                 The RPKI Repository Delta Protocol (RRDP)

Abstract

   In the Resource Public Key Infrastructure (RPKI), Certificate
   Authorities (CAs) publish certificates, including end-entity
   certificates, Certificate Revocation Lists (CRLs), and RPKI signed
   objects to repositories. Relying Parties retrieve the published
   information from those repositories. This document specifies a new
   RPKI Repository Delta Protocol (RRDP) for this purpose. RRDP was
   specifically designed for scaling. It relies on an Update
   Notification File which lists the current Snapshot and Delta Files
   that can be retrieved using HTTPS (HTTP over Transport Layer Security
   (TLS)), and it enables the use of Content Distribution Networks
   (CDNs) or other caching infrastructures for the retrieval of these
   files.

Table of Contents

1. Introduction

In the Resource Public Key Infrastructure (RPKI), Certificate
Authorities publish certificates [RFC6487], RPKI signed objects
[RFC6488], manifests [RFC6486], and CRLs to repositories. CAs may
have an embedded mechanism to publish to these repositories, or they
may use a separate Repository Server and publication protocol. RPKI
repositories are currently accessible using the rsync protocol
[RSYNC], allowing Relying Parties to synchronize a local copy of the
RPKI repository used for validation with the remote repositories
[RFC6481].

rsync [RSYNC] has proven valuable in the early deployment of RPKI,
because it allowed operators to gain experience without the need to
invent a custom protocol. However, operational experience has
brought concerns to light that we wish to address here:

o rsync [RSYNC] is designed to limit the amount of data that needs
   to be transferred between client and server. However, the server
   needs to spend significant resources in terms of CPU and memory
   for every connection. This is a problem in an envisioned RPKI
   deployment where thousands of Relying Parties query a small number
   of central repositories, and it makes these repositories weak to
   denial-of-service attacks.

o A secondary concern is the lack of supported rsync server and
   client libraries. In practice, all implementations have to make
   system calls to an rsync binary. This is inefficient; it
   introduces fragility with regards to updates of this binary, makes
   it difficult to catch and report problems to operators, and
   complicates software development and testing.

This document specifies an alternative repository access protocol
based on Update Notification, Snapshot, and Delta Files that a
Relying Party can retrieve over the HTTPS protocol. This allows
Relying Parties to either perform a full (re-)synchronization of
their local copy of the repository using Snapshot Files or use Delta
Files to keep their local repository updated after initial
synchronization. We call this the RPKI Repository Delta Protocol, or
RRDP in short.

RRDP was designed to support scaling in RPKI's asymmetric deployment.
It is consistent (in terms of data structures) with the publication
protocol [RFC8181] and treats publication events of one or more
repository objects as discrete events that can be communicated to
Relying Parties. This approach helps to minimize the amount of data
that traverses the network and thus helps minimize the amount of time
until repository convergence occurs. RRDP also provides a standards-

based way to obtain consistent, point-in-time views of a single
repository, eliminating a number of consistency-related issues.
Finally, this approach allows these discrete events to be
communicated as immutable files. This enables Repository Servers to
pre-calculate these files only once for all clients, thus limiting
the CPU and memory investments required, and enables the use of a
caching infrastructure to reduce the load on a Repository Server when
a large number of Relying Parties are querying it.

This document allows the use of RRDP as an additional repository
distribution mechanism for RPKI. In time, RRDP may replace rsync
[RSYNC] as the only mandatory-to-implement repository distribution
mechanism. However, this transition is outside of the scope of this
document.

2. Requirements Notation

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.

3. RPKI Repository Delta Protocol Implementation

3.1. Informal Overview

Certification Authorities in the RPKI use a Repository Server to
publish their RPKI products, such as manifests, CRLs, signed
certificates, and RPKI-signed objects. This Repository Server may be
remote or embedded in the Certificate Authority engine itself.
Certificates in the RPKI that use a Repository Server that supports
RRDP include a special Subject Information Access (SIA) pointer
referring to an Update Notification File.

The Update Notification File includes a globally unique session_id in
the form of a version 4 Universally Unique IDentifier (UUID)
[RFC4122] and serial number that can be used by the Relying Party to
determine if it and the repository are synchronized. Furthermore, it
includes a link to the most recent complete snapshot of current
objects that are published by the Repository Server, and a list of
links to Delta Files, for each revision starting at a point
determined by the Repository Server, up to the current revision of
the repository.

A Relying Party that learns about an Update Notification File
location for the first time can download it and then proceed to
download the latest Snapshot File, thus creating a local copy of the

repository that is in sync with the Repository Server. The Relying
Party records the location of this Update Notification File, the
session_id, and the current serial number.

Relying Parties are encouraged to re-fetch this Update Notification
File at regular intervals, but not more often than once per minute.
After re-fetching the Update Notification File, the Relying Party may
find that there are one or more Delta Files available that allow it
to synchronize its local repository with the current state of the
Repository Server. If no contiguous chain of deltas from the Relying
Party's serial to the latest repository serial is available, or if
the session_id has changed, the Relying Party performs a full
resynchronization instead.

As soon as the Relying Party fetches new content in this way, it
could start a validation process. An example of a reason why a
Relying Party may not choose to do this immediately is because it has
learned of more than one notification location, and it prefers to
complete all its updates before validating.

The Repository Server could use a caching infrastructure to reduce
its load, particularly because snapshots and deltas for any given
session_id and serial number contain an immutable record of the state
of the Repository Server at a certain point in time. For this
reason, these files can be cached indefinitely. Update Notification
Files are polled by Relying Parties to discover if updates exist; for
this reason, Update Notification Files may not be cached for longer
than one minute.

3.2. Certificate Authority Use

Certificate Authorities that use RRDP MUST include an instance of an
SIA AccessDescription extension in resource certificates they
produce, in addition to the ones defined in [RFC6487]:

```
        AccessDescription ::= SEQUENCE {
           accessMethod OBJECT IDENTIFIER,
           accessLocation GeneralName }
```

This extension MUST use an accessMethod of id-ad-rpkiNotify; see
Section 6:

```
   id-pkix OBJECT IDENTIFIER ::= { iso(1) identified-organization(3)
     dod(6) internet(1) security(5) mechanisms(5) pkix(7) }

   id-ad OBJECT IDENTIFIER ::= { id-pkix 48 }

   id-ad-rpkiNotify OBJECT IDENTIFIER ::= { id-ad 13 }
```

   The accessLocation MUST be an HTTPS URI as defined in [RFC7230] that
   will point to the Update Notification File for the Repository Server
   that publishes the products of this Certificate Authority
   certificate.

3.3. Repository Server Use

3.3.1. Initialization

   When the Repository Server initializes, it performs the following
   actions:

   o The server MUST generate a new random version 4 UUID (see
      Section 4.1.3 of [RFC4122]) to be used as the session_id.

   o The server MUST then generate a Snapshot File for serial number
      ONE for this new session that includes all currently known
      published objects that the Repository Server is responsible for.
      Note that this Snapshot File may contain zero publish elements at
      this point if no objects have been submitted for publication yet.

   o This Snapshot File MUST be made available at a URL that is unique
      to this session_id and serial number, so that it can be cached
      indefinitely. The format and caching concerns for Snapshot Files
      are explained in more detail in Section 3.5.2.

   o After the Snapshot File has been published, the Repository Server
      MUST publish a new Update Notification File that contains the new
      session_id, has serial number ONE, has one reference to the
      Snapshot File that was just published, and contains no delta
      references. The format and caching concerns for Update
      Notification Files are explained in more detail in Section 3.5.1.

3.3.2. Publishing Updates

   Whenever the Repository Server receives updates from a Certificate
   Authority, it MUST generate new snapshot and Delta Files within one
   minute. If a Repository Server services a large number of
   Certificate Authorities, it MAY choose to combine updates from
   multiple CAs. If a Repository Server combines updates in this way,
   it MUST ensure that publication never postponed for longer than one
   minute for any of the CAs involved.

   Updates are processed as follows:

   o The new repository serial number MUST be one greater than the
      current repository serial number.

   o A new Delta File MUST be generated for this new serial. This
      Delta File MUST include all new, replaced, and withdrawn objects
      for multiple CAs, if applicable, as a single change set.

   o This Delta File MUST be made available at a URL that is unique to
      the current session_id and serial number, so that it can be cached
      indefinitely.

   o The format and caching concerns for Delta Files are explained in
      more detail in Section 3.5.3.

   o The Repository Server MUST also generate a new Snapshot File for
      this new serial. This file MUST contain all "publish" elements
      for all current objects.

   o The Snapshot File MUST be made available at a URL that is unique
      to this session and new serial, so that it can be cached
      indefinitely.

   o The format and caching concerns for Snapshot Files are explained
      in more detail in Section 3.5.2.

   o Any older Delta Files that, when combined with all more recent
      Delta Files, will result in the total size of deltas exceeding the
      size of the snapshot MUST be excluded to avoid that Relying
      Parties download more data than necessary.

   o A new Update Notification File MUST now be created by the
      Repository Server. This new Update Notification File MUST include
      a reference to the new Snapshot File and all Delta Files selected
      in the previous steps.

   o The format and caching concerns for Update Notification Files are
      explained in more detail in Section 3.5.1.

   If the Repository Server is not capable of performing the above for
   some reason, then it MUST perform a full re-initialization, as
   explained above in Section 3.3.1.

3.4. Relying Party Use

3.4.1. Processing the Update Notification File

   When a Relying Party performs RPKI validation and learns about a
   valid certificate with an SIA entry for the RRDP protocol, it SHOULD
   use this protocol as follows.

The Relying Party MUST download the Update Notification File, unless
an Update Notification File was already downloaded and processed from
the same location in this validation run or a polling strategy was
used (see Section 3.4.4).

It is RECOMMENDED that the Relying Party uses a "User-Agent" header
explained in Section 5.5.3. of [RFC7231] to identify the name and
version of the Relying Party software used. It is useful to track
capabilities of Relying Parties in the event of changes to the RPKI
standards.

When the Relying Party downloads an Update Notification File, it MUST
verify the file format and validation steps described in
Section 3.5.1.3. If this verification fails, the file MUST be
rejected and RRDP cannot be used. See Section 3.4.5 for
considerations.

The Relying Party MUST verify whether the session_id matches the last
known session_id for this Update Notification File location. Note
that even though the session_id is a random UUID value, it alone MUST
NOT be used by a Relying Party as a unique identifier of a session
but always together with the location of the Update Notification
File. The reason for this is that a malicious server can use an
existing session_id from another Repository Server.

If the session_id matches the last known session_id, then a Relying
Party MAY download and process missing Delta Files as described in
Section 3.4.2, provided that all Delta Files for serial numbers
between the last processed serial number and the current serial
number in the Update Notification File can be processed this way.

If the session_id matches the last known session_id, but Delta Files
were not used, then the Relying Party MUST download and process the
Snapshot File on the Update Notification File as described in
Section 3.4.3.

If the session_id does not match the last known session_id, the
Relying Party MUST update its last known session_id to the value
specified in the downloaded Update Notification File. The Relying
Party MUST then download and process the Snapshot File specified in
the downloaded Update Notification File as described in
Section 3.4.3.

3.4.2. Processing Delta Files

   If an Update Notification File contains a contiguous chain of links
   to Delta Files from the last processed serial number to the current
   serial number, then Relying Parties MUST attempt to download and
   process all Delta Files in order of serial number as follows.

   When the Relying Party downloads a Delta File, it MUST verify the
   file format and perform validation steps described in
   Section 3.5.3.3. If this verification fails, the file MUST be
   rejected.

   Furthermore, the Relying Party MUST verify that the hash of the
   contents of this file matches the hash on the Update Notification
   File that referenced it. In case of a mismatch of this hash, the
   file MUST be rejected.

   If a Relying Party retrieved a Delta File that is valid according to
   the above criteria, it performs the following actions:

   o The Relying Party MUST verify that the session_id matches the
      session_id of the Update Notification File. If the session_id
      values do not match, the file MUST be rejected.

   o The Relying Party MUST verify that the serial number of this Delta
      File is exactly one greater than the last processed serial number
      for this session_id, and if not, this file MUST be rejected.

   o The Relying Party SHOULD add all publish elements to a local
      storage and update its last processed serial number to the serial
      number of this Delta File.

   o When a Relying Party encounters a "withdraw" element, or a
      "publish" element where an object is replaced, in a delta that it
      retrieves from a Repository Server, it MUST verify that the object
      to be withdrawn or replaced was retrieved from this same
      Repository Server before applying the appropriate action. Failing
      to do so will leave the Relying Party vulnerable to malicious
      Repository Servers instructing it to delete or change arbitrary
      objects.

   If any Delta File is rejected, Relying Parties MUST process the
   current Snapshot File instead, as described in Section 3.4.3.

3.4.3. Processing a Snapshot File

   Snapshot Files MUST only be used if Delta Files are unavailable or
   were rejected; for a description of the process, see Section 3.4.1.

   When the Relying Party downloads a Snapshot File, it MUST verify the
   file format and validation steps described in Section 3.5.2.3. If
   this verification fails, the file MUST be rejected.

   Furthermore, the Relying Party MUST verify that the hash of the
   contents of this file matches the hash on the Update Notification
   File that referenced it. In case of a mismatch of this hash, the
   file MUST be rejected.

   If a Relying Party retrieved a Snapshot File that is valid according
   to the above criteria, it performs the following actions:

   o The Relying Party MUST verify that the session_id matches the
      session_id of the Update Notification File. If the session_id
      values do not match, the file MUST be rejected.

   o The Relying Party MUST verify that the serial number of this
      Snapshot File is greater than the last processed serial number for
      this session_id. If this fails, the file MUST be rejected.

   o The Relying Party SHOULD then add all publish elements to a local
      storage and update its last processed serial number to the serial
      number of this Snapshot File.

   If a Snapshot File is rejected, it means that RRDP cannot be used.
   See Section 3.4.5 for considerations.

3.4.4. Polling the Update Notification File

   Once a Relying Party has learned about the location, session_id, and
   last processed serial number of the repository that uses the RRDP
   protocol, the Relying Party MAY start polling the Repository Server
   for updates. However, the Relying Party MUST NOT poll for updates
   more often than once every 1 minute, and in order to reduce data
   usage, Relying Parties MUST use the "If-Modified-Since" header
   explained in Section 3.3 of [RFC7232] in requests.

   If a Relying Party finds that updates are available, it SHOULD
   download and process the file as described in Section 3.4.1 and
   initiate a new RPKI object validation process. However, a detailed
   description of the RPKI object validation process itself is out of
   scope of this document.

3.4.5. Considerations Regarding Operational Failures in RRDP

   If a Relying Party experiences any issues with retrieving or
   processing any of the files used in this protocol, it will be unable
   to retrieve new RPKI data from the affected Repository Server.

   Relying Parties could attempt to use alternative repository access
   mechanisms, if they are available, according to the accessMethod
   element value(s) specified in the SIA of the associated certificate
   (see Section 4.8.8 of [RFC6487]).

   Furthermore, Relying Parties may wish to employ re-try strategies
   while fetching RRDP files. Relying Parties are also advised to keep
   old objects in their local cache so that validation can be done using
   old objects.

   It is also recommendable that re-validation and retrieval is
   performed pro-actively before manifests or CRLs go stale, or
   certificates expire, to ensure that problems on the side of the
   Relying Party can be identified and resolved before they cause major
   concerns.

3.5. File Definitions

3.5.1. Update Notification File

3.5.1.1. Purpose

   The Update Notification File is used by Relying Parties to discover
   whether any changes exist between the state of the repository and the
   Relying Party's cache. It describes the location of the files
   containing the snapshot and incremental deltas, which can be used by
   the Relying Party to synchronize with the repository.

3.5.1.2. Cache Concerns

   A Repository Server MAY use caching infrastructure to cache the
   Update Notification File and reduce the load of HTTPS requests.
   However, since this file is used by Relying Parties to determine
   whether any updates are available, the Repository Server SHOULD
   ensure that this file is not cached for longer than 1 minute. An
   exception to this rule is that it is better to serve a stale Update
   Notification File rather than no Update Notification File.

   How this is achieved exactly depends on the caching infrastructure
   used. In general, a Repository Server may find certain HTTP headers
   to be useful, such as: "Cache-Control: max-age=60" (see Section 5.2
   of [RFC7234]). Another approach can be to have the Repository Server

push out new versions of the Update Notification File to the caching
infrastructure when appropriate.

In case of a high load on a Repository Server or its distribution
network, the Cache-Control HTTP header, or a similar mechanism, MAY
be used to suggest an optimal (for the Repository Server) poll
interval for Relying Parties. However, setting it to an interval
longer than 1 hour is NOT RECOMMENDED. Relying parties SHOULD align
the suggested interval with their operational practices and the
expected update frequency of RPKI repository data and MAY discard the
suggested value.

3.5.1.3. File Format and Validation

Example Update Notification File:

```
<notification xmlns="http://www.ripe.net/rpki/rrdp"
      version="1"
      session_id="9df4b597-af9e-4dca-bdda-719cce2c4e28"
      serial="3">
  <snapshot uri="https://host/9d-8/3/snapshot.xml" hash="AB"/>
  <delta serial="3" uri="https://host/9d-8/3/delta.xml" hash="CD"/>
  <delta serial="2" uri="https://host/9d-8/2/delta.xml" hash="EF"/>
</notification>
```

Note: URIs and hash values in this example are shortened because of
formatting.

The following validation rules MUST be observed when creating or
parsing Update Notification Files:

o A Relying Party MUST reject any Update Notification File that is
   not well-formed or does not conform to the RELAX NG schema
   outlined in Section 3.5.4 of this document.

o The XML namespace MUST be "http://www.ripe.net/rpki/rrdp".

o The encoding MUST be "US-ASCII".

o The version attribute in the notification root element MUST be
   "1".

o The session_id attribute MUST be a random version 4 UUID
   [RFC4122], unique to this session.

o The serial attribute MUST be an unbounded, unsigned positive
   integer in decimal format indicating the current version of the
   repository.

   o The Update Notification File MUST contain exactly one 'snapshot'
      element for the current repository version.

   o If delta elements are included, they MUST form a contiguous
      sequence of serial numbers starting at a revision determined by
      the Repository Server, up to the serial number mentioned in the
      notification element. Note that the elements may not be ordered.

   o The hash attribute in snapshot and delta elements MUST be the
      hexadecimal encoding of the SHA-256 [SHS] hash of the referenced
      file. The Relying Party MUST verify this hash when the file is
      retrieved and reject the file if the hash does not match.

3.5.2. Snapshot File

3.5.2.1. Purpose

   A snapshot is intended to reflect the complete and current contents
   of the repository for a specific session and version. Therefore, it
   MUST contain all objects from the repository current as of the time
   of the publication.

3.5.2.2. Cache Concerns

   A snapshot reflects the content of the repository at a specific point
   in time; for that reason, it can be considered immutable data.
   Snapshot Files MUST be published at a URL that is unique to the
   specific session and serial.

   Because these files never change, they MAY be cached indefinitely.
   However, in order to prevent these files from using a lot of space in
   the caching infrastructure, it is RECOMMENDED that a limited interval
   is used in the order of hours or days.

   To avoid race conditions where a Relying Party downloads an Update
   Notification File moments before it's updated, Repository Servers
   SHOULD retain old Snapshot Files for at least 5 minutes after a new
   Update Notification File is published.

3.5.2.3. File Format and Validation

   Example Snapshot File:

```
<snapshot xmlns="http://www.ripe.net/rpki/rrdp"
        version="1"
        session_id="9df4b597-af9e-4dca-bdda-719cce2c4e28"
        serial="2">
  <publish uri="rsync://rpki.ripe.net/Alice/Bob.cer">
    ZXhhbXBsZTE=
  </publish>
  <publish uri="rsync://rpki.ripe.net/Alice/Alice.mft">
    ZXhhbXBsZTI=
  </publish>
  <publish uri="rsync://rpki.ripe.net/Alice/Alice.crl">
    ZXhhbXBsZTM=
  </publish>
</snapshot>
```

   The following rules MUST be observed when creating or parsing
   Snapshot Files:

   o A Relying Party MUST reject any Snapshot File that is not well-
      formed or does not conform to the RELAX NG schema outlined in
      Section 3.5.4 of this document.

   o The XML namespace MUST be "http://www.ripe.net/rpki/rrdp".

   o The encoding MUST be "US-ASCII".

   o The version attribute in the notification root element MUST be
      "1".

   o The session_id attribute MUST match the expected session_id in the
      reference in the Update Notification File.

   o The serial attribute MUST match the expected serial in the
      reference in the Update Notification File.

   o Note that the publish element is similar to the publish element
      defined in the publication protocol [RFC8181]. However, the "tag"
      attribute is not used here because it is not relevant to Relying
      Parties. The "hash" attribute is not used here because this file
      represents a complete current state of the repository; therefore,
      it is not relevant to know which existing RPKI object (if any) is
      updated.

3.5.3. Delta File

3.5.3.1. Purpose

   An incremental Delta File contains all changes for exactly one serial
   increment of the Repository Server. In other words, a single delta
   will typically include all the new objects, updated objects, and
   withdrawn objects that a Certification Authority sent to the
   Repository Server. In its simplest form, the update could concern
   only a single object, but it is RECOMMENDED that CAs send all changes
   for one of their key pairs (updated objects as well as a new manifest
   and CRL) as one atomic update message.

3.5.3.2. Cache Concerns

   Deltas reflect the difference between two consecutive versions of a
   repository for a given session. For that reason, deltas can be
   considered immutable data. Delta Files MUST be published at a URL
   that is unique to the specific session and serial.

   Because these files never change, they MAY be cached indefinitely.
   However, in order to prevent these files from using a lot of space in
   the caching infrastructure, it is RECOMMENDED that a limited interval
   is used in the order of hours or days.

   To avoid race conditions where a Relying Party downloads an Update
   Notification File moments before it's updated, Repository Servers
   SHOULD retain old Delta Files for at least 5 minutes after they are
   no longer included in the latest Update Notification File.

3.5.3.3. File Format and Validation

   Example Delta File:

```
      <delta xmlns="http://www.ripe.net/rpki/rrdp"
             version="1"
             session_id="9df4b597-af9e-4dca-bdda-719cce2c4e28"
             serial="3">
        <publish uri="rsync://rpki.ripe.net/repo/Alice/Alice.mft"
              hash="50d8...545c">
          ZXhhbXBsZTQ=
        </publish>
        <publish uri="rsync://rpki.ripe.net/repo/Alice/Alice.crl"
              hash="5fb1...6a56">
          ZXhhbXBsZTU=
        </publish>
        <withdraw uri="rsync://rpki.ripe.net/repo/Alice/Bob.cer"
                hash="caeb...15c1"/>
      </delta>
```

   Note that a formal RELAX NG specification of this file format is
   included later in this document. A Relying Party MUST NOT process
   any Delta File that is incomplete or not well-formed.

   The following validation rules MUST be observed when creating or
   parsing Delta Files:

   o A Relying Party MUST reject any Delta File that is not well-formed
      or does not conform to the RELAX NG schema outlined in
      Section 3.5.4 of this document.

   o The XML namespace MUST be "http://www.ripe.net/rpki/rrdp".

   o The encoding MUST be "US-ASCII".

   o The version attribute in the delta root element MUST be "1".

   o The session_id attribute MUST be a random version 4 UUID unique to
      this session.

   o The session_id attribute MUST match the expected session_id in the
      reference in the Update Notification File.

   o The serial attribute MUST match the expected serial in the
      reference in the Update Notification File.

   o Note that the publish element is similar to the publish element
      defined in the publication protocol [RFC8181]. However, the "tag"
      attribute is not used here because it is not relevant to Relying
      Parties.

3.5.4. XML Schema

   The following is a RELAX NG compact form schema describing version 1
   of this protocol.

```
   #
   # RELAX NG schema for the RPKI Repository Delta Protocol (RRDP).
   #

   default namespace = "http://www.ripe.net/rpki/rrdp"

   version = xsd:positiveInteger    { maxInclusive="1" }
   serial  = xsd:positiveInteger
   uri     = xsd:anyURI
   uuid    = xsd:string                { pattern = "[\-0-9a-fA-F]+" }
   hash    = xsd:string                { pattern = "[0-9a-fA-F]+" }
   base64  = xsd:base64Binary

   # Notification File: lists current snapshots and deltas.

   start |= element notification {
     attribute version    { version },
     attribute session_id { uuid },
     attribute serial     { serial },
     element snapshot {
       attribute uri  { uri },
       attribute hash { hash }
     },
     element delta {
       attribute serial { serial },
       attribute uri    { uri },
       attribute hash   { hash }
     }*
   }

   # Snapshot segment: think DNS AXFR.

   start |= element snapshot {
     attribute version    { version },
     attribute session_id { uuid },
     attribute serial     { serial },
     element publish      {
       attribute uri { uri },
```

```
      base64
    }*
  }

  # Delta segment: think DNS IXFR.

  start |= element delta {
    attribute version    { version },
    attribute session_id { uuid },
    attribute serial     { serial },
    delta_element+
  }

  delta_element |= element publish  {
    attribute uri  { uri },
    attribute hash { hash }?,
    base64
  }

  delta_element |= element withdraw {
    attribute uri  { uri },
    attribute hash { hash }
  }

  # Local Variables:
  # indent-tabs-mode: nil
  # comment-start: "# "
  # comment-start-skip: "#[ \t]*"
  # End:
```

4. Operational Considerations

4.1. Compatibility with previous standards

   This protocol has been designed to replace rsync as a distribution
   mechanism of an RPKI repository. However, it is also designed to
   coexist with existing implementations based on rsync, to enable
   smooth transition from one distribution mechanism to another.

   For every repository object listed in the Snapshot and Delta Files,
   both the hash of the object's content and the rsync URI [RFC5781] of
   its location in the repository are listed. This makes it possible to
   distribute the same RPKI repository, represented by a set of files on
   a filesystem, using both rsync and RRDP. It also enables Relying
   Parties tools to query, combine, and consequently validate objects
   from repositories of different types.

4.2. Distribution Considerations

   One of the design goals of RRDP was to minimize load on a Repository
   Server while serving clients. To achieve this, neither the content
   nor the URLs of the Snapshot and Delta Files are modified after they
   have been published in the Update Notification File. This allows
   their effective distribution by using either a single HTTP server or
   a CDN.

   The RECOMMENDED way for Relying Parties to keep up with the
   repository updates is to poll the Update Notification File for
   changes. The content of that file is updated with every new serial
   version of a repository (while its URL remains stable). To
   effectively implement distribution of the Update Notification File,
   an "If-Modified-Since" HTTP request header is required to be present
   in all requests for the Update Notification File (see Section 3.4.4).
   Therefore, it is RECOMMENDED that Relying Party tools implement a
   mechanism to keep track of a previous successful fetch of an Update
   Notification File.

   Implementations of RRDP should also take care of not producing new
   versions of the repository (and subsequently, new Update
   Notification, Snapshot, and Delta Files) too often. Usually the
   maintenance of the RPKI repository includes regular updates of
   manifest and CRL objects performed on a schedule. This often results
   in bursts of repository updates during a short period of time. Since
   the Relying Parties are required to poll for the Update Notification
   File not more often than once per minute (Section 3.4.4), it is not
   practical to generate new serial versions of the repository much more
   often than 1 per minute. It is allowed to combine multiple updates,
   possibly from different CAs, into a new serial repository version
   (Section 3.3.2). This will significantly shorten the size of the
   Update Notification File and total amount of data distributed to all
   Relying Parties.

4.3. HTTPS Considerations

   Note that a Man in the Middle (MITM) cannot produce validly signed
   RPKI data but can perform withhold or replay attacks targeting a
   Relying Party and keep the Relying Party from learning about changes
   in the RPKI. Because of this, Relying Parties SHOULD do TLS
   certificate and host name validation when they fetch from an RRDP
   Repository Server.

   Relying Party tools SHOULD log any TLS certificate or host name
   validation issues found, so that an operator can investigate the
   cause. However, such validation issues are often due to
   configuration errors or a lack of a common TLS trust anchor. In

these cases, it is better if the Relying Party retrieves the signed
RPKI data regardless and performs validation on it. Therefore, the
Relying Party MUST continue to retrieve the data in case of errors.
The Relying Party MAY choose to log encountered issues only when
fetching the Update Notification File, but not when it subsequently
fetches Snapshot or Delta Files from the same host. Furthermore, the
Relying Party MAY provide a way for operators to accept untrusted
connections for a given host, after the cause has been identified.

It is RECOMMENDED that Relying Parties and Repository Servers follow
the Best Current Practices outlined in [RFC7525] on the use of HTTP
over TLS (HTTPS) [RFC7230]. Relying Parties SHOULD do TLS
certificate and host name validation using subjectAltName dNSName
identities as described in [RFC6125]. The rules and guidelines
defined in [RFC6125] apply here, with the following considerations:

o Relying Parties and Repository Servers SHOULD support the DNS-ID
   identifier type. The DNS-ID identifier type SHOULD be present in
   Repository Server certificates.

o DNS names in Repository Server certificates SHOULD NOT contain the
   wildcard character "*".

o A Common Name (CN) field may be present in a Repository Server
   certificate's subject name but SHOULD NOT be used for
   authentication within the rules described in [RFC6125].

o This protocol does not require the use of SRV-IDs.

o This protocol does not require the use of URI-IDs.

Note, however, that this validation is done on a best-effort basis
and serves to highlight potential issues, but RPKI object security
does not depend on this. Therefore, Relying Parties MAY deviate from
the validation steps listed above.

5. Security Considerations

RRDP deals exclusively with the transfer of RPKI objects from a
Repository Server to a Relying Party. The trust relation between a
Certificate Authority and its Repository Server is out of scope for
this document. However, it should be noted that from a Relying Party
point of view, all RPKI objects (certificates, CRLs, and objects
wrapped in Cryptographic Message Syntax (CMS)) are already covered by
object security mechanisms including signed manifests. This allows
validation of these objects even though the Repository Server itself
is not trusted. This document makes no change to RPKI validation
procedures per se.

   The original RPKI transport protocol is rsync, which offers no
   channel security mechanism. RRDP replaces the use of rsync by HTTPS;
   while the channel security mechanism underlying RRDP (HTTPS) is not a
   cure-all, it does make some forms of denial-of-service attacks more
   difficult for the attacker. HTTPS issues are discussed in more
   detail in Section 4.3.

   Supporting both RRDP and rsync necessarily increases the number of
   opportunities for a malicious RPKI Certificate Authority to perform
   denial-of-service attacks on Relying Parties, by expanding the number
   of URIs which the Relying Party may need to contact in order to
   complete a validation run. However, other than the relative cost of
   HTTPS versus rsync, adding RRDP to the mix does not change this
   picture significantly: with either RRDP or rsync a malicious
   Certificate Authority can supply an effectively infinite series of
   URIs for the Relying Party to follow. The only real solution to this
   is for the Relying Party to apply some kind of bound to the amount of
   work it is willing to do. Note also that the attacker in this
   scenario must be an RPKI Certificate Authority; otherwise, the normal
   RPKI object security checks would reject the malicious URIs.

   Processing costs for objects retrieved using RRDP may be somewhat
   different from the same objects retrieved using rsync: because RRDP
   treats an entire set of changes as a unit (one "delta"), it may not
   be practical to start processing any of the objects in the delta
   until the entire delta has been received. With rsync, by contrast,
   incremental processing may be easy, but the overall cost of transfer
   may be higher, as may be the number of corner cases in which the
   Relying Party retrieves some but not all of the updated objects.
   Overall, RRDP's behavior is closer to a proper transactional system,
   which (probably) leads to an overall reliability increase.

   RRDP is designed to scale much better than rsync. In particular,
   RRDP is designed to allow use of an HTTPS caching infrastructure to
   reduce load on primary Repository Servers and increase resilience
   against denial-of-service attacks on the RPKI publication service.

6. IANA Considerations

   IANA has updated the reference for id-ad-rpkiNotify to point to this
   document in the "SMI Security for PKIX Access Descriptor" registry
   [IANA-AD-NUMBERS].

7. References

7.1. 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,
              <http://www.rfc-editor.org/info/rfc2119>.

   [RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally
              Unique IDentifier (UUID) URN Namespace", RFC 4122,
              DOI 10.17487/RFC4122, July 2005,
              <http://www.rfc-editor.org/info/rfc4122>.

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

   [RFC6125] Saint-Andre, P. and J. Hodges, "Representation and
              Verification of Domain-Based Application Service Identity
              within Internet Public Key Infrastructure Using X.509
              (PKIX) Certificates in the Context of Transport Layer
              Security (TLS)", RFC 6125, DOI 10.17487/RFC6125, March
              2011, <http://www.rfc-editor.org/info/rfc6125>.

   [RFC6481] Huston, G., Loomans, R., and G. Michaelson, "A Profile for
              Resource Certificate Repository Structure", RFC 6481,
              DOI 10.17487/RFC6481, February 2012,
              <http://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,
              <http://www.rfc-editor.org/info/rfc6487>.

   [RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer
              Protocol (HTTP/1.1): Message Syntax and Routing",
              RFC 7230, DOI 10.17487/RFC7230, June 2014,
              <http://www.rfc-editor.org/info/rfc7230>.

   [RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer
              Protocol (HTTP/1.1): Semantics and Content", RFC 7231,
              DOI 10.17487/RFC7231, June 2014,
              <http://www.rfc-editor.org/info/rfc7231>.

   [RFC7232] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer
              Protocol (HTTP/1.1): Conditional Requests", RFC 7232,
              DOI 10.17487/RFC7232, June 2014,
              <http://www.rfc-editor.org/info/rfc7232>.

   [RFC7234] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke,
              Ed., "Hypertext Transfer Protocol (HTTP/1.1): Caching",
              RFC 7234, DOI 10.17487/RFC7234, June 2014,
              <http://www.rfc-editor.org/info/rfc7234>.

   [RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre,
              "Recommendations for Secure Use of Transport Layer
              Security (TLS) and Datagram Transport Layer Security
              (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May
              2015, <http://www.rfc-editor.org/info/rfc7525>.

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

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

   [SHS]   National Institute of Standards and Technology, "Secure
              Hash Standard (SHS)", FIPS PUB 180-4,
              DOI 10.6028/NIST.FIPS.180-4, August 2015,
              <http://nvlpubs.nist.gov/nistpubs/FIPS/
              NIST.FIPS.180-4.pdf>.

7.2. Informative References

   [IANA-AD-NUMBERS]
              IANA, "Structure of Management Information (SMI) Numbers
              (MIB Module Registrations)",
              <http://www.iana.org/assignments/smi-numbers>.

   [RFC6486] Austein, R., Huston, G., Kent, S., and M. Lepinski,
              "Manifests for the Resource Public Key Infrastructure
              (RPKI)", RFC 6486, DOI 10.17487/RFC6486, February 2012,
              <http://www.rfc-editor.org/info/rfc6486>.

   [RFC6488] Lepinski, M., Chi, A., and S. Kent, "Signed Object
              Template for the Resource Public Key Infrastructure
              (RPKI)", RFC 6488, DOI 10.17487/RFC6488, February 2012,
              <http://www.rfc-editor.org/info/rfc6488>.

   [RSYNC]   "rsync", <https://rsync.samba.org>.

Authors' Addresses

   Tim Bruijnzeels
   RIPE NCC

   Email: tim@ripe.net

   Oleg Muravskiy
   RIPE NCC

   Email: oleg@ripe.net

   Bryan Weber
   Cobenian

   Email: bryan@cobenian.com

   Rob Austein
   Dragon Research Labs

   Email: sra@hactrn.net