Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 13, 2012

S. Weiler
A. Sonalker
SPARTA, Inc.
R. Austein
Dragon Research Labs

March 12, 2012

A Publication Protocol for the Resource Public Key Infrastructure (RPKI)

draft-ietf-sidr-publication-02

Abstract

This document defines a protocol for publishing Resource Public Key Infrastructure (RPKI) objects. Even though the RPKI will have many participants issuing certificates and creating other objects, it is operationally useful to consolidate the publication of those objects. This document provides the protocol for doing so.

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 http://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 September 13, 2012.

Copyright Notice

Table of Contents

1. Introduction

This document assumes a working knowledge of the Resource Public Key Infrastructure (RPKI), which is intended to support improved routing security on the Internet. [RFC6480]

In order to make participation in the RPKI easier, it is helpful to have a few consolidated repositories for RPKI objects, thus saving every participant from the cost of maintaining a new service. Similarly, relying parties using the RPKI objects will find it faster and more reliable to retrieve the necessary set from a smaller number of repositories.

These consolidated RPKI object repositories will in many cases be outside the administrative scope of the organization issuing a given RPKI object. Hence the need for a protocol to publish RPKI objects.

This document defines the RPKI publication protocol, including a sub-protocol for configuring the publication engine.

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

"Publication engine" and "publication server" are used interchangeably to refer to the server providing the service described in this document.

"Business Public Key Infrastructure" ("Business PKI" or "BPKI") refers to a PKI, separate from the RPKI, used to authenticate clients to the publication engine.

2. Context

This protocol was designed specifically for the case where an internet registry, already issuing RPKI certificates to its children, also wishes to run a publication service for its children.

We use the term "Business PKI" here because an internet registry might already have a PKI, separate from the RPKI, for authenticating its clients and might wish to reuse that PKI for this protocol. Such reuse is not a requirement.

3. Protocol Specification

In summary, the publication protocol uses XML messages wrapped in CMS, carried over HTTP transport.

The publication procotol consists of two separate subprotocols. The first is a control protocol used to configure a publication engine. The second subprotocol, which we refer to by the overloaded term "publication protocol", is used to request publication of specific objects. The publication engine operates a single HTTP server on a single port. It distinguishes between the two protocols by using different URLs for them.

3.1. Common Details

This section discusses details that the two subprotocols have in common, including the transport and CMS wrappers.

Both protocols use a simple request/response interaction. The client passes a request to the server, and the server generates a corresponding response.

A message exchange commences with the client initiating an HTTP POST with content type of "application/rpki-publication", with the message object as the body. The server's response will similarly be the body of the response with a content type of "application/ rpki-publication".

The content of the POST and the server's response will be a well-

formed Cryptographic Message Syntax (CMS) [RFC5652] object with OID = 1.2.840.113549.1.7.2 as described in Section 3.1 of [RFC6492].

3.1.1. Common XML Message Format

The XML schema for this protocol (including both subprotocols) is below in Section 3.5. Both subprotocols use the same basic XML message format, which looks like:


```
<?xml version='1.0' encoding='us-ascii'?>
<msg xmlns="http://www.hactrn.net/uris/rpki/publication-spec/"
   version="2"
   type="message type">
   [one or more PDUs]
</msg>
```

version:
The value of this attribute is the version of this protocol. This document describes version 2.

type:
The possible values of this attribute are "reply" and "query".

A query PDU may be one of four types: config_query, client_query, publish_query, or withdraw_query. The first two are used by the control sub-protocol, the latter two by the publication sub-protocol.

A reply PDU may be one of five types: config_reply, client_reply, publish_reply, withdraw_reply, or report_error_reply.

Each of these PDUs may include an optional tag to facilitate bulk

operation. If a tag is set in a query PDU, the corresponding reply(s) MUST have the tag attribute set to the same value.

3.2. Control Sub-Protocol

The control sub-protocol is used to configure a publication server. It can set global variables (at the moment, limited to a BPKI CRL) and manage clients who are allowed to publish data on the server.

3.2.1. Config Object

The <config/> object allows configuration of data that apply to the entire publication server rather than a particular client. There is exactly one <config/> object in the publication server, and it only supports the "set" and "get" actions -- it cannot be created or destroyed. Its use is typically restricted to the repository operator.

The <config/> object only has one data element that can be set: the bpki_crl. This is used by the publication server when authenticating clients.

3.2.2. Client Object

Unlike the <config/> object, the <client/> object represents one client authorized to use the publication server. There may be more

than one <client/> object on each publication server. Again, its use is typically restricted to the respository operator.

The <client/> object supports five actions: "create", "set", "get", "list", and "destroy". Each client has a "client_handle" attribute, which is used in responses and must be specified in "create", "set", "get", or "destroy" actions.

Payload data which can be configured in a <client/> object include:

- base_uri (attribute): This attribute represents the base URI below which the client will be allowed to publish data. Additional constraints may be imposed by the publication server in certain cases, for e.g., a child publishing directly under its parent.

- bpki_cert (element): This represents the X.509 BPKI CA certificate for this client. This should be used as part of the certificate chain when validating incoming CMS messages. Two valid approaches exist. If the optional bpki_glue certificate is being used, then the bpki_cert certificate should be issued by the bpki_glue certificate; otherwise, the bpki_cert certificate should be issued by the publication engine's bpki_ta certificate.

- bpki_glue (element): This is an additional (optional) type of X.509 certificate for this client. It may be used in certain pathological cross-certification cases which require a two-certificate chain due to issuer name conflicts. When being used, issuing order is that the bpki_glue certificate should be the issuer of the bpki_cert certificate. Otherwise, it should be issued by the publication engine's bpki_ta certificate. Since this is an optional use certificate, it may be left unset if not needed.

3.3. Publication Sub-Protocol

The publication sub-protocol requests publication or withdrawal from publication of RPKI objects.

The publication protocol uses a common message format to request publication of any RPKI object. This format was chosen specifically to allow this protocol to accommodate new types of RPKI objects without needing changes to this protocol.

Both the <publish/> and <withdraw/> objects have a payload of an optional tag and a URI. The <publish/> query also contains the DER object to be published, encoded in Base64.

Note that every publish and withdraw action requires a new manifest,

thus every publish or withdraw action will involve at least two objects.

3.4. Error handling

Errors are handled similarly in both subprotocols, and they're handled at two levels.

Since all messages in this protocol are conveyed over HTTP connections, basic errors are indicated via the HTTP response code. 4xx and 5xx responses indicate that something bad happened. Errors that make it impossible to decode a query or encode a response are handled in this way.

Where possible, errors will result in an XML <report_error/> message which takes the place of the expected protocol response message. <report_error/> messages are CMS-signed XML messages like the rest of this protocol, and thus can be archived to provide an audit trail.

<report_error/> messages only appear in replies, never in queries. The <report_error/> message can appear in both the control and publication subprotocols.

Like all other messages in this protocol, the <report_error/> message includes a "tag" attribute to assist in matching the error with a particular query when using batching. It is optional to set the tag on queries but, if set on the query, it MUST be set on the reply or error.

The error itself is conveyed in the error_code (attribute). The value of this attribute is a token indicating the specific error that occurred.

The body of the <report_error/> element itself is an optional text string; if present, this is debugging information.

3.5. XML Schema

The following is a RelaxNG compact form schema describing the Publication Protocol.

```
default namespace = "http://www.hactrn.net/uris/rpki/publication-spec/"

    # Top level PDU
    start = element msg {
       attribute version { "2" } ,
       ( ( attribute type { "query" }, query_elt*) |
```



```
       (attribute type { "reply" }, reply_elt*))
```

```
}

# PDUs allowed in a query
query_elt = ( config_query | client_query | publish_query |
   withdraw_query )

# PDUs allowed in a reply
reply_elt = ( config_reply | client_reply | publish_reply |
   withdraw_reply | report_error_reply )

# Tag attributes for bulk operations
tag = attribute tag { xsd:token {maxLength="1024" } }

# Base64 encoded DER stuff
base64 = xsd:base64Binary

# Publication URLs
uri_t = xsd:anyURI { maxLength="4096" }
uri = attribute uri { uri_t }

# Handles on remote objects (replaces passing raw SQL IDs).  NB:
# Unlike the up-down protocol, handles in this protocol allow
# "/" as a hierarchy delimiter.
object_handle = xsd:string {
   maxLength="255" pattern="[\-_A-Za-z0-9/]*" }

# <config/> element (use restricted to repository operator)
# config_handle attribute: create, list, and destroy commands
# omitted deliberately.
config_payload = (element bpki_crl { base64 }?)
config_query |= element config { attribute action { "set" }, tag?,
   config_payload }
config_reply |= element config { attribute action { "set" }, tag? }
config_query |= element config { attribute action { "get" }, tag? }
config_reply |= element config { attribute action { "get" }, tag?,
   config_payload }

# <client/> element (use restricted to repository operator)
client_handle = attribute client_handle { object_handle }
client_payload = (attribute base_uri { uri_t }?, element bpki_cert {
   base64 }?, element bpki_glue { base64 }?)
client_query |= element client { attribute action { "create" },
   tag?, client_handle, client_payload }
client_reply |= element client { attribute action { "create" },
   tag?, client_handle }
client_query |= element client { attribute action { "set" }, tag?,
   client_handle, client_payload }
```

```
    client_reply |= element client { attribute action { "set" }, tag?,
       client_handle }
    client_query |= element client { attribute action { "get" }, tag?,
       client_handle }
    client_reply |= element client { attribute action { "get" }, tag?,
       client_handle, client_payload }
    client_query |= element client { attribute action { "list" }, tag? }
    client_reply |= element client { attribute action { "list" }, tag?,
       client_handle, client_payload }
    client_query |= element client { attribute action { "destroy" },
       tag?, client_handle }
    client_reply |= element client { attribute action { "destroy" },
       tag?, client_handle }

    # <publish/> element
    publish_query |= element publish { tag?, uri, base64 }
    publish_reply |= element publish { tag?, uri }

    # <withdraw/> element
    withdraw_query |= element withdraw { tag?, uri }
    withdraw_reply |= element withdraw { tag?, uri }

    # <report_error/> element
    error = xsd:token { maxLength="1024" }
    report_error_reply = element report_error {
       tag?,
       attribute error_code { error },
       xsd:string { maxLength="512000" }?
       }
```

4. Examples

Following are various queries and the corresponding replies for the RPKI publication protocol

4.1. Config Set Query and Response

A. Config "Set" Query

```
<msg xmlns="http://www.hactrn.net/uris/rpki/publication-spec/"
    type="query" version="2">
    <config action="set">
    <bpki_crl>
    MIIBezBlAgEBMA0GCSqGSIb3DQEBCwUAMCMxITAfBgNVBAMTGFRlc3QgQ2Vyd
    GlmaWNhdGUgcHViZCBUQRcNMDgwNjAyMjE0OTQ1WhcNMDgwNzAyMjE0OTQ1Wq
    AOMAwwCgYDVR0UBAMCAQEwDQYJKoZIhvcNAQELBQADggEBAFWCWgBl4ljVqX/
    CHo+RpqYtvmKMnjPVflMXUB7i28RGP4DAq4l7deDU7Q82xEJyE4TXMWDWAV6U
    G6uUGum0VHWOcj9ohqyiZUGfOsKg2hbwkETm8sAENOsi1yNdyKGk6jZ16aF5f
    ubxQqZa1pdGCSac1/ZYC5sLLhEz3kmz+B9z9mXFVc5TgAh4dN3Gy5ftF8zZAF
    pDGnS4biCnRVqhGv6R0Lh/5xmii+ZU6kNDhbeMsjJg+ZOmtN+wMeHSIbjiy0W
    uuaZ3k2xSh0C94anrHBZAvvCRhbazjR0Ef5OMZ5lcllw3uO8IHuoisHKkehy4
    Y0GySdj98fV+OuiRTH9vt/M=
    </bpki_crl>
    </config>
    </msg>
```

B. Config "Set" Reply

```
<msg xmlns="http://www.hactrn.net/uris/rpki/publication-spec/"
type="reply" version="2">
<config action="set"/>
     </msg>
```

4.2. Config Get Query and Response

A. Config "Get" Query

```
<msg xmlns="http://www.hactrn.net/uris/rpki/publication-spec/"
    type="query" version="2">
    <config action="get"/>
    </msg>
```

B. Config "Get" Reply

```
<msg xmlns="http://www.hactrn.net/uris/rpki/publication-spec/"
    type="reply" version="2">
    <config action="get">
    <bpki_crl>
    MIIBezBlAgEBMA0GCSqGSIb3DQEBCwUAMCMxITAfBgNVBAMTGFRlc3QgQ2Vyd
    GlmaWNhdGUgcHViZCBUQRcNMDgwNjAyMjE0OTQ1WhcNMDgwNzAyMjE0OTQ1Wq
    AOMAwwCgYDVR0UBAMCAQEwDQYJKoZIhvcNAQELBQADggEBAFWCWgBl4ljVqX/
    CHo+RpqYtvmKMnjPVflMXUB7i28RGP4DAq4l7deDU7Q82xEJyE4TXMWDWAV6U
    G6uUGum0VHWOcj9ohqyiZUGfOsKg2hbwkETm8sAENOsi1yNdyKGk6jZ16aF5f
    ubxQqZa1pdGCSac1/ZYC5sLLhEz3kmz+B9z9mXFVc5TgAh4dN3Gy5ftF8zZAF
    pDGnS4biCnRVqhGv6R0Lh/5xmii+ZU6kNDhbeMsjJg+ZOmtN+wMeHSIbjiy0W
    uuaZ3k2xSh0C94anrHBZAvvCRhbazjR0Ef5OMZ5lcllw3uO8IHuoisHKkehy4
    Y0GySdj98fV+OuiRTH9vt/M=
</bpki_crl>
    </config>
    </msg>
```

4.3. Example 3: Client Create Query and Reply

A. Client "Create" Query

```
<msg xmlns="http://www.hactrn.net/uris/rpki/publication-spec/"
    version="2" type="query">
    <client action="create" client_handle="3"
    base_uri="rsync://wombat.invalid/">
    <bpki_cert>
    MIIDGzCCAgOgAwIBAgIJAKi+/+wUhQlxMA0GCSqGSIb3DQEBBQUAMCQxIjAgB
    gNVBAMTGVRlc3QgQ2VydGlmaWNhdGUgQm9iIFJvb3QwHhcNMDcwODAxMTk1Mz
    EwWhcNMDcwODMxMTk1MzEwWjAkMSIwIAYDVQQDExlUZXN0IENlcnRpZmljYXR
    lIEJvYiBSb290MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEArKYU
    tJaM5PH5917SG2ACc7iBYdQO2HYyu8Gb6i9Q2Gxc3cWEX7RTBvgOL79pWf3GI
```

```
dnoupzMnoZVtY3GUx2G/0WkmLui2TCeDhcfXdQ4rcp8J3V/6ESj+yuEPPOG8U
N17mUKKgujrch6ZvgCDO9AyOK/uXu+ABQXTPsn2pVe2EVh3V004ShLi8GKgVd
qb/rW/6GTg0Xb/zLT6WWMuT++6sXTlztJdQYkRamJvKfQDU1naC8mAkGf79Tb
a0xyBGAUII0GfREY6t4/+NAP2Yyb3xNlBqcJoTov0JfNKHZcCZePr79j7LK/h
kZxxip+Na9xDpE+oQRV+DRukCRJdiqg+wIDAQABo1AwTjAMBgNVHRMEBTADAQ
H/MB0GA1UdDgQWBBTDEsXJe6pjAQD4ULlB7+GMDBlimTAfBgNVHSMEGDAWgBT
DEsXJe6pjAQD4ULlB7+GMDBlimTANBgkqhkiG9w0BAQUFAAOCAQEAWWkNcW6S
1tKKqtzJsdfhjJiAAPQmOXJskv0ta/8f6Acgcum1YieNdtT0n96P7CUHOWP8Q
Bb91JzeewR7b6WJLwb1Offs3wNq3kk75pJe89r4XY39EZHhMW+Dv0PhIKu2Cg
D4LeyH1FVTQkF/QObGEmkn+s+HTsuzd1l2VLwcP1Smsqep6LAlFj62qqaIJzN
eQ9NVkBqtkygnYlBOkaBTHfQTux3jYNpEo8JJB5e/WFdHYyMNrG2xMOtIC7T4
+IOHgT8PgrNhaeDg9ctewj0X8Qi9nI9nXeinicLX8vj6hdEq3ORv7RZMJNYqv
1HQ3wUE2B7fCPFv7EUwzaCds1kgRQ==
</bpki_cert>
</client>
</msg>
```


B. Client "Create" Reply

```
<msg xmlns="http://www.hactrn.net/uris/rpki/publication-spec/"
    version="2" type="reply">
    <client action="create" client_handle="3"/>
    </msg>
```

4.4. Example 4: Client Set Query and Reply

A. Client "Set" Query

```
<msg xmlns="http://www.hactrn.net/uris/rpki/publication-spec/"
    version="2" type="query">
    <client action="set" client_handle="3">
    <bpki_glue>
    MIIDGzCCAgOgAwIBAgIJAKi+/+wUhQlxMA0GCSqGSIb3DQEBBQUAMCQxIjAgB
    gNVBAMTGVRlc3QgQ2VydGlmaWNhdGUgQm9iIFJvb3QwHhcNMDcwODAxMTk1Mz
    EwWhcNMDcwODMxMTk1MzEwWjAkMSIwIAYDVQQDExlUZXN0IENlcnRpZmljYXR
    lIEJvYiBSb290MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEArKYU
    tJaM5PH5917SG2ACc7iBYdQO2HYyu8Gb6i9Q2Gxc3cWEX7RTBvgOL79pWf3GI
    dnoupzMnoZVtY3GUx2G/0WkmLui2TCeDhcfXdQ4rcp8J3V/6ESj+yuEPPOG8U
    N17mUKKgujrch6ZvgCDO9AyOK/uXu+ABQXTPsn2pVe2EVh3V004ShLi8GKgVd
    qb/rW/6GTg0Xb/zLT6WWMuT++6sXTlztJdQYkRamJvKfQDU1naC8mAkGf79Tb
    a0xyBGAUII0GfREY6t4/+NAP2Yyb3xNlBqcJoTov0JfNKHZcCZePr79j7LK/h
    kZxxip+Na9xDpE+oQRV+DRukCRJdiqg+wIDAQABo1AwTjAMBgNVHRMEBTADAQ
```

```
            H/MB0GA1UdDgQWBBTDEsXJe6pjAQD4ULlB7+GMDBlimTAfBgNVHSMEGDAWgBT
            DEsXJe6pjAQD4ULlB7+GMDBlimTANBgkqhkiG9w0BAQUFAAOCAQEAWWkNcW6S
            1tKKqtzJsdfhjJiAAPQmOXJskv0ta/8f6Acgcum1YieNdtT0n96P7CUHOWP8Q
            Bb91JzeewR7b6WJLwb1Offs3wNq3kk75pJe89r4XY39EZHhMW+Dv0PhIKu2Cg
            D4LeyH1FVTQkF/QObGEmkn+s+HTsuzd1l2VLwcP1Smsqep6LAlFj62qqaIJzN
            eQ9NVkBqtkygnYlBOkaBTHfQTux3jYNpEo8JJB5e/WFdHYyMNrG2xMOtIC7T4
            +IOHgT8PgrNhaeDg9ctewj0X8Qi9nI9nXeinicLX8vj6hdEq3ORv7RZMJNYqv
            1HQ3wUE2B7fCPFv7EUwzaCds1kgRQ==
            </bpki_glue>
            </client>
            </msg>
```

B. Client "Set" Reply

```
        <msg xmlns="http://www.hactrn.net/uris/rpki/publication-spec/"
            version="2" type="reply">
            <client action="set" client_handle="3"/>
            </msg>
```


4.5. Example 5: Client Get Query and Reply

A. Client "Get" Query

```
        <msg xmlns="http://www.hactrn.net/uris/rpki/publication-spec/"
            version="2" type="query">
            <client action="get" client_handle="3"/>
            </msg>
```

B. Client "Get" Reply

```
        <msg xmlns="http://www.hactrn.net/uris/rpki/publication-spec/"
            version="2" type="reply">
            <client action="get" client_handle="3"
            base_uri="rsync://wombat.invalid/">
            <bpki_cert>
            MIIDGzCCAgOgAwIBAgIJAKi+/+wUhQlxMA0GCSqGSIb3DQEBBQUAMCQxIjAgB
            gNVBAMTGVRlc3QgQ2VydGlmaWNhdGUgQm9iIFJvb3QwHhcNMDcwODAxMTk1Mz
```

```
EwWhcNMDcwODMxMTk1MzEwWjAkMSIwIAYDVQQDExlUZXN0IENlcnRpZmljYXR
lIEJvYiBSb290MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEArKYU
tJaM5PH5917SG2ACc7iBYdQO2HYyu8Gb6i9Q2Gxc3cWEX7RTBvgOL79pWf3GI
dnoupzMnoZVtY3GUx2G/0WkmLui2TCeDhcfXdQ4rcp8J3V/6ESj+yuEPPOG8U
N17mUKKgujrch6ZvgCDO9AyOK/uXu+ABQXTPsn2pVe2EVh3V004ShLi8GKgVd
qb/rW/6GTg0Xb/zLT6WWMuT++6sXTlztJdQYkRamJvKfQDU1naC8mAkGf79Tb
a0xyBGAUII0GfREY6t4/+NAP2Yyb3xNlBqcJoTov0JfNKHZcCZePr79j7LK/h
kZxxip+Na9xDpE+oQRV+DRukCRJdiqg+wIDAQABo1AwTjAMBgNVHRMEBTADAQ
H/MB0GA1UdDgQWBBTDEsXJe6pjAQD4ULlB7+GMDBlimTAfBgNVHSMEGDAWgBT
DEsXJe6pjAQD4ULlB7+GMDBlimTANBgkqhkiG9w0BAQUFAAOCAQEAWWkNcW6S
1tKKqtzJsdfhjJiAAPQmOXJskv0ta/8f6Acgcum1YieNdtT0n96P7CUHOWP8Q
Bb91JzeewR7b6WJLwb1Offs3wNq3kk75pJe89r4XY39EZHhMW+Dv0PhIKu2Cg
D4LeyH1FVTQkF/QObGEmkn+s+HTsuzd1l2VLwcP1SmsqeP6LAlFj62qqaIJzN
eQ9NVkBqtkygnYlBOkaBTHfQTux3jYNpEo8JJB5e/WFdHYyMNrG2xMOtIC7T4
+IOHgT8PgrNhaeDg9ctewj0X8Qi9nI9nXeinicLX8vj6hdEq3ORv7RZMJNYqv
1HQ3wUE2B7fCPFv7EUwzaCds1kgRQ==
</bpki_cert>
</client>
</msg>
```

4.6. Example 6: Client List Query and Reply

A. Client "List" Query

```
<msg xmlns="http://www.hactrn.net/uris/rpki/publication-spec/"
    version="2" type="query">
    <client action="list"/>
    </msg>
```

B. Client "List" Reply


```
<msg xmlns="http://www.hactrn.net/uris/rpki/publication-spec/"
    version="2" type="reply">
    <client action="list" client_handle="3">
    <bpki_cert>
    MIIDGzCCAgOgAwIBAgIJAKi+/+wUhQlxMA0GCSqGSIb3DQEBBQUAMCQxIjAgB
    gNVBAMTGVRlc3QgQ2VydGlmaWNhdGUgQm9iIFJvb3QwHhcNMDcwODAxMTk1Mz
    EwWhcNMDcwODMxMTk1MzEwWjAkMSIwIAYDVQQDExlUZXN0IENlcnRpZmljYXR
    lIEJvYiBSb290MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEArKYU
    tJaM5PH5917SG2ACc7iBYdQO2HYyu8Gb6i9Q2Gxc3cWEX7RTBvgOL79pWf3GI
    dnoupzMnoZVtY3GUx2G/0WkmLui2TCeDhcfXdQ4rcp8J3V/6ESj+yuEPPOG8U
    N17mUKKgujrch6ZvgCDO9AyOK/uXu+ABQXTPsn2pVe2EVh3V004ShLi8GKgVd
```

```
            qb/rW/6GTg0Xb/zLT6WWMuT++6sXTlztJdQYkRamJvKfQDU1naC8mAkGf79Tb
            a0xyBGAUII0GfREY6t4/+NAP2Yyb3xNlBqcJoTov0JfNKHZcCZePr79j7LK/h
            kZxxip+Na9xDpE+oQRV+DRukCRJdiqg+wIDAQABo1AwTjAMBgNVHRMEBTADAQ
            H/MB0GA1UdDgQWBBTDEsXJe6pjAQD4ULlB7+GMDBlimTAfBgNVHSMEGDAWgBT
            DEsXJe6pjAQD4ULlB7+GMDBlimTANBgkqhkiG9w0BAQUFAAOCAQEAWWkNcW6S
            1tKKqtzJsdfhjJiAAPQmOXJskv0ta/8f6Acgcum1YieNdtT0n96P7CUHOWP8Q
            Bb91JzeewR7b6WJLwb1Offs3wNq3kk75pJe89r4XY39EZHhMW+Dv0PhIKu2Cg
            D4LeyH1FVTQkF/QObGEmkn+s+HTsuzd1l2VLwcP1SmsqeP6LAlFj62qqaIJzN
            eQ9NVkBqtkygnYlBOkaBTHfQTux3jYNpEo8JJB5e/WFdHYyMNrG2xMOtIC7T4
            +IOHgT8PgrNhaeDg9ctewj0X8Qi9nI9nXeinicLX8vj6hdEq3ORv7RZMJNYqv
            1HQ3wUE2B7fCPFv7EUwzaCds1kgRQ==
            </bpki_cert>
            </client>
            </msg>
```

4.7. Example 7: Client Destroy Query and Reply

A. Client "Destroy" Query

```
<msg xmlns="http://www.hactrn.net/uris/rpki/publication-spec/"
    version="2" type="query">
    <client action="destroy" client_handle="3"/>
    </msg>
```

B. Client "Destroy" Reply

```
<msg xmlns="http://www.hactrn.net/uris/rpki/publication-spec/"
    version="2" type="reply">
    <client action="destroy" client_handle="3"/>
    </msg>
```

4.8. Example 8: Publish Query and Reply

A. Publish Query


```
<msg xmlns="http://www.hactrn.net/uris/rpki/publication-spec/"
    version="2" type="query">
    <publish uri="rsync://wombat.invalid/testbed/RIR/1/j7ghjwblCr
    cCp9ltyPDNzYKPfxc.cer">
```

```
            MIIE+jCCA+KgAwIBAgIBDTANBgkqhkiG9w0BAQsFADAzMTEwLwYDVQQDEyhER
            jRBODAxN0U2NkE5RTkxNzJFNDYxMkQ4Q0Y0QzgzRjIzOERFMkEzMB4XDTA4MD
            UyMjE4MDUxMloXDTA4MDUyNDE3NTQ1M1owMzExMC8GA1UEAxMoOEZCODIxOEY
            wNkU1MEFCNzAyQTdEOTZEQzhGMENEQ0Q4MjhGN0YxNzCCASIwDQYJKoZIhvcN
            AQEBBQADggEPADCCAQoCggEBAMeziKp0k5nP7v6SZoNsXIMQYRgNtC6Fr/9Xm
            /1yQHomiPqHUk47rHhGojYiK5AhkrwoYhkH4UjJl2iwklDYczXuaBU3F5qrKl
            Z4aZnjIxdlP7+hktVpeApL6yuJTUAYeC3UIxnLDVdD6phydZ/FOQluffiNDjz
            teCCvoyOUatqt8WB+oND6LToHp028g1YUYLHG6mur0dPdcHOVXLSmUDuZ1HDz
            1nDuYvIVKjB/MpH9aW9XeaQ6ZFIlZVPwuuvI2brR+ThH7Gv27GL/o8qFdC300
            VQfoTZ+rKPGDE8K1cI906BL4kiwx9z0oiDcE96QCz+B0vsjc9mGaA1jgAxlXW
            sCAwEAAaOCAhcwggITMB0GA1UdDgQWBBSPuCGPBuUKtwKn2W3I8M3Ngo9/FzA
            fBgNVHSMEGDAWgBTfSoAX5mqekXLkYS2M9Mg/I43iozBVBgNVHR8ETjBMMEqg
            SKBGhkRyc3luYzovL2xvY2FsaG9zdDo0NDAwL3Rlc3RiZWQvUklSLzEvMzBxQ
            UYtWnFucEZ5NUdFdGpQVElQeU9ONHFNLmNybDBFBggrBgEFBQcBAQQ5MDcwNQ
            YIKwYBBQUHMAKGKXJzeW5jOi8vbG9jYWxob3N0OjQ0MDAvdGVzdGJlZC9XT01
            CQVQuY2VyMBgGA1UdIAEB/wQOMAwwCgYIKwYBBQUHDgIwDwYDVR0TAQH/BAUw
            AwEB/zAOBgNVHQ8BAf8EBAMCAQYwgZsGCCsGAQUFBwELBIGOMIGLMDQGCCsGA
            QUFBzAFhihyc3luYzovL2xvY2FsaG9zdDo0NDAwL3Rlc3RiZWQvUklSL1IwLz
            EvMFMGCCsGAQUFBzAKhkdyc3luYzovL2xvY2FsaG9zdDo0NDAwL3Rlc3RiZWQ
            vUklSL1IwLzEvajdnaGp3YmxDcmNDcDlsdHlQRE56WUtQZnhjLm1uZjAaBggr
            BgEFBQcBCAEB/wQLMAmgBzAFAgMA/BUwPgYIKwYBBQUHAQcBAf8ELzAtMCsEA
            gABMCUDAwAKAzAOAwUAwAACAQMFAcAAAiAwDgMFAsAAAiwDBQDAAAJkMA0GCS
            qGSIb3DQEBCwUAA4IBAQCEhuH7jtI2PJY6+zwv306vmCuXhtu9Lr2mmRw2ZEr
            B8EMcb5xypMrNqMoKeu14K2x4a4RPJkK4yAThM81FPNRsU5mM0acIRnAPtxjH
            vPME7PHN2w2nGLASRsZmaa+b8A7SSOxVcFURazENztppsolHeTpm0cpLItK7m
            NpudUg1JGuFo94VLf1MnE2EqARG1vTsNhel/SM/UvOArCCOBvf0Gz7kSuupDS
            Z7qx+LiDmtEsLdbGNQBiYPbLrDk41PHrxdx28qIj7ejZkRzNFw/3pi8/XK281
            h8zeHoFVu6ghRPy5dbOA4akX/KG6b8XIx0iwPYdLiDbdWFbtTdPcXBauY
            </publish>
    </msg>
```

B. Publish Reply

```
        <msg xmlns="http://www.hactrn.net/uris/rpki/publication-spec/"
            version="2" type="reply">
            <publish uri="rsync://wombat.invalid/testbed/RIR/1/j7ghjwblCr
            cCp9ltyPDNzYKPfxc.cer"/>
            </msg>
```

4.9. Example 9: Withdraw Query and Reply

A. Withdraw Query

```
<msg xmlns="http://www.hactrn.net/uris/rpki/publication-spec/"
    version="2" type="query">
<withdraw uri="rsync://wombat.invalid/testbed/RIR/1/j7ghjwblCr
    cCp9ltyPDNzYKPfxc.cer"/>
    </msg>
```

B. Withdraw Reply

```
<msg xmlns="http://www.hactrn.net/uris/rpki/publication-spec/"
    version="2" type="reply">
<withdraw uri="rsync://wombat.invalid/testbed/RIR/1/j7ghjwblCr
    cCp9ltyPDNzYKPfxc.cer"/>
    </msg>
```

4.10. Example 10: Report Error Reply

A. Report Error Reply 1

```
<msg xmlns="http://www.hactrn.net/uris/rpki/publication-spec/"
    version="2" type="reply">
<report_error error_code="your_hair_is_on_fire">text string</
    report_error>
    </msg>
```

B. Report Error Reply 2

```
<msg xmlns="http://www.hactrn.net/uris/rpki/publication-spec/"
    version="2" type="reply">
    <report_error error_code="your_hair_is_on_fire"/>
    </msg>
```

5. Operational Considerations

There are two basic options open to the repository operator as to how the publication tree is laid out. The first option is simple: each publication client is given its own directory one level below the top of the rcynic module, and there is no overlap between the publication spaces used by different clients. For example:

```
rsync://example.org/rpki/Alice/
rsync://example.org/rpki/Bob/
rsync://example.org/rpki/Carol/
```

This has the advantage of being very easy for the publication operator to manage, but has the drawback of making it difficult for relying parties to fetch published objects both safely and as

efficiently as possible.

Given that the mandatory-to-implement retrieval protocol for relying parties is rsync, a more efficient repository structure would be one which minimized the number of rsync fetches required. One such structure would be one in which the publication directories for subjects were placed underneath the publication directories of their issuers: since the normal synchronization tree walk is top-down, this can significantly reduce the total number of rsync connections required to synchronize. For example:

rsync://example.org/rpki/Alice/
rsync://example.org/rpki/Alice/Bob/
rsync://example.org/rpki/Alice/Bob/Carol/

Preliminary measurement suggests that, in the case of large numbers of small publication directories, the time needed to set up and tear down individual rsync connections becomes significant, and that a properly optimized tree structure can reduce synchronization time by an order of magnitude.

The more complex tree structure does require careful attention to the base_uri attribute values when setting up clients. In the example above, assuming that Alice issues to Bob who in turn issues to Carol, Alice has ceded control of a portion of her publication space to Bob, who has in turn ceded a portion of that to Carol, and the base_uri attributes in the <client/> setup messages should reflect this.

The details of how the repository operator determines that Alice has given Bob permission to nest Bob's publication directory under Alice's is outside the scope of this protocol.

6. IANA Considerations

IANA is asked to register the application/rpki-publication MIME media type as follows:

```
MIME media type name:  application
MIME subtype name:  rpki-publication
Required parameters:  None
Optional parameters:  None
Encoding considerations:  binary
Security considerations:  Carries an RPKI Publication Protocol
   Message, as defined in this document.
Interoperability considerations:  None
Published specification:  This document
Applications which use this media type: HTTP
Additional information:
   Magic number(s):  None
   File extension(s):
   Macintosh File Type Code(s):
Person & email address to contact for further information:
   Rob Austein <sra@isc.org>
Intended usage:  COMMON
Author/Change controller: Rob Austein <sra@isc.org>
```

7. Security Considerations

The RPKI publication protocol and the data it publishes use entirely separate PKIs for authentication. The published data is authenticated within the RPKI, and this protocol has nothing to do with that authentication, nor does it require that the published objects be valid in the RPKI. The publication protocol uses a separate Business PKI (BPKI) to authenticate its messages.

Each of the RPKI publication protocol messages is CMS-signed. Because of that protection at the application layer, this protocol does not require the use of HTTPS or other transport security

mechanisms.

Compromise of a publication server, perhaps through mismanagement of BPKI keys, could lead to a denial-of-service attack on the RPKI. An attacker gaining access to BPKI keys could use this protocol delete (withdraw) RPKI objects, leading to routing changes or failures. Accordingly, as in most PKIs, good key management practices are important.

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, September 2009.

[RFC6492] Huston, G., Loomans, R., Ellacott, B., and R. Austein, "A Protocol for Provisioning Resource Certificates", RFC 6492, February 2012.

8.2. Informative References

[RFC6480] Lepinski, M. and S. Kent, "An Infrastructure to Support Secure Internet Routing", RFC 6480, February 2012.

Authors' Addresses

Samuel Weiler
SPARTA, Inc.
7110 Samuel Morse Drive
Columbia, Maryland 21046
US

Email: weiler@tislabs.com

Anuja Sonalker
SPARTA, Inc.
7110 Samuel Morse Drive
Columbia, Maryland 21046
US

Email: Anuja.Sonalker@sparta.com

Rob Austein
Dragon Research Labs

Email: sra@hactrn.net