rsync`. For software popularity see Appendix C.

This experiment exposes a looming problem in RP behavior with delegated PPs. For example, if provider A publishes the associated ROA for an IPV4 /16, allocates a /19 from it to child B, and the child CA has the ROA which is not fetched, B's announcement for the /19 will be judged invalid. Either through natural failure modes or a targeted service interface attack, the loss of the RRDP service interface on a child PP can lead to a loss of reachability for B's /19 by any network operator using the RPKI for route validation. This RP behavior anomaly led to a lively exchange on the IETF SIDROPS working group mailing list [7].

4 CONCLUSION AND OUTLOOK

The RPKI is the rapidly growing foundation upon which improved security of the Internet BGP routing system is being constructed. In this paper, we presented a new reproducible RPKI measurement framework that focuses on the relying party cache servers to highlight their role in providing a complete and timely view of RPKI data. Our child CA and publication point deployment model ushers in the next stage in RPKI system understanding. Our initial results show a significant portion of deployed RPs are not obtaining a complete nor timely copy of RPKI data. Refresh intervals are crucial, but we have also identified systemic behavior by multiple implementations that may cause them to lose access to child CAs and PPs. Our study shows this problem may currently exist for up to 20% of deployed RPs. Considering that there is an increasing trend of both deployment of child CAs and RPKI-based route filtering, Internet routing may experience larger disturbances in the future if the erroneous fetching behavior of RPs is not fixed.

Our approach allows for greater data transparency, enabling RP connection data to be shared and made available to third parties. We will run and extend our measurement platform in the future. We plan to include child CAs and PPs under all RIRs. This is important, because as we have seen some operators choose not to accept the licensing agreement of one of the five RIRs in practice. Future work will help us understand how widespread these decisions are.

We also seek to further examine RP cache servers directly, to complement the completeness work started here. It is not enough to know that an RP contacted a publication point, it is important to also know what the outcome of the contact was. Are RPs adhering to certificate expiration dates? Do RPs correctly validate all ROAs? Are there ROAs that trigger different validation results across RP deployments, perhaps due to policy or implementation differences?

Finally, we will investigate whether we can leverage this new data source outside of the RPKI context such as extending IP hit lists by RP addresses.

Artifacts. All artifacts are available on https://rp-study.rpki.net.

ACKNOWLEDGMENTS

We would like to thank our shepherd kc claffy and the anonymous reviewers for their valuable detailed feedback. We are indebted to Stephen Kent for his detailed comments on an earlier version of this paper. For insight into RP software behavior we greatly appreciate the time and assistance provided by Pier Chiodi, Andrew Gallo, Jason Murray, Michael Sinatra, and Job Snijders.

