Network Working Group
Internet-Draft
Intended status: Best Current Practice
Expires: 3 September 2026

T. Bruijnzeels
T. de Kock
RIPE NCC
F. Hill
ARIN
T. Harrison
APNIC
J. Snijders
BSD

2 March 2026

Best Practises for Operating Resource Public Key Infrastructure (RPKI) Publication Services
draft-ietf-sidrops-publication-server-bcp-06

Abstract

This document describes best current practices for operating an RFC 8181 RPKI publication engine and its associated publicly accessible rsync (RFC 5781) and RRDP (RFC 8182) repositories.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 3 September 2026.

Copyright Notice

Table of Contents

1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Introduction

RPKI material is created by Certification Authorities (CAs), this signed data is then submitted to a publication engine using the [RFC8181] publication protocol, and finally made available to RPKI Relying Parties (RPs) through publicly accessible rsync [RFC5781] and RRDP [RFC8182] repositories.

This document describes best current practices for operating RPKI publication services at a scale suitable for use with the global Internet routing system. The practises are based on more than a decade of operational experience from several implementers and operators of both client and server side.

3. Glossary

Term	Description
Publication engine	[RFC8181] publication server
Publishers	[RFC8181] clients (Certification Authorities)
RRDP server	Public-facing [RFC8182] RRDP server
Rsync server	Public-facing [RFC5781] rsync server
rsyncd	Software daemon package providing rsync service
RIR	Regional Internet Registry
NIR	National Internet Registry

Table 1

4. Publication Server

The publication engine handles the server side of the [RFC8181] publication protocol, that is, CAs interact with the publication engine. The publication engine also prepares the content for public consumption through RRDP and rsync. It is RECOMMENDED to deploy these engine functions on dedicated machines separate from those serving public requests via rsync and RRDP.

4.1. Self-Hosted CA and Self-Hosted Repository Considerations

The most common and RECOMMENDED approach for resource holders wishing to make use of the RPKI is to leverage a CA instance hosted and operated by the provider of those resources (e.g. an RIR or NIR). However, in some rare instances, resource holders might instead choose to deploy and manage a CA themselves. This latter type of CA is commonly referred to as a "self-hosted" or "delegated" CA.

If the resource holder chooses to operate their CA in a self-hosted fashion, the holder must also decide how they make their RPKI material available to the public: the holder can either deploy and operate their own publication engine and associated rsync and RRDP infrastructure (referred to as a "self-hosted repository"), or make use a third-party operated publication service (e.g., operated by an RIR).

If the resource holder chooses to self-host the repository, then they take on the responsibility for ensuring the high availability of their signed data via RRDP and rsync (further described in section 5 and 6 of this document).

Because RPs are expected to make use of cached data from previous successful fetches (Section 6 of [RFC9286]), short outages on the server side do not need to be cause for immediate concern -- provided the self-hosting operator restores access availability in a timely fashion, i.e., before objects become stale.

However, in practice, self-hosted repositories tend to exhibit more frequent availability issues when compared with those provided by larger specialized organisations such as RIRs and NIRs. Additionally, the greater the number of distinct repositories, the more workload for RPs and greater the chance for negative impact on the overall ecosystem. Therefore, CAs that act as parents of other CAs are RECOMMENDED to provide a publication service for their child CAs, and CAs with a parent who offers a publication service are RECOMMENDED to use that service (rather than self-hosting). If a CA’s parent does not offer a publication service, but the CA operator is able to use a another reliable third-party publication service, the CA operator SHOULD make use of that service in order to consolicate their data with other CAs and increase efficiency for RPs.

For the case of a ’grandchild’ CA, where CA1 is a TA, CA2 is a child CA of CA1, and CA3 is a child CA of CA2, there are several options for providing publication service to CA3:

1. RFC 8183 defines a 'referral' mechanism as part of the out-of-band CA setup protocol. If supported by CA1 and CA2, then this simplifies the process of registering CA3 as a direct publication client of CA1.

2. CA1 may support the registration of multiple publishers by CA2, by using the publisher_request/repository_response XML exchange defined in RFC 8183. CA2 would then be able to register a separate publisher on behalf of CA3.

3. CA2 may operate a publication proxy service (e.g. [rpki-publication-proxy]), which acts as the Publication Server for CA3. This proxy would set aside part of CA2's namespace at CA1 for the publication of CA3's objects, adjusting and forwarding requests from CA3 to CA1 accordingly.

