Internet Engineering Task Force (IETF)
Request for Comments: 8183
Category: Standards Track
ISSN: 2070-1721

R. Austein
Dragon Research Labs

July 2017

An Out-of-Band Setup Protocol for Resource Public Key Infrastructure (RPKI) Production Services

Abstract

This note describes a simple out-of-band protocol to ease setup of the Resource Public Key Infrastructure (RPKI) provisioning and publication protocols between two parties. The protocol is encoded in a small number of XML messages, which can be passed back and forth by any mutually agreeable means which provides acceptable data integrity and authentication.

This setup protocol is not part of the provisioning or publication protocol; rather, it is intended to simplify configuration of these protocols by setting up relationships and exchanging keying material used to authenticate those relationships.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc8183.

Table of Contents

1. Introduction

This note describes a small XML-based out-of-band protocol used to set up relationships between parents and children in the RPKI provisioning protocol [RFC6492] and between publishers and repositories in the RPKI publication protocol [RFC8181].

The basic function of this protocol is public key exchange, in the form of self-signed X.509 certificates, but workshop experience has demonstrated that it's simpler for the user if we also bundle the other configuration information needed to bring up a new player into the messages used in the key exchange.

The underlying transport for this protocol is deliberately unspecified. It might be a USB stick, a web interface secured with conventional HTTPS, PGP-signed email, a T-shirt printed with a Quick Response (QR) code, or a carrier pigeon.

Since much of the purpose of this protocol is key exchange, authentication and integrity of the key exchange MUST be ensured via external means. Typically, such means will tie directly to a new or existing business relationship.

2. Terminology

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.

All of the protocols configured by this setup protocol have their own terminology for their actors, but in the context of this protocol that terminology becomes somewhat confusing. All of the players in this setup protocol issue certificates, are the subjects of other certificates, operate servers, and, in most cases, act as clients for one protocol or another. Therefore, this note uses its own terms for the actors in this protocol.

Child: An entity acting in the client ("subject") role of the provisioning protocol defined in [RFC6492].

Parent: An entity acting in the server ("issuer") role of the provisioning protocol defined in [RFC6492].

Publisher: An entity acting in the client role of the publication protocol defined in [RFC8181].

Repository: An entity acting in the server role of the publication protocol defined in [RFC8181].

Note that a given entity might act in more than one of these roles; for example, in one of the simplest cases, the child is the same entity as the publisher, while the parent is the same entity as the repository.

3. History

The protocol described in this document grew out of a series of workshops held starting in 2010, at which it became clear that manual configuration of keying material and service URLs was both error prone and unnecessarily confusing. The basic mechanism and semantics have been essentially unchanged since the earliest versions of the protocol, but there were several workshop-driven syntax changes and simplifications before the protocol made its way into the IETF, and a few more simplifications and minor extensions have occurred since that time.

4. Overview of the BPKI

Several protocols related to RPKI provisioning use signed Cryptographic Message Syntax (CMS) messages [RFC5652] to authenticate the underlying XML-based protocols. Verification of these CMS messages requires X.509 certificates. The PKI that holds these certificates is distinct from the RPKI and contains no RFC 3779 resources. We refer to this as the "Business PKI" (BPKI), to distinguish it from the RPKI. The "B" is a hint that the certificate relationships in the BPKI are likely to follow and become part of existing contractual relationships between the issuers and subjects of this PKI.

The RPKI provisioning protocol does not dictate a particular structure for the BPKI, beyond the basic requirement that it be possible for one party to sign and the other party to verify the CMS messages. This allows a certain amount of flexibility to allow an Internet registry to reuse an existing PKI as the BPKI if that makes sense in their context.