REFERENCES

[1] AMS-IX. 2020. AMS-IX Route Servers. https://www.ams-ix.net/ams/documentation/ams-ix-route-servers.
[2] Alexander Azimov, Eugene Uskov, Randy Bush, Keyur Patel, Job Snijders, and Russ Housley. 2020. *A Profile for Autonomous System Provider Authorization.* Internet-Draft. Internet Engineering Task Force. https://datatracker.ietf.org/doc/html/draft-ietf-sidrops-aspa-profile-03 Work in Progress.
[3] Alexander Azimov, Eugene Uskov, Randy Bush, Keyur Patel, Job Snijders, and Russ Housley. 2020. *Verification of AS_PATH Using the Resource Certificate Public Key Infrastructure and Autonomous System Provider Authorization.* Internet-Draft. Internet Engineering Task Force. https://datatracker.ietf.org/doc/html/draft-ietf-sidrops-aspa-verification-05 Work in Progress.
[4] The Telia Carrier Blog. 2020. Dropping RPKI Invalid Prefixes. Retrieved May 23, 2020 from https://blog.teliacarrier.com/2020/02/05/dropping-rpki-invalid-prefixes/
[5] T. Bruijnzeels, O. Muravskiy, B. Weber, and R. Austein. 2017. *The RPKI Repository Delta Protocol (RRDP).* RFC 8182. IETF.
[6] R. Bush. 2014. *Origin Validation Operation Based on the Resource Public Key Infrastructure (RPKI).* RFC 7115. IETF.
[7] Randy Bush. 2020. Re: [Sidrops] nlnet rp and rsync. https://mailarchive.ietf.org/arch/msg/sidrops/p5v0fGfagEDHXkhV_DjGRZ13L_o/
[8] R. Bush and R. Austein. 2017. *The Resource Public Key Infrastructure (RPKI) to Router Protocol, Version 1.* RFC 8210. IETF.
[9] Randy Bush, Jay Borkenhagen, Tim Bruijnzeels, and Job Snijders. 2020. *Timing Parameters in the RPKI based Route Origin Validation Supply Chain.* Internet-Draft. Internet Engineering Task Force. https://tools.ietf.org/html/draft-ietf-sidrops-rpki-rov-timing-00 Work in Progress.
[10] Cloudflare. 2019–2020. OctoRPKI. https://github.com/cloudflare/cfrpki
[11] Raytheon BBN Technologies Corporation. 2011-2017. rpstir. https://github.com/bgpsecurity/rpstir
[12] Fiber Telecom AS41327. 2020. Peering Policy. https://www.fibertelecom.com/it/peering-policy.php.
[13] W. George and S. Murphy. 2017. *BGPsec Considerations for Autonomous System (AS) Migration.* RFC 8206. IETF.
[14] Yossi Gilad, Avichai Cohen, Amir Herzberg, Michael Schapira, and Haya Shulman. 2017. Are We There Yet? On RPKI's Deployment and Security. In *Proc. of NDSS.* ISOC.
[15] Sharon Goldberg. 2014. Why is It Taking So Long to Secure Internet Routing? *Commun. ACM* 57, 10 (September 2014), 56–63.
[16] G. Huston, S. Weiler, G. Michaelson, and S. Kent. 2016. *Resource Public Key Infrastructure (RPKI) Trust Anchor Locator.* RFC 7730. IETF.
[17] Daniele Iamartino, Cristel Pelsser, and Randy Bush. 2015. Measuring BGP route origin registration validation. In *Proc. of PAM (LNCS).* Springer, Berlin, 28–40.
[18] Dragon Research Labs. 2006-2016. rcynic. https://github.com/dragonresearch/rpki.net
[19] NLnet Labs. 2019–2020. Routinator 3000. https://www.nlnetlabs.nl/projects/rpki/routinator/
[20] M. Lepinski and S. Kent. 2012. *An Infrastructure to Support Secure Internet Routing.* RFC 6480. IETF.
[21] Martin J Levy. 2018. *RPKI – The required cryptographic upgrade to BGP routing.* The Cloudflare Blog. Cloudflare, https://blog.cloudflare.com/rpki/.
[22] AusNOG mailing list archive. 2020. Telstra AS1221 RPKI Implementation. Retrieved May 23, 2020 from http://lists.ausnog.net/pipermail/ausnog/2020-February/043901.html
[23] NANOG mailing list archive. 2019. AT&T/as7018 now drops invalid prefixes from peers. Retrieved May 23, 2020 from https://mailman.nanog.org/pipermail/nanog/2019-February/099501.html
[24] Zhuoqing Mao, Randy Bush, Timothy Griffin, and Matthew Roughan. 2003. BGP Beacons. In *In Proceedings of the Internet Measurement Conference* (Miami, Florida, USA) *(IMC 2003).* Association of Computing Machinery, New York, NY, USA, 1–14. https://doi.org/10.1145/948205.948207
[25] RIPE NCC. 2011–2018. RIPE NCC Validator 2. https://github.com/RIPE-NCC/rpki-validator
[26] RIPE NCC. 2019–2020. RIPE NCC Validator 3. https://www.ripe.net/manage-ips-and-asns/resource-management/certification/tools-and-resources
[27] PeeringDB. 2019. The Interconnection Database. https://www.peeringdb.com/.
[28] FORT project. 2019–2020. FORT Validator. https://fortproject.net/validator
[29] OpenBSD Project. 2019–2020. rpki-client. https://www.rpki-client.org/
[30] Y. Rekhter, T. Li, and S. Hares. 2006. *A Border Gateway Protocol 4 (BGP-4).* RFC 4271. IETF.
[31] Andreas Reuter, Randy Bush, Italo Cunha, Ethan Katz-Bassett, Thomas C. Schmidt, and Matthias Wahlisch. 2018. Towards a Rigorous Methodology for Measuring Adoption of RPKI Route Validation and Filtering. *ACM SIGCOMM Computer Communications Review* 48, 1 (April 2018), 19–27.
[32] K. Sriram, D. Montgomery, D. McPherson, E. Osterweil, and B. Dickson. 2016. *Problem Definition and Classification of BGP Route Leaks.* RFC 7908. IETF.
[33] Cecilia Testart, Philipp Richter, Alistair King, Alberto Dainotti, and David Clark. 2020. To Filter or Not to Filter: Measuring the Benefits of Registering in the RPKI Today. In *Proc. of PAM (LNCS, Vol. 12048).* Springer, Berlin Heidelberg, 71–87.
[34] Andrew Tridgell, Paul Mackerras, and Wayne Davison. 1998–2020. rsync. https://rsync.samba.org/
[35] NTT News & Video. 2020. NTT Improves Security of the Internet with RPKI Origin Validation Deployment. Retrieved May 23, 2020 from https://www.gin.ntt.net/ntt-improves-security-of-the-internet-with-rpki-origin-validation-deployment/
[36] Matthias Wählisch, Olaf Maennel, and Thomas C. Schmidt. 2012. Towards Detecting BGP Route Hijacking Using the RPKI. *SIGCOMM Comput. Commun. Rev.* 42, 4 (Aug. 2012), 103–104.
[37] Matthias Wählisch, Robert Schmidt, Thomas C. Schmidt, Olaf Maennel, Steve Uhlig, and Gareth Tyson. 2015. RiPKI: The Tragic Story of RPKI Deployment in the Web Ecosystem. In *Proc. of 14th ACM Workshop on Hot Topics in Networks (HotNets).* ACM, New York, 11:1–11:7.