For options 1 and 2, CAs operating as CA1 should consider the implications of providing direct publication service to CA3 in this way: for example, CA3 may expect publication service technical support from CA1 directly.

4.2. Publication Server as a Service

The CA-facing publication engine and public-facing repository services have different requirements on their availability and reachability. While the publication engine only needs to be accessed by publishers, the repository content MUST be highly available to any RP worldwide. Depending on the specific setup, this may allow for additional access restrictions in this context: for example, the publication engine can limit access to known publisher source IP addresses or apply rate limits.

If the publication engine is unavailable for some reason, this will prevent publishers from making new RPKI material available. The most immediate impact of this is that the publisher cannot distribute new issuances and revocations of ROAs ([RFC9582]), ASPAs ([I-D.ietf-sidrops-aspa-profile]), and BGPsec Router Certificates ([RFC8209]) for the duration of this outage. Thus, in effect, the resource holder cannot inform the world about changes to its routing intentions. If the outage persists for an extended period, then RPKI Manifests, CRLs, and Signed Objects cached by RPs will became stale, in turn hampering, for example, BGP Origin Validation ([RFC6811]). For the aforementioned reasons, the publication engine MUST be highly available.

Research ([rpki-time-in-flight]) on RPKI material propagation time, specifically the delay period between issuance of ROAs and the eventual application of the resulting Validated ROA Payloads in

Internet routing, showed that propagation time ranged between 15 and 95 minutes for the CAs and associated repositories that were part of the study. The study highlighted how the delay between signing and publication can be a major contributor to long propagation times.

It is RECOMMENDED to monitor the availability and latency of publication engines in a round-trip fashion by keeping track of expected and observed appearance of re-issued objects.

To make publishers aware of the root cause of disruption in the publication engine and allow them to plan accordingly ahead of time, maintenance windows SHOULD be planned and communicated to publishers.

4.3. Data Loss

Publication service operators MUST aim to minimise data loss. If a server restore was needed and a content regression occurred (for example, due to restoration from a slightly out-of-date backup), then the server MUST perform an RRDP session reset.

CAs typically only check in with their publication server when they have produced changes in RPKI material that need to be shared with the world. As a result, the CA may not be aware if the server performed a restore and their content regressed to an earlier state. This could result in a number of problems:

* The currently published ROAs no longer reflect the CA’s intentions.

* The CA might not reissue their Manifest or CRL in time, because they operated under the assumption that the currently-published Manifest and CRL have not yet became stale.

* Changes to publishers may not have been persisted. Newly registered publishers may not be present, and recently removed publishers may still be present.

Therefore, the publication engine operator SHOULD notify its dependent CAs about any service affecting issues as soon as possible, so that affected CAs know to initiate a full resynchronisation.

4.4. Publisher Repository Synchronisation

It is RECOMMENDED that publishing CAs always perform a list query as described in section 2.3 of [RFC8181] before submitting changes to the publication server. This approach means that any desynchronisation issue can be resolved at least as soon as the publisher is aware of updates that it needs to publish.

When publishing changes in material, CAs SHOULD send all of their changes using multiple PDUs contained within a single multi-element query message (described in section 2.2 and section 3.7.1 of [RFC8181]). This approach reduces the risk of changesets that were intended to take effect as an atomic action from taking effect in an inconsistent fashion.

In addition to the above, the publishing CA MAY perform regular planned synchronisation events where it issues an [RFC8181] list query and ensures that the publication server has the expected state, even if the CA has no new material to publish. For publication engines that serve a large number of CAs (e.g., thousands) this operation could become costly from a resource consumption perspective. Unfortunately, the [RFC8181] publication protocol has no proper support for rate limiting or signaling requests to CAs to backoff. Therefore, publishers SHOULD NOT perform this resynchronisation more frequently than once every 10 minutes, unless otherwise agreed with the publication engine operator.

5. Hostnames

It is RECOMMENDED that a different hostname is used in the public RRDP Server URI from that of the [RFC8181] service_uri used by publishers, as well as that of any rsync URIs (i.e., sia_base) used by the relevant publication service.

Using a unique hostnames for the different components will allow the operator to use dedicated infrastructure and/or a Content Delivery Network (CDN) for its RRDP content without interfering with the other functions.