In order to keep this protocol simple, we adopt a somewhat constrained model of the BPKI. The first two operations in this protocol are an exchange of public keys between child and parent for use in the provisioning protocol; the latter two operations in this protocol are an exchange of public keys between publisher and repository for use in the publication protocol. In each of these operations, the sending party includes its public key, in the form of a self-signed X.509 Certification Authority (CA) certificate. The

private keys corresponding to the exchanged certificates are not used to sign CMS messages directly; instead, the exchanged CA certificates are the issuers of the BPKI end-entity (EE) certificates which will be included in the CMS messages and can be used, along with the exchanged certificates, to verify the CMS messages.

Details of how to tie the exchanged certificates into an implementation's local BPKI are left to the implementation, but the recommended approach is to cross-certify the received public key and subject name under one's own BPKI, using a Basic Constraints extension with cA = TRUE, pathLenConstraint = 0, indicating that the cross-certified certificate is a CA certificate which is allowed to issue EE certificates but is not allowed to issue CA certificates. See Section 4.2.1.9 of [RFC5280] for more information about the Basic Constraints extension.

For example, suppose that Alice and Bob each have their own self-signed BPKI certificates:

```
Issuer:       CN = Alice CA
Subject:      CN = Alice CA
Public Key:   [Alice CA Public Key]
BasicConstraints: cA = TRUE

Issuer:       CN = Bob CA
Subject:      CN = Bob CA
Public Key:   [Bob CA Public Key]
BasicConstraints: cA = TRUE
```

Alice sends Bob her self-signed BPKI certificate, and Bob cross certifies its public key and subject name under Bob's own self-signed BPKI certificate:

```
Issuer:       CN = Bob CA
Subject:      CN = Alice CA
Public Key:   [Alice CA Public Key]
BasicConstraints: cA = TRUE, pathLenConstraint = 0
```

Later, when Bob receives a CMS message from Alice, Bob can verify this message via a trust chain back to Bob's own trust anchor:

```
Issuer:       CN = Alice CA
Subject:      CN = Alice EE
Public Key:   [Alice EE Public Key]
```

A complete description of the certificates allowed here is beyond the scope of this document, as it is determined primarily by what is acceptable to the several other protocols for which this protocol is

handling setup. Furthermore, we expect the requirements to change over time to track changes in cryptographic algorithms, required key length, and so forth. Finally, since this protocol is restricted to setting up pairwise relationships, all that's really required is that the two parties involved in a particular conversation agree on what constitutes an acceptable certificate.

All of that said, in practice, the certificates currently exchanged by this protocol at the time this document was written are what a reader familiar with the technology would probably expect: RSA keys with lengths in the 2048-4096 bit range, SHA-2 digests, and a few common X.509v3 extensions (principally Basic Constraints, Authority Key Identifier, and Subject Key Identifier). Since the most likely usage is a cross-certification operation in which the recipient simply extracts the subject name and public key after checking the self-signature and discards the rest of the incoming certificate, the practical value of esoteric X.509v3 extensions is somewhat limited.

5. Protocol Elements

Each message in the protocol is a distinct XML element in the <http://www.hactrn.net/uris/rpki/rpki-setup/> XML namespace.

The outermost XML element of each message contains a version attribute. This document describes version 1 of the protocol.

Appendix A is a [RELAX-NG] schema for this protocol. The schema is normative: in the event of a disagreement between the schema and the following textual description, the schema is authoritative.

Since "1" is currently the only value allowed for the version attribute in the schema, an incorrect protocol version can be detected either by checking the version attribute directly or as a schema validation error.

5.1. Common Protocol Elements

Most messages contain, among other things, a self-signed BPKI X.509 certificate. These certificates are represented as XML elements whose text value is the Base64 text ([RFC4648], Section 4, with line breaks within the Base64 text permitted but not required) encoding the DER representation of the X.509 certificate.

A number of attributes contain "handles". A handle in this protocol is a text string in the US-ASCII character set consisting of letters, digits, and the special characters "/", "-", and "_". This protocol places no special semantics on the structure of these handles, although implementations might. Handles are protocol elements, not

necessarily meaningful to humans, thus the simplicity of a restricted character set makes more sense than the complex rules which would be needed for internationalized text.