A ETHICAL CONSIDERATIONS

We release all data gathered at our delegated CAs on https://rp-study.rpki.net. This will include IP addresses of the relying parties. We do not consider this data privacy-sensitive as the RPKI is a public repository.

B PUBLICATION POINTS IN MARCH 2020

RPKI publication point hosts	DNS records	
	A (IPv4)	AAAA (IPv6)
AFRINIC		
`rpki.afrinic.net`	✓	✓
APNIC		
`rpki.apnic.net`	✓	✓
`rpki.rand.apnic.net`	✓	✓
`rpki.cnnic.cn`	✓	✘
`rpki-ca.idnic.net`	✓	✘
`rpkica.twnic.tw`	✓	✘
`rpki-repository.nic.ad.jp`	✓	✘
ARIN		
`rpki.arin.net`	✓	✓
`rpki.admin.freerangecloud.com`	✓	✓
`rpki.tools.westconnect.ca`	✓	✓
`rpkica.mckay.com`	✓	✘
LACNIC		
`repository.lacnic.net`	✓	✓
`rpki-repo.registro.br`	✓	✓
RIPE		
`rpki.ripe.net`	✓	✓
`repository.rpki.rocks`	✓	✓
`rpki.admin.freerangecloud.com`	✓	✓
`rpki.qs.nu`	✓	✓
`ca.rg.net`	✓	✘
`rsync.rpki.nlnetlabs.nl`	✓	✘
`krill.heficed.net`	✓	✘

C RP SOFTWARE POPULARITY

Figure 8 shows the number of distinct RP IP addresses that fetch data via RRDP and are visible at our research PP. We map each RP to RP software based on the user agent string in HTTP.