If feasible, there is merit in using different TLDs and/or subdomains for these hostnames, as DNS issues at any level could otherwise be a single point of failure affecting both RRDP and rsync. Operators need to weigh this benefit against potential increased operational risk and the burden of maintaining multiple domains. Because the usefulness of this approach is highly context-dependent, no normative recommendation is provided here.

Furthermore, it is RECOMMENDED that DNSSEC is used in accordance with best current practice as described in [RFC9364].

6. IP Address Space and Autonomous Systems

To prevent failure scenarios which persist beyond remediation, the topological placement and reachability of publication servers in the global Internet routing system need to be considered very carefully. See section 6 of [RFC7115].

An example of a problematic scenario would be when a prefix or AS path related to a repository becomes invalid because of RPKI objects published in that repository. As a result, RPs may be unable to retrieve remediating updates from that repository.

With the above in mind, it is RECOMMENDED to use IP addresses for RRDP and rsync services from IP address space which is not subordinate to authorities solely dependent on those service endpoints.

It is also RECOMMENDED to host RRDP and rsync services in Autonomous Systems which are not subordinate to authorities publishing through those same endpoints.

It is RECOMMENDED to host RRDP and rsync services in different networks.

7. RRDP Server

7.1. Same Origin URIs

Publication server operators need to be aware of the normative updates to [RFC8182] in section 3.1 of [RFC9674]. In short, this update means that all delta and snapshot URIs need to reside on the same host, i.e., HTTP redirects or references to other origins are not allowed and not followed by RPs.

7.2. Endpoint Protection

Repository operators SHOULD use access control to protect their RRDP endpoints. For example, if the repository operator knows HTTP GET parameters are not used to provide service, then the operator can safely block any requests containing GET parameters.

7.3. Bandwidth and Data Usage

The bandwidth requirements for RRDP evolve over time and depends on many factors, consisting of three main groups:

1. RRDP-specific repository properties, such as the size of notification, delta, and snapshot files.

2. Properties of the CAs publishing through a particular server, such as the number of updates, number of objects, the length of the validity period, and size of objects.

3. Relying party behaviour, e.g. using HTTP compression, requiring timeouts or minimum transfer speed for downloads, and using conditional HTTP requests for notification.xml.

When an RRDP repository server is hosted behind a congested network link or otherwise overloaded (i.e. demand somehow exceeds available capacity), this can cause a cascading failure in which the aggregate load on the server continues to increase, resulting in degraded service for all RPs. For example, when an RP attempts to fetch one or more delta files, and one fails, it will typically try to fetch the snapshot (which is a larger object than the failed delta). If this also fails, the RP falls back to rsync. Furthermore, when the RP tries to use RRDP again on the next run, it typically starts by fetching the snapshot.

A publication service operator SHOULD attempt to prevent these issues by closely monitoring performance metrics (e.g. consumed bandwidth, available memory, disk I/O, expected functioning of a canary RP outside their network, watch logs for unexpected fallbacks to snapshot). Other than just increasing the capacity, several other measures to reduce demand for bandwidth are discussed in what follows.

The RRDP XML container and the embedded Base64-encoded content are highly compressible. Compression can reduce transferred data by about 50%. Therefore, HTTP endpoints SHOULD support at the very least gzip content encoding as described in section 8.4.1.3 of [RFC9110], in addition to any other popular compression algorithms that the server can support.

Because RRDP snapshots can be substantial in size (tens to hundreds of megabytes), special care must be taken for HTTP compression is not be unintentionally unavailable due to large file sizes (this is a behaviour observed in some CDNs).

7.4. Content Availability

Publication service operators MUST ensure that their RRDP servers are highly available.

The RRDP snapshot and delta files SHOULD remain available for two hours after they have become unreferenced by the latest RRDP notification file. Not doing so could lead to files being not found due to race conditions or slow fetching by RPs, and force RPs to fall back to full snapshot or RSYNC fetching.

If possible, it is RECOMMENDED that a CDN is used to serve the RRDP content. Special care MUST be taken to ensure that the notification file is not cached for longer than 1 minute unless the backend RRDP server is unavailable, in which case it is RECOMMENDED that stale files are served.

