Secure Inter-Domain Routing
Internet-Draft
Intended status: Standards Track
Expires: September 5, 2014

S. Kent
D. Mandelberg
BBN Technologies

March 4, 2014

Suspenders: A Fail-safe Mechanism for the RPKI
draft-kent-sidr-suspenders-01

Abstract

The Resource Public Key Infrastructure (RPKI) is an authorization infrastructure that allows the holder of Internet Number Resources (INRs) to make verifiable statements about those resources. The certification authorities (CAs) in the RPKI issue certificates to match their allocation of INRs. These entities are trusted to issue certificates that accurately reflect the allocation state of resources as per their databases. However, there is some risk that a CA will make inappropriate changes to the RPKI, either accidentally or deliberately (e.g., as a result of some form of "government mandate"). The mechanisms described below, and referred to as "Suspenders" are intended to address this risk.

Suspenders enables an INR holder to publish information about changes to objects it signs and publishes in the RPKI repository system. This information is made available via a file that is external to the RPKI repository, so that Relying Parties (RPs) can detect erroneous or malicious changes related to these objects. RPs can then decide, individually, whether to accept changes that are not corroborated by independent assertions by INR holders, or to revert to previously verified RPKI data.

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 5, 2014.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Overview 2
2. Terminology 4
3. The LOCK Record and INRD File 4
3.1. LOCK Record Format and Semantics 4
3.2. INRD File Format and Semantics 6
4. Self-checking by RPs 10
5. Detection & Remediation 11
6. INRD Management Scenarios 13
7. IANA Considerations 14
8. Security Considerations 15
9. Acknowledgements 15
10. References 15
10.1. Informative References 15
10.2. Normative References 15
Appendix A. RPKI Object Whacking and Competition 16
Appendix B. Design Criteria (do we still need this section?) . . 17
Authors' Addresses 18

1. Overview

The Resource Public Key Infrastructure (RPKI) is an authorization infrastructure that allows the holder of Internet number resources (INRs) to make verifiable statements about those resources. For example, the holder of a block of IP(v4 or v6) addresses can issue a Route Origination Authorization (ROA) to authorize an autonomous system to originate routes for that block.

The certification authorities (CAs) in the RPKI issue certificates to match their allocation of INRs. These entities are trusted to issue certificates that accurately reflect the allocation state of resources as per their databases. However, there is some risk that a

CA will make inappropriate changes to the RPKI, either accidentally or deliberately (e.g., as a result of some form of "government mandate").

Assertions by INR holders about their resources, and about bindings among resources, are realized by publishing RPKI signed objects via the RPKI repository system [RFC6481]. For example, authorization to originate a route for a prefix is accomplished by issuing a ROA. Changes in the RPKI can have an adverse impact on routing in the Internet, by changing the set of (valid) signed objects for a resource. Invalidating a ROA could cause the origin authorized by the ROA in question to be less preferred; adding a ROA for a more specific prefix could enable an unauthorized party to represent itself as the legitimate origin for traffic for that prefix.

The goal of Suspenders is to minimize the likelihood that changes to the RPKI will adversely affect INR holders, irrespective of whether the changes are inadvertent or malicious. Suspenders should work when an INR holder acts as its own CA (and manages its own publication point), and when the INR holder has outsourced these management functions. Suspenders allows each INR holder to assert a "lock" on selected objects at its publication point, to protect the bindings asserted by these objects. Changes to protected objects are confirmed by the INR holder, via a file published outside the repository system. Changes to the validity of protected objects, effected by changes to any other objects in the RPKI, are presumed to be unauthorized (and thus suspicious), unless independently confirmed by the INR holder.

Detection of potentially adverse changes is carried out by each INR holder for its own resources, and by each RP that elects to implement Suspenders. It is critical that an INR holder be able to quickly detect adverse changes that affect its own resources, so that it can initiate actions to remedy the problem. RPs should be able to detect potentially adverse changes, that are not authorized by INR holders, so that they can (at their discretion) use cached, validated data in lieu of such changes. The model adopted here is to assume that changes to previously-validated data should not be accepted, unless authorized by the relevant INR holder. Thus RPs who detect changes need to be able to verify that these changes are authorized by the INR holder. Because not all INR holders manage their own CAs and publication points, an external mechanism is used to signal authorized changes to RPs.

2. 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].