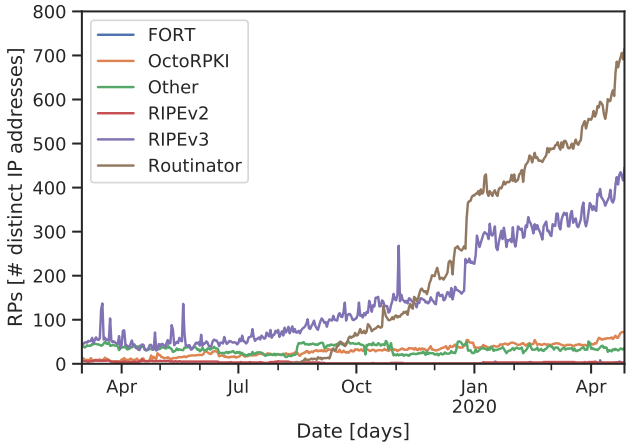


Figure 8: RP software popularity. Note: Research PP re-enabled RRDP on 2020-12-25.

D SET DIAGRAM OF RP IP ADDRESSES

Figure 9 shows the overlap of distinct RP IP addresses for a single day, measured across all three PPs.

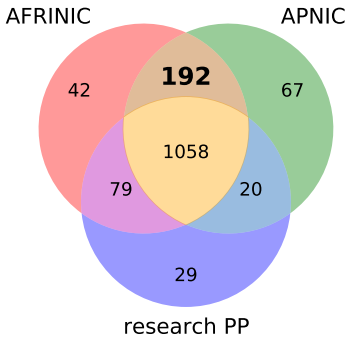
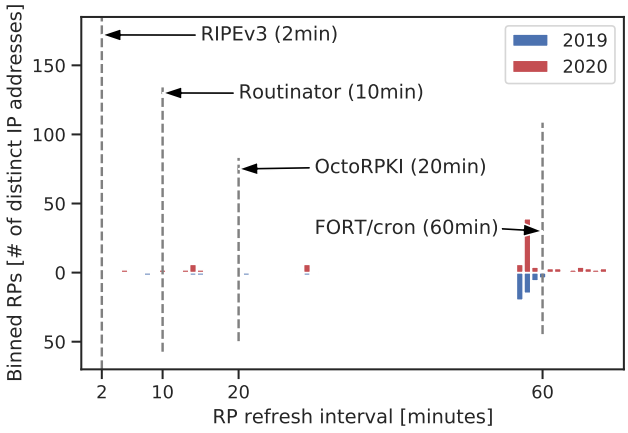


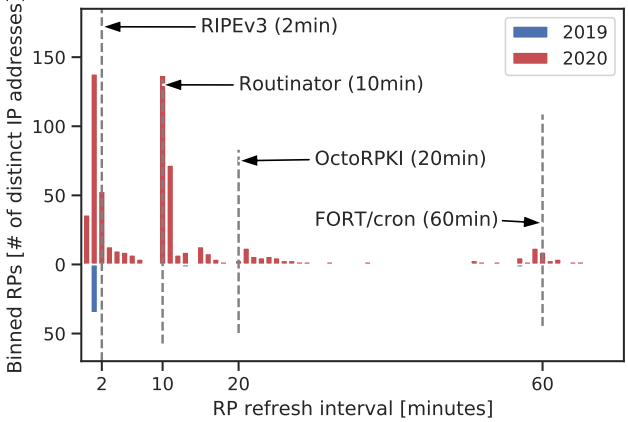
Figure 9: Set diagram showing the overlap of RP IP addresses seen across three PPs on March 30, 2020 (proportions not to scale).

E CONNECTION INTERVALS

Figure 10 shows the refresh intervals of RPs measured at our research PP. Details for AFRINIC and APNIC are presented in Section 3.3.



(a) rsync



(b) RRDP

Figure 10: Average connection interval at research PP.