Some CDN services might cache 404 responses for resources not found on the backend server. Because of this, publication engines SHOULD use randomised unpredictable paths for snapshot and delta files, to avoid the intermediate CDN caching such 404 responses hampering future updates. Alternatively, the publication engine operator can instruct the CDN to purge cached information for the paths on which new files are published.

Note that some organisations that run a publication server may be able to attain a similar level of availability themselves without the use of a third-party CDN. This document makes no specific recommendations on achieving this, as this is highly dependent on local circumstances and operational preferences.

Also note that small repositories that serve a single CA, and which contain only a small amount of RPKI material that does not change frequently, may attain high availability using a modest setup. Short downtime would not lead to immediate issues for the CA, provided that the service is restored before their manifest and CRL become stale. This may be acceptable to the CA operator, however, because connecting to many distinct publication points can negatively impact RP workload, it is RECOMMENDED that these CAs instead use a publication service provided by their RIR or NIR.

7.5. Limit Notification File Size

Most RP implementations use conditional requests (e.g. If-Modified-Since) when fetching notification files, as this reduces the traffic for repositories that do not often update relative to the resynchronisation frequency of RPs. On the other hand, for repositories that update frequently, the underlying snapshot and delta content accounts for most of the traffic. For example, for a large repository in January 2024, with a notification file with 144 deltas covering 14 hours, the requests for the notification file accounted for 251GB of traffic out of a total of 55.5TB (i.e. less than 0.5% of the total traffic during that period).

However, for some servers, this ratio may be different. [RFC8182] stipulates that the sum of the size of deltas MUST not exceed the snapshot size, in order to avoid RPs downloading more data than necessary. However, this does not account for the size of the notification file that all RPs download. Keeping many deltas present

may allow RPs to recover more efficiently if they are significantly out of sync. Still, including all such deltas can also increase the total data transfer, because it increases the size of the notification file.

In order to mitigate potential problems here, the notification file size SHOULD be reduced by removing delta file entries from the notification file that already have been available for an extended period of time. Because some RP instances will only synchronize every 1-2 hours (observed in 2024), the RRDP server SHOULD include deltas for at least 4 hours.

Furthermore, it is RECOMMENDED that publication engines do not produce RRDP delta files more frequently than once per minute. A possible approach for this is that the publication engine SHOULD publish changes at a regular (one minute) interval. The RRDP server then makes available the new materials received from all Publishers in this interval in a single RRDP delta file.

While the latter may not reduce the amount of data due to changed objects, this will result in shorter notification files, and will reduce the number of delta files that RPs need to fetch and process.

7.6. Manifest and CRL Update Times

The manifest and CRL nextUpdate times and validity periods are determined by the issuing CA rather than the publication engine operator.

From the CA’s point of view, longer validity periods mean that there is more time to resolve unforeseen operational issues, since the current RPKI objects will remain valid for longer. On the other hand, longer validity periods also increase the risk of a successful replay attack.

From the publication engine’s point of view, shorter update times result in more data churn due to manifest and CRL reissuance. While the choice is made by the CAs, in certain modes of operation (e.g. hosted RPKI services) it may be possible to adjust the timing of manifest and CRL reissuance. In one large repository it was observed that increasing the reissuance cycle from once every 24 hours to once every 48 hours reduced data usage by approximately 50%, this is because generally most changes in the repository content are reissuance of manifests and CRLs, rather than newly issued ROAs and ASPAs.

7.7. Using Short and Unique Filenames for each Issuance

While the filenames of signed objects are determined by the issuing CA rather than the publication engine operator, publishers should be cognizant that their choice the file naming scheme can positively or negatively impact publication point operations.

* Because filenames are repeated multiple times throughout RPKI materials (e.g., in SIA fields, AIA fields, CRLDP fields, as part of in Manifest fileLists, and in the uri field in RRDP publish elements, etc), use of shorter filenames has a positive impact on storage & bandwidth required in the overall ecosystem. Therefore publishers are RECOMMENDED to use filenames shorter than 32 characters.

* The algorithm most commonly used in the preparation phase of rsync transfers relies on the 3-tuple of filename, filesize, and last-modification timestamp to determine whether material was updated and should be transferred. Therefore, using a new unique filename for each new issuance may help improve reliable propagation of newly signed material in the pipeline from publishers to RPs.

An additional benefit of using new unique filenames for new content is improved debuggability, because using the same name throughout time for different things may hamper, for example, swift and accurate interpretation of log messages containing filenames.