3. The LOCK Record and INRD File

An INR holder that elects to protect its resources and resource bindings creates a LOCK record in its publication point. The INR holder also generates and signs an Internet Number Resource Declaration (INRD) file, and publishes it at a location independent of the RPKI repository system. The LOCK record consists of a URL that points to the INRD file, and a public key used to verify a signature on the content of that file. (This public key is distinct from any used by the INR holder in the RPKI context.) The INRD file contains the date at which the most recent changes were made, and enumerates those changes. The formats of the LOCK record and INRD file are described below.

3.1. LOCK Record Format and Semantics

The LOCK record conforms to the signed object specification from [RFC6488], which, in turn, uses the CMS [RFC5652] signed-data object format. See [RFC6488] for the top-level signed-data format and the constraints imposed on that format for use in the RPKI context. The LOCK encapsulated content is defined below:

```
EncapsulatedContentInfo ::= SEQUENCE {
    eContentType ContentType,
    eContent [0] EXPLICIT OCTET STRING OPTIONAL }

ContentType ::= OBJECT IDENTIFIER
```

The eContentType for an LOCK record is defined as id-ct-rpkiLOCK and it has the numeric value 1.2.840.113549.1.9.16.1.XX.

```
id-smime OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840)
                                 rsadsi(113549) pkcs(1) pkcs9(9) 16 }

id-ct OBJECT IDENTIFIER ::= { id-smime 1 }

id-ct-rpkiLOCK OBJECT IDENTIFIER ::= { id-ct XX }
```

The eContent for an LOCK record is defined by the following ASN.1

```
LOCK ::= SEQUENCE {
    version     [0] INTEGER DEFAULT 0,
    outsourced  BOOLEAN,
    uRL         IA5String,
    publicKey   SubjectPublicKeyInfo }

SubjectPublicKeyInfo ::= SEQUENCE {
    algorithm         AlgorithmIdentifier,
    subjectPublicKey  BIT STRING  }
```

The EE certificate embedded in the LOCK record MUST use the inherit flag in the [RFC3779] extensions. (The content of the LOCK is independent of the 3779 extensions in the EE certificate, so it is appropriate to use the inherit flag here.)

The version number of the LOCK record determines the set of RPKI object types that it protects. Version 0 protects the LOCK record itself, ROAs, (subordinate) CA certificates, and router certificates (if present).

The algorithm and subjectPublicKey fields in the publicKey MUST conform to the guidance in Section 3 of [RFC6485].

If an RP elects to process a LOCK record, it verifies the signature on the record using the procedure described in [RFC6488]. If the signature verification fails, it ignores the record. (If the RP has a previously validated LOCK record, it continues to use that record instance.)

If the signature verification succeeds, the RP extracts the version number and verifies that the RP is prepared to process this version of the record. If not, it ignores the record. If it is prepared to process this version, it extracts the URL and public key fields. The URL is used to fetch the corresponding INRD file, and the public key is used to verify the signature on that file.

If the RP has a copy of an INRD file for this publication point, and if the RP detects no material changes to the protected records at the publication point, the RP SHOULD NOT fetch the INRD file. (A material change is one that affects the semantics of the object. For example, for a ROA, only changes to the prefixes and/or ASN are material.) If the RP does not hold a copy of the INRD file, or if a protected record has changed, the RP fetches a new INRD file using the URL, and proceeds as described in Section 3.2.

When an INR holder has outsourced management of its RPKI CA function and publication point, it is susceptible to attacks in which the LOCK record itself is changed. This is because the entity providing these

functions could create a new LOCK record containing a new URL and public key, thus defeating the LOCK/INRD mechanism. An authorized change to the content of a LOCK record should be very rare. A location selected as a home for an INRD file should be stable, and thus the URL should rarely change. The public key used to verify the signature on an INRD file should also be constant for long intervals. The LOCK record contains a flag that indicates whether the INR holder has outsourced CA and publication point management. If this flag is FALSE, an RP will accept changes to the LOCK record (see Section 5) just as it would changes to any other object at a protected publication point. If the flag is TRUE, then any change to a LOCK record is regarded as suspicious by RPs. In such cases the RP delays accepting the new LOCK record and associated INRD file, as discussed in Section 5.

3.2. INRD File Format and Semantics

The INRD file is a DER-encoded ASN.1 file that contains data associated with a single INR holder (publication point owner). The file is encoded using ASN.1, since most of the values it holds will be compared to data from RPKI objects that also are ASN.1 encoded, and because it is a signed object.

```
INRD ::= SEQUENCE {
    tbsINRD    TBSINRD,
    algorithm  AlgorithmIdentifier,
    signature  OCTET STRING
    }

TBSINRD ::= SEQUENCE {
    version       [0] INTEGER DEFAULT 0,
    lastChange        UTCTime,
    changeWindow      ENUMERATED
        {
        1week (7) DEFAULT
        2week (14)
        4week (28)
        },
    additions     [1] SEQUENCE SIZE (1..MAX) OF
                          ProtectedObject OPTIONAL,
    deletions     [2] SEQUENCE SIZE (1..MAX) OF
                          ProtectedObject OPTIONAL,
    keyRollover   [3] OCTET STRING OPTIONAL,
    algRollover   [4] OCTET STRING OPTIONAL
    }

ProtectedObject ::= CHOICE {
    cmsObject    [0] EncapsulatedContentInfo,
    rtrCert      [1] RtrCertInfo,
    cACert       [2] CACertInfo
    }

RtrCertInfo ::= SEQUENCE {
    subjKeyId  OCTET STRING,
    aSNum      INTEGER
    }

CACertInfo ::= SEQUENCE {
    subjKeyId          OCTET STRING,
    ipAddrBlocks   [0] IPAddrBlocks OPTIONAL,
    aSIdentifiers  [1] ASIdentifiers OPTIONAL
    }

-- See [RFC3779] for the definitions of IPAddrBlocks and
-- ASIdentifiers.
```

The lastChange and changeWindow values are used to bound the set of additions and deletions stored in an INRD file. The default is a one week window, but two and four week values also may be expressed. The window determines the oldest time at which changes to protected

objects at the publication point are represented in the additions and deletions portions of the file.

The additions element is used by an INR holder to list all protected objects that have been added to the publication point over the interval defined by the change window. If no objects have been added during this interval, the element is omitted. Similarly, the deletions element is used by an INR holder to list all protected objects that have been removed from the publication point over the interval defined by the change window. If no objects have been removed during this interval, the element is omitted.

A LOCK or ROA is listed in the additions and/or deletions fields by putting its EncapsulatedContentInfo in the cmsObject field of a ProtectedObject. A router certificate is listed by putting its SKI and AS number in the rtrCert field. A CA certificate is listed by putting its SKI and [RFC3779] resources in the cACert field. If the outsourced flag in the LOCK record is FALSE, then no CA certificates should be included in the additions or deletions elements. If any CA certificates are included in these elements, they are ignored. RPs SHOULD accept all valid CA certificates issued at this publication point when the outsourced flag is FALSE.

The key rollover element is present only during the time when a key rollover [RFC6489] is taking place. It signals to RPs that an additional set of objects exist that would ordinarily be viewed as competing with the objects protected by this INRD file. The SKI contained here is that of the CA for the "other" key. During key rollover each CA will have its own LOCK record, that points to its own INRD file. The old CA will list the new CA's SKI here; the new CA will not include this field. Key rollover is a transient condition, so the need for both LOCK records and INRD files ought not be very long.

The algorithm rollover element is present during the time when an algorithm rollover [RFC6916] is taking place. It signals to RPs that an additional set of objects exist that would ordinarily be viewed as competing with the objects protected by this INRD file. The SKI contained here is that of the CA for the "other" algorithm. During algorithm rollover each CA will have its own LOCK record, that points to its own INRD file, and each of them will list the other CA's SKI here. (Note that the SKI value is compared against the SKI in the CA certificate in question. An RP does not compute an SKI. This means that changes to the hash algorithm used to compute an SKI do not affect how an RP processes this field. An RP MUST be prepared to deal with an SKI length other than the 20 octet value in common use today.) Algorithm transition is a long process, so both sets of LOCK records and INRD files will persist for an extended period.

An RP fetches an INRD file using a URL from a LOCK record, as noted above. If the file cannot be located, the RP software logs an error and regards any changes to the publication point as suspicious. If the file is located, the RP verifies the signature on the file using the public key (and indicated algorithms) from the same LOCK record. If the signature fails, the RP software logs an error and regards any changes to the publication point as suspicious. If the signature is valid, the RP extracts the data elements from the INRD file.

If this is the first time that an INRD file is fetched for this publication point, the file is accepted, and its content is used to populate the RP's expanded local cache. If the INRD file is replacing a previously acquired instance for this publication point, the content is used to confirm changes to protected objects at this publication point. If an RP detects changes to protected publication point objects that occurred after the lastChange time, these changes are treated as suspicious.

Authorizing changes to subordinate CA certificates in an INRD file is critical when an INR holder outsources CA and publication point management. Listing these CAs and their associated 3779 extension data enables an RP to detect creation of unauthorized CAs that could then create competing ROAs or router certificates. However, if an INR holder operates its own CA and manages its publication point, it is not necessary to protect against such attacks. To signal this situation, the "outsourced" flag in