Most messages allow an optional "tag" attribute. This is an opaque cookie supplied by the client in a particular exchange and echoed by the server; the intent is to simplify the process of matching a response received by the client with an outstanding request.

5.2. Protocol Messages

The core of this protocol consists of four message types, representing the basic request and response semantics needed to configure an RPKI engine to talk to its parent and its repository via the provisioning and publication protocols, respectively.

5.2.1. <child_request/>

The <child_request/> message is an initial setup request from a provisioning protocol child to its provisioning protocol parent.

Fields in the <child_request/> message:

version: The version attribute specifies the protocol version. This note describes protocol version 1.

tag: The child MAY include a "tag" attribute in the request message.

child_handle: The child_handle attribute is what the child calls itself. This is just a hint from the child to the parent, and the parent need not honor it.

child_bpki_ta: The <child_bpki_ta/> element is the child's BPKI identity, a self-signed X.509 BPKI certificate, encoded in Base64.

This CA certificate will be the issuer of the BPKI EE certificates corresponding to private keys that the child will use when sending provisioning protocol messages to the parent.

```
------------------------------------------------------------------------
<child_request
    xmlns="http://www.hactrn.net/uris/rpki/rpki-setup/"
    version="1"
    child_handle="Bob">
  <child_bpki_ta>
    R29kIGlzIHJlYWwgdW5sZXNzIGRlY2xhcmVkIGludGVnZXI=
  </child_bpki_ta>
</child_request>
------------------------------------------------------------------------
```

5.2.2. <parent_response/>

The <parent_response/> message is a response from a provisioning protocol parent to a provisioning protocol child that had previously sent a <child_request/> message.

Fields in the <parent_response/> message:

version: The version attribute specifies the protocol version. This note describes protocol version 1.

tag: If the <child_request/> message included a "tag" attribute, the parent MUST include an identical "tag" attribute in the <parent_response/> message; if the request did not include a tag attribute, the response MUST NOT include a tag attribute either.

service_uri: The service_uri attribute contains an HTTP or HTTPS URL [RFC7230] that the child should contact for up-down [RFC6492] service.

child_handle: The child_handle attribute is the parent's name for the child. This MAY match the child_handle from the <child_request/> message. If they do not match, the parent wins, because the parent gets to dictate the names in the provisioning protocol. This value is the sender field in provisioning protocol request messages and the recipient field in provisioning protocol response messages.

parent_handle: The parent_handle attribute is the parent's name for itself. This value is the recipient field in provisioning protocol request messages and the sender field in provisioning protocol response messages.

parent_bpki_ta: The <parent_bpki_ta/> element is the parent's BPKI identity, a self-signed X.509 BPKI certificate.

This certificate is the issuer of the BPKI EE certificates corresponding to private keys that the parent will use to sign provisioning protocol messages to the child.

offer: If an <offer/> element is present, the parent is offering publication service to the child. The <offer/> element, if present, is empty.

referral: If <referral/> elements are present, they suggest third-party publication services which the child might use, and contain:

referrer: A referrer attribute, containing the handle by which the publication repository knows the parent,

contact_uri: An optional contact_uri attribute that the child may be able to follow for more information, and

Authorization token: The text of the <referral/> element is the Base64 encoding of a signed authorization token granting the child the right to use a portion of the parent's namespace at the publication repository in question. See Section 5.3 for details on the authorization token.

A parent is unlikely to need to send both <offer> and <referral> elements, but strictly speaking they are not mutually exclusive, so a parent which really needs to express that it both offers repository service to its child and is also willing to refer its child to one or more other repository servers can do so.

```
<parent_response
    xmlns="http://www.hactrn.net/uris/rpki/rpki-setup/"
    version="1"
    service_uri="http://a.example/up-down/Alice/Bob-42"
    child_handle="Bob-42"
    parent_handle="Alice">
  <parent_bpki_ta>
    WW91IGNhbiBoYWNrIGFueXRoaW5nIHlvdSB3YW50IHdpdGggVEVDTyBhbmQgRERU
  </parent_bpki_ta>
  <offer/>
</parent_response>
```

```
------------------------------------------------------------------------
<parent_response
    xmlns="http://www.hactrn.net/uris/rpki/rpki-setup/"
    version="1"
    service_uri="http://bob.example/up-down/Bob/Carol"
    child_handle="Carol"
    parent_handle="Bob">
  <parent_bpki_ta>
    R29kIGlzIHJlYWwgdW5sZXNzIGRlY2xhcmVkIGludGVnZXI=
  </parent_bpki_ta>
  <referral
      referrer="Alice/Bob-42">
    R28sIGxlbW1pbmdzLCBnbyE=
  </referral>
</parent_response>
------------------------------------------------------------------------
```

5.2.3. <publisher_request/>

The <publisher_request/> message is a setup request from a publisher to a repository. Generally, this will not take place until after the publisher has set up the provisioning protocol via a <child_request/> / <parent_response/> exchange: in particular, the <referral> sub-element here requires an <authorization/> token provided by the provisioning protocol exchange.

Fields in the <publisher_request/> message:

version: The version attribute specifies the protocol version. This note describes protocol version 1.

tag: The publisher MAY include a "tag" attribute in the request message.

publisher_handle: The publisher_handle attribute is the publisher's name for itself. This is just a hint; the repository need not honor it.

publisher_bpki_ta: The <publisher_bpki_ta/> element is the publisher's BPKI identity, a self-signed X.509 BPKI certificate. This certificate is the issuer of the BPKI EE certificates corresponding to private keys that the publisher will use to sign publication protocol messages to the repository.

referral: If a <referral/> element is present, it contains:

referrer: A referrer attribute containing the publication handle of the referring parent, and

Authorization token: The text of the <referral/> element is the Base64 encoding of a signed authorization token granting the publisher the right to use a portion of its parent's namespace at this repository. See Section 5.3 for details on the authorization token.

These fields are copies of values that a parent provided to the child in the <parent_response/> message (see Section 5.2.2). The referrer attribute is present to aid lookup of the corresponding certificate by the repository. Note that the repository operator makes the final decision on whether to grant publication service to the prospective publisher. The <referral/> element just conveys a parent's grant of permission to use a portion of that parent's namespace.

```
<publisher_request
    xmlns="http://www.hactrn.net/uris/rpki/rpki-setup/"
    version="1"
    tag="A0001"
    publisher_handle="Bob">
  <publisher_bpki_ta>
    R29kIGlzIHJlYWwgdW5sZXNzIGRlY2xhcmVkIGludGVnZXI=
  </publisher_bpki_ta>
</publisher_request>
```

5.2.4. <repository_response/>

The <repository_response/> message is a repository's response to a publisher which has previously sent a <publisher_request/> message.

Fields in the <repository_response/> message:

version: The version attribute specifies the protocol version. This note describes protocol version 1.

tag: If the <publisher_request/> message included a "tag" attribute, the repository MUST include an identical "tag" attribute in the <repository_response/> message; if the request did not include a tag attribute, the response MUST NOT include a tag attribute either.

service_uri: The service_uri attribute contains an HTTP or HTTPS URL [RFC7230] that the publisher should contact for publication service [RFC8181].

publisher_handle: The publisher_handle attribute is the repository's name for the publisher. This may or may not match the publisher_handle attribute in the publisher's <publisher_request/> message.

sia_base: The sia_base attribute is the rsync:// URI for the base of the publication space allocated to the publisher.

rrdp_notification_uri: The optional rrdp_notification_uri attribute is the URI for the RPKI Repository Delta Protocol (RRDP) notification file covering the publication space allocated to the publisher [RFC8182].

repository_bpki_ta: The <repository_bpki_ta/> element is the repository's BPKI identity, a self-signed X.509 BPKI certificate.

```
<repository_response
    xmlns="http://www.hactrn.net/uris/rpki/rpki-setup/"
    version="1"
    tag="A0001"
    service_uri="http://a.example/publication/Alice/Bob-42"
    publisher_handle="Alice/Bob-42"
    sia_base="rsync://a.example/rpki/Alice/Bob-42/"
    rrdp_notification_uri="https://rpki.example/rrdp/notify.xml">
  <repository_bpki_ta>
    WW91IGNhbiBoYWNrIGFueXRoaW5nIHlvdSB3YW50IHdpdGggVEVDTyBhbmQgRERU
  </repository_bpki_ta>
</repository_response>
```

5.3. <authorization/>

The <authorization/> element is a separate message which is signed with CMS and then included as the Base64 content of <referral/> elements in other messages.

The eContentType for the signed CMS message is id-ct-xml [RFC6492].

Fields in the <authorization/> element:

version: The version attribute specifies the protocol version. This note describes protocol version 1.

authorized_sia_base: The value of the authorized_sia_base attribute is the rsync:// URI of the base of the namespace which the referrer is delegating.

BPKI TA: The text of the <authorization/> element is the identity of the entity to whom the referrer is delegating the portion of the namespace named in the authorized_sia_base attribute, represented as a Base64-encoded self-signed X.509 BPKI certificate.

```
------------------------------------------------------------------------
<authorization
    xmlns="http://www.hactrn.net/uris/rpki/rpki-setup/"
    version="1"
    authorized_sia_base="rsync://a.example/rpki/Alice/Bob-42/Carol/">
  SSd2ZSBoYWQgZnVuIGJlZm9yZS4gIFRoaXMgaXNuJ3QgaXQu
</authorization>
------------------------------------------------------------------------
```

5.4. <error/>

The <error/> element is an optional message which can be used in response to any of the core protocol messages described in Section 5.2.

Whether an <error/> element is an appropriate way to signal errors back to the sender of a protocol message depends on details of the implementation, which are outside this specification. For example, if this protocol is embedded in a web portal interface which is designed to let a human being upload and download these messages via upload and download forms, a human-readable error message may be more appropriate. On the other hand, a portal intended to be driven by a robotic client might well want to use an <error/> message to signal errors. Similar arguments apply to non-web encapsulations (such as email or a USB stick); the primary factor is likely to be whether the implementation expects the error to be handled by a human being or by a program.

Fields in the <error/> message:

version: The version attribute specifies the protocol version. This note describes protocol version 1.

reason: The reason attribute contains a code indicating what was wrong with the message. This version of the protocol defines the following codes:

syntax-error: Receiver could not parse the offending message.

authentication-failure: Receiver could not authenticate the offending message.

refused: Receiver refused to perform the requested action.

Offending message: The <error/> element contains a verbatim copy of the message to which this error applies.

```
<error
    xmlns="http://www.hactrn.net/uris/rpki/rpki-setup/"
    version="1"
    reason="refused">
  <child_request
      xmlns="http://www.hactrn.net/uris/rpki/rpki-setup/"
      version="1"
      child_handle="Carol">
    <child_bpki_ta>
      SSd2ZSBoYWQgZnVuIGJlZm9yZS4gIFRoaXMgaXNuJ3QgaXQu
    </child_bpki_ta>
  </child_request>
</error>
```

6. Protocol Walk-Through

This section walks through a few simple examples of the protocol in use and stars our old friends, Alice, Bob, and Carol. In this example, Alice is the root of an RPKI tree, Bob wants to get address and Autonomous System Number (ASN) resources from Alice, and Carol wants to get some of those resources in turn from Bob. Alice offers publication service, which is used by all three.

Alice, Bob, and Carol each generate his or her own self-signed BPKI certificate.

Bob constructs a <child_request/> message and sends it to Alice:

```
<child_request
    xmlns="http://www.hactrn.net/uris/rpki/rpki-setup/"
    version="1"
    child_handle="Bob">
  <child_bpki_ta>
    R29kIGlzIHJlYWwgdW5sZXNzIGRlY2xhcmVkIGludGVnZXI=
  </child_bpki_ta>
</child_request>
```

- Bob's preferred handle is "Bob", so Bob uses that when setting child_handle.

- <child_bpki_ta/> is Bob's self-signed BPKI certificate.

Alice replies with a <parent_response/> message, but Alice already has 41 other children named Bob, so she calls this one "Bob-42". Alice's provisioning protocol server happens to use a RESTful URL scheme so that it can find the expected validation context for the provisioning protocol CMS message just by looking at the URL, so the service URL she provides to Bob includes both her name and Bob's. Alice offers publication service, so she offers to let Bob use it; Alice doesn't have to do this, she could just omit this and leave Bob to find publication service on his own, but Alice is trying to be helpful to her customer Bob. Bob doesn't have to accept Alice's offer, but may choose to do so.

```
<parent_response
    xmlns="http://www.hactrn.net/uris/rpki/rpki-setup/"
    version="1"
    service_uri="http://a.example/up-down/Alice/Bob-42"
    child_handle="Bob-42"
    parent_handle="Alice">
  <parent_bpki_ta>
    WW91IGNhbiBoYWNrIGFueXRoaW5nIHlvdSB3YW50IHdpdGggVEVDTyBhbmQgRERU
  </parent_bpki_ta>
  <offer/>
</parent_response>
```

- <parent_bpki_ta/> is Alice's own self-signed BPKI certificate.

Bob receives Alice's <parent_response/> and extracts the fields Bob's RPKI engine will need to know about (child_handle, parent_handle, service_uri, and <parent_bpki_ta/>). Bob also sees the repository offer, decides to take Alice up on this offer, and constructs a <publisher_request/> message accordingly:

```
------------------------------------------------------------------------
<publisher_request
    xmlns="http://www.hactrn.net/uris/rpki/rpki-setup/"
    version="1"
    tag="A0001"
    publisher_handle="Bob">
  <publisher_bpki_ta>
    R29kIGlzIHJlYWwgdW5sZXNzIGRlY2xhcmVkIGludGVnZXI=
  </publisher_bpki_ta>
</publisher_request>
------------------------------------------------------------------------
```

Alice receives Bob's request to use Alice's publication service, decides to honor the offer she made, and sends back a <repository_response/> message in response. Alice recognizes Bob as one of her own children, because she's already seen Bob's self-signed BPKI certificate, so she allocates publication space to Bob under her own publication space, so that relying parties who rsync her products will pick up Bob's products automatically without needing an additional fetch operation.

```
------------------------------------------------------------------------
<repository_response
    xmlns="http://www.hactrn.net/uris/rpki/rpki-setup/"
    version="1"
    tag="A0001"
    service_uri="http://a.example/publication/Alice/Bob-42"
    publisher_handle="Alice/Bob-42"
    sia_base="rsync://a.example/rpki/Alice/Bob-42/"
    rrdp_notification_uri="https://rpki.example/rrdp/notify.xml">
  <repository_bpki_ta>
    WW91IGNhbiBoYWNrIGFueXRoaW5nIHlvdSB3YW50IHdpdGggVEVDTyBhbmQgRERU
  </repository_bpki_ta>
</repository_response>
------------------------------------------------------------------------
```

Bob should now have everything he needs to talk to Alice for both provisioning and publication.

A more interesting case is Bob's child, Carol. Carol wants to get her resources from Bob and, like Bob, does not particularly want to operate a publication service. Bob doesn't have a publication service of his own to offer, but he can refer Carol to Alice, along with his permission for Carol to use a portion of the namespace that Alice gave him.

Carol's <child_request/> to Bob looks very similar to Bob's earlier request to Alice:

```
<child_request
    xmlns="http://www.hactrn.net/uris/rpki/rpki-setup/"
    version="1"
    child_handle="Carol">
  <child_bpki_ta>
    SSd2ZSBoYWQgZnVuIGJlZm9yZS4gIFRoaXMgaXNuJ3QgaXQu
  </child_bpki_ta>
</child_request>
```

Bob's <parent_response/> to Carol also looks a lot like Alice's response to Bob, except that Bob includes a <referral/> element instead of an <offer/> element. Carol is an only child, so Bob leaves her name alone:

```
<parent_response
    xmlns="http://www.hactrn.net/uris/rpki/rpki-setup/"
    version="1"
    service_uri="http://bob.example/up-down/Bob/Carol"
    child_handle="Carol"
    parent_handle="Bob">
  <parent_bpki_ta>
    R29kIGlzIHJlYWwgdW5sZXNzIGRlY2xhcmVkIGludGVnZXI=
  </parent_bpki_ta>
  <referral
      referrer="Alice/Bob-42">
    R28sIGxlbW1pbmdzLCBnbyE=
  </referral>
</parent_response>
```

Bob's response includes a <referral/> element with a referrer attribute of "Alice/Bob-42", since that's Bob's name in Alice's repository. The Base64-encoded authorization token is an <authorization/> element in a CMS message that can be verified against Bob's self-signed BPKI certificate, using a BPKI EE certificate included in the CMS wrapper. The <authorization/> text is Carol's self-signed BPKI certificate; Bob's signature over this element indicates Bob's permission for Carol to use the indicated portion of Bob's publication space.

```
<authorization
    xmlns="http://www.hactrn.net/uris/rpki/rpki-setup/"
    version="1"
    authorized_sia_base="rsync://a.example/rpki/Alice/Bob-42/Carol/">
  SSd2ZSBoYWQgZnVuIGJlZm9yZS4gIFRoaXMgaXNuJ3QgaXQu
</authorization>
```

Carol, not wanting to have to run a publication service, presents Bob's referral to Alice in the hope that Alice will let Carol use Alice's publication service. So Carol constructs a <publisher_request/> message, including the referral information received from Bob, and sends it all to Alice:

```
<publisher_request
    xmlns="http://www.hactrn.net/uris/rpki/rpki-setup/"
    version="1"
    tag="A0002"
    publisher_handle="Carol">
  <publisher_bpki_ta>
    SSd2ZSBoYWQgZnVuIGJlZm9yZS4gIFRoaXMgaXNuJ3QgaXQu
  </publisher_bpki_ta>
  <referral
      referrer="Alice/Bob-42">
    R28sIGxlbW1pbmdzLCBnbyE=
  </referral>
</publisher_request>
```

Alice sees the signed authorization token Bob gave to Carol, checks its signature, and unpacks it. When the signature proves valid and the contained BPKI trust anchor (TA) matches Carol's, Alice knows that Bob is willing to let Carol use a portion of Bob's namespace. Given this, Alice is willing to provide publication service to Carol in the subtree allocated by Bob for this purpose, so Alice sends back a <repository_response/>:

```
-------------------------------------------------------------------
<repository_response
    xmlns="http://www.hactrn.net/uris/rpki/rpki-setup/"
    version="1"
    tag="A0002"
    service_uri="http://a.example/publication/Alice/Bob-42/Carol"
    publisher_handle="Alice/Bob-42/Carol"
    sia_base="rsync://a.example/rpki/Alice/Bob-42/Carol/">
  <repository_bpki_ta>
    WW91IGNhbiBoYWNrIGFueXRoaW5nIHlvdSB3YW50IHdpdGggVEVDTyBhbmQgRERU
  </repository_bpki_ta>
</repository_response>
-------------------------------------------------------------------
```

Once Carol receives this response, Carol should be good to go.

In theory, the publication referral mechanism can extend indefinitely (for example, Carol can refer her child Dave to Alice for publication service, and it should all work). In practice, this has not yet been implemented, much less tested. In order to keep the protocol relatively simple, we've deliberately ignored perverse cases such as Bob being willing to refer Carol to Alice but not wanting Carol to be allowed to refer Dave to Alice.

Any RPKI operator is free to run their own publication service should they feel a need to do so, and a child need not accept any particular <offer/> or <referral/>. In general, having a smaller number of larger publication repositories is probably good for overall system performance, because it will tend to reduce the number of distinct repositories from which each relying party will need to fetch, but the decision on where to publish is up to individual RPKI CA operators and out of scope for this protocol.

7. IANA Considerations

This document does not require any IANA actions.

8. Security Considerations

As stated in Section 1, the basic function of this protocol is an exchange of public keys to be used as BPKI trust anchors. Integrity and authentication of these exchanges MUST be ensured via external mechanisms deliberately left unspecified in this protocol.

9. References

9.1. Normative References

[RELAX-NG] Clark, J., "RELAX NG Compact Syntax", OASIS Committee Specification, November 2002, <https://www.oasis-open.org/committees/relax-ng/compact-20021121.html>.

[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>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <http://www.rfc-editor.org/info/rfc4648>.

[RFC6492] Huston, G., Loomans, R., Ellacott, B., and R. Austein, "A Protocol for Provisioning Resource Certificates", RFC 6492, DOI 10.17487/RFC6492, February 2012, <http://www.rfc-editor.org/info/rfc6492>.

[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>.

[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)", RFC 8181, DOI 10.17487/RFC8181, July 2017, <http://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, <http://www.rfc-editor.org/info/rfc8182>.

9.2. Informative References

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <http://www.rfc-editor.org/info/rfc5280>.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <http://www.rfc-editor.org/info/rfc5652>.

Appendix A. RELAX NG Schema

Here is a [RELAX-NG] schema describing the protocol elements.

This schema is normative: in the event of a disagreement between this schema and the document text above, this schema is authoritative.

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

version = "1"

base64  = xsd:base64Binary { maxLength="512000" }
handle  = xsd:string { maxLength="255" pattern="[\-_A-Za-z0-9/]*" }
uri     = xsd:anyURI { maxLength="4096" }
any     = element * { attribute * { text }*, ( any | text )* }
tag     = xsd:token { maxLength="1024" }

authorization_token = base64
bpki_ta = base64

start |= element child_request {
  attribute version { version },
  attribute child_handle { handle },
  attribute tag { tag }?,
  element child_bpki_ta { bpki_ta }
}

start |= element parent_response {
  attribute version { version },
  attribute service_uri { uri },
  attribute child_handle { handle },
  attribute parent_handle { handle },
  attribute tag { tag }?,
  element parent_bpki_ta { bpki_ta },
  element offer { empty }?,
  element referral {
    attribute referrer { handle },
    attribute contact_uri { uri }?,
    authorization_token
  }*
}

start |= element publisher_request {
  attribute version { version },
  attribute publisher_handle { handle },
  attribute tag { tag }?,
  element publisher_bpki_ta { bpki_ta },
  element referral {
```

```
      attribute referrer { handle },
      authorization_token
    }*
  }

  start |= element repository_response {
    attribute version { version },
    attribute service_uri { uri },
    attribute publisher_handle { handle },
    attribute sia_base { uri },
    attribute rrdp_notification_uri { uri }?,
    attribute tag { tag }?,
    element repository_bpki_ta { bpki_ta }
  }

  start |= element authorization {
    attribute version { version },
    attribute authorized_sia_base { uri },
    bpki_ta
  }

  start |= element error {
    attribute version { version },
    attribute reason {
      "syntax-error" |
      "authentication-failure" |
      "refused"
    },
    any?
  }
```

Acknowledgements

The author would like to thank: Byron Ellacott, George Michaelson, Leif Johansson, Matsuzaki Yoshinobu, Michael Elkins, Randy Bush, Seiichi Kawamura, Tim Bruijnzeels, and anybody else who helped along the way but whose name the author has temporarily forgotten.

Author’s Address

Rob Austein
Dragon Research Labs

Email: sra@hactrn.net