In summary, to conserve bandwidth, to improve reliability of object propagation, and to make debugging easier, publishers are RECOMMENDED use "one-time-use" EE certificates (section 3 of [RFC6487]) and to adhere to the guidelines for naming objects described in section 2.2 of [RFC6481].

7.8. ROA Prefix Aggregation

The practice of issuing ROAs with only a single prefix per ROA ([RFC9455]) can lead to many ROA objects being published by a given CA. However, clustering multiple prefixes in a single ROA (per origin AS) can achieve a significant reduction in the number of objects and the total size of a repository. In order to reduce bandwidth consumption and reduce the number of signatures, it is RECOMMENDED that issuing CAs cluster as many prefixes per ROA as possible, provided:

It is RECOMMENDED that issuing CAs cluster multiple prefix per ROA in case:

* Fate sharing is not a concern, for example, when both the parent and issuing CA are controlled by the same entity.

* The operational impact of publishing many ROAs outweighs the perceived fate sharing risks, e.g. because it leads to excessive bandwidth demands on the repository, or it's causing overly large manifest(s), or it leads to an excessive amount of data or number of files for RPs.

7.9. Consistent Load-Balancing

7.9.1. Notification File Timing

New RRDP notification files MUST NOT be made available to RPs before the associated snapshot and delta files also are available.

As a result, when using a load-balancing setup, special care SHOULD be taken to ensure that RPs that make multiple subsequent requests receive content from the same node (e.g. consistent hashing). This way, clients follow the timeline on one node where the referenced snapshot and delta files are available. Alternatively, publication infrastructure SHOULD ensure a particular ordering of the visibility of the snapshot plus delta and notification file. All nodes should receive the new snapshot and delta files before any node receives the new notification file.

When using a load-balancing setup with multiple backends, each backend MUST provide a consistent view and MUST update more frequently than the typical refresh rate for rsync repositories used by RPs. When these conditions hold, RPs observe the same RRDP session with the serial monotonically increasing.

Unfortunately, [RFC8182] does not specify RP behavior if the serial regresses. As a result, some RP implementations will fetch the snapshot to re-sync if a (substantial) serial regression is observed.

7.9.2. L4 Load-Balancing

If an RRDP repository uses L4 load-balancing, some load balancer implementations will keep in the pool connections to a node that is no longer active (e.g. one that is disabled because of maintenance). Due to HTTP keepalive, requests from an RP (or CDN edge) may continue to try to use the disabled node for an extended period. This issue is more pronounced with CDNs that use HTTP proxies internally when connecting to the origin while also load-balancing over multiple proxies. As a result, some requests may use a connection to the disabled server and retrieve stale content while other connections retrieve data from another server. Depending on the exact

configuration - for example, nodes behind the load balancer may have different RRDP sessions - this can lead to clients observing an inconsistent RRDP repository state.

Because of this issue, it is RECOMMENDED to, firstly, limit HTTP keepalive to a short period on the servers in the pool and, secondly, limit the number of HTTP requests per connection. When applying these recommendations, this issue is limited (and effectively less impactful when using a CDN due to caching) to a failover between RRDP sessions, where clients also risk reading a notification file for which some of the content is unavailable.

8. Rsync Server

This section elaborates on the following topics:

* Use of symlinks to provide consistent views on repository content

* Use of deterministic timestamps for files

* Load balancing and testing

8.1. Internally Consistent Repository Content

A naive implementation of the rsync server might change the repository content while RPs are transferring files. Even when the repository is consistent from the repository server’s point of view, clients may read an internally inconsistent set of files. Clients may get a combination of newer and older objects. This "phantom read" can lead to unpredictable and unreliable results. While modern RPs will treat such inconsistencies as a "Failed Fetch" ([RFC9286]), it is best to avoid this situation altogether, since a failed fetch for one repository can cause the rejection of delegated certificates and/or RPKI signed objects for a sub-CA when resources change.

One way to ensure that rsyncd serves its connected clients (RPs) with a consistent view of the repository is by configuring the rsyncd ’module’ path to a path that contains a symlink that the repository-writing process updates for every repository publication.

Following this process, when an update is published:

1. write the complete updated repository into a new directory

2. fix-up the timestamps of files (see next section)

3. change the symlink to point to the new directory

Most modern day implementations follow the above process (e.g. [krill-sync], [rpki-core], [rsyncit], the rpki.apnic.net repository implementation, and [rsync-move]).

Because rsyncd resolves this symlink when it chdirs into the module directory when a client connects, any connected RPs can read a consistent state for the duration of the connection. To limit the amount of disk space a repository uses, a rsync server must clean up old copies of the repository; the timing of these removal operations involves balancing the provision of service to slow clients against the additional disk space required to support those clients.

A repository can safely remove old hierarchies when no RP is still reading that data at a reasonable rate. Since the last moment an RP can start reading from a copy is when it was last "current", the time a client has to read a copy begins when it was last current (cf. the time when it was originally written).

Empirical data suggests that rsync server operators MAY assume it is safe to remove old versions of repositories after two hours. It is RECOMMENDED to monitor for "file has vanished" (or similar) lines in the rsync log file to detect how many clients are affected by the cleanup process timing parameters.

8.2. Deterministic Timestamps

By default, rsync implementations use the modification time and file size to determine if it should transfer a file. Therefore, throughout a file’s lifetime, the modification time SHOULD NOT change -- unless the file’s content changes.

The following deterministic heuristics are RECOMMEND as the file’s timestamp when writing objects to disk:

* For CRLs, use the value of thisUpdate.

* For RPKI Signed Objects, use the CMS signing-time (see ([RFC9589])).

* For CA and BGPsec Router Certificates, use the value of notBefore.

* For directories, use any constant value.

8.3. Load Balancing and Testing

To increase availability during both regular maintenance and exceptional situations, a rsync repository that strives for high availability should be deployed on multiple nodes load-balanced by an L4 load balancer. Because rsync sessions use a single TCP connection per synchronisation attempt, there is no need for consistent load-balancing between multiple rsync servers as long as they each provide a consistent view.

It is RECOMMENDED that the rsync server is load tested to ensure that it can handle simultaneous requests from all RPs, in case those RPs need to fall back from using RRDP (as is currently preferred).

It is RECOMMENDED to serve rsync repositories from local storage, so that the host operating system can optimally use its I/O cache. Using network storage is NOT RECOMMENDED, because it may not benefit from this cache. For example, when using NFS, the operating system might not be able to cache the directory listing(s) of the repository.

It is RECOMMENDED to set the "max connections" to a value that allows a single node to handle simultaneous resynchronisation by that number of RPs, taking into account the amount of time that RP implementations usually allow for rsync resychronisation. Load-testing results show that machine memory is likely the limiting factor for large repositories that are not IO limited.

The number of rsync servers needed depends on the number of RPs, their refresh rate, and the "max connections" used. These values are subject to change over time, so it is hard to give clear recommendations here except to restate that it is RECOMMENDED to load-test rsync service and reevaluating parameters over time.

9. Acknowledgments

The authors wish to thank Mike Hollyman for editorial suggestions.

10. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC5781] Weiler, S., Ward, D., and R. Housley, "The rsync URI Scheme", RFC 5781, DOI 10.17487/RFC5781, February 2010, <https://www.rfc-editor.org/info/rfc5781>.

[RFC6481] Huston, G., Loomans, R., and G. Michaelson, "A Profile for Resource Certificate Repository Structure", RFC 6481, DOI 10.17487/RFC6481, February 2012, <https://www.rfc-editor.org/info/rfc6481>.

[RFC6487] Huston, G., Michaelson, G., and R. Loomans, "A Profile for X.509 PKIX Resource Certificates", RFC 6487, DOI 10.17487/RFC6487, February 2012, <https://www.rfc-editor.org/info/rfc6487>.

[RFC6811] Mohapatra, P., Scudder, J., Ward, D., Bush, R., and R. Austein, "BGP Prefix Origin Validation", RFC 6811, DOI 10.17487/RFC6811, January 2013, <https://www.rfc-editor.org/info/rfc6811>.

[RFC7115] Bush, R., "Origin Validation Operation Based on the Resource Public Key Infrastructure (RPKI)", BCP 185, RFC 7115, DOI 10.17487/RFC7115, January 2014, <https://www.rfc-editor.org/info/rfc7115>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8181] Weiler, S., Sonalker, A., and R. Austein, "A Publication Protocol for the Resource Public Key Infrastructure (RPKI)", RFC 8181, DOI 10.17487/RFC8181, July 2017, <https://www.rfc-editor.org/info/rfc8181>.

[RFC8182] Bruijnzeels, T., Muravskiy, O., Weber, B., and R. Austein, "The RPKI Repository Delta Protocol (RRDP)", RFC 8182, DOI 10.17487/RFC8182, July 2017, <https://www.rfc-editor.org/info/rfc8182>.

[RFC8209] Reynolds, M., Turner, S., and S. Kent, "A Profile for BGPsec Router Certificates, Certificate Revocation Lists, and Certification Requests", RFC 8209, DOI 10.17487/RFC8209, September 2017, <https://www.rfc-editor.org/info/rfc8209>.

[RFC9110] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", STD 97, RFC 9110, DOI 10.17487/RFC9110, June 2022, <https://www.rfc-editor.org/info/rfc9110>.

[RFC9286] Austein, R., Huston, G., Kent, S., and M. Lepinski, "Manifests for the Resource Public Key Infrastructure (RPKI)", RFC 9286, DOI 10.17487/RFC9286, June 2022, <https://www.rfc-editor.org/info/rfc9286>.

[RFC9364] Hoffman, P., "DNS Security Extensions (DNSSEC)", BCP 237, RFC 9364, DOI 10.17487/RFC9364, February 2023, <https://www.rfc-editor.org/info/rfc9364>.

[RFC9455] Yan, Z., Bush, R., Geng, G., de Kock, T., and J. Yao, "Avoiding Route Origin Authorizations (ROAs) Containing Multiple IP Prefixes", BCP 238, RFC 9455, DOI 10.17487/RFC9455, August 2023, <https://www.rfc-editor.org/info/rfc9455>.

[RFC9582] Snijders, J., Maddison, B., Lepinski, M., Kong, D., and S. Kent, "A Profile for Route Origin Authorizations (ROAs)", RFC 9582, DOI 10.17487/RFC9582, May 2024, <https://www.rfc-editor.org/info/rfc9582>.

[RFC9589] Snijders, J. and T. Harrison, "On the Use of the Cryptographic Message Syntax (CMS) Signing-Time Attribute in Resource Public Key Infrastructure (RPKI) Signed Objects", RFC 9589, DOI 10.17487/RFC9589, May 2024, <https://www.rfc-editor.org/info/rfc9589>.

[RFC9674] Snijders, J., "Same-Origin Policy for the RPKI Repository Delta Protocol (RRDP)", RFC 9674, DOI 10.17487/RFC9674, December 2024, <https://www.rfc-editor.org/info/rfc9674>.

11. Informative References

[I-D.ietf-sidrops-aspa-profile]
Azimov, A., Uskov, E., Bush, R., Snijders, J., Housley, R., and B. Maddison, "A Profile for Autonomous System Provider Authorization", Work in Progress, Internet-Draft, draft-ietf-sidrops-aspa-profile-22, 6 February 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-sidrops-aspa-profile-22>.

[krill-sync]
Bruijnzeels, T., "krill-sync", 2023, <https://github.com/NLnetLabs/krill-sync>.

[rpki-core]
Team, R., "rpki-core", 2023, <https://github.com/RIPE-NCC/rpki-core>.

[rpki-publication-proxy]
APNIC, "rpki-publication-proxy", 2018,
<https://github.com/APNIC-net/rpki-publication-proxy>.

[rpki-time-in-flight]
Fontugne, R., Phokeer, A., Pelsser, C., Vermeulen, K., and R. Bush, "RPKI Time-of-Flight: Tracking Delays in the Management, Control, and Data Planes", 2022, <https://www.iijlab.net/en/members/romain/pdf/romain_pam23.pdf>.

[rsync-move]
Snijders, J., "rpki-rsync-move.sh.txt", 2023,
<https://www.bsd.nl/publications/rpki-rsync-move.sh.txt>.

[rsyncit] Team, R., "rpki-core", 2023,
<https://github.com/RIPE-NCC/rsyncit>.

Authors' Addresses

Tim Bruijnzeels
RIPE NCC
Email: tbruijnzeels@ripe.net

Ties de Kock
RIPE NCC
Email: tdekock@ripe.net

Frank Hill
ARIN
Email: frank@arin.net

Tom Harrison
APNIC
Email: tomh@apnic.net

Job Snijders
BSD Software Development
Amsterdam
Netherlands
Email: job@bsd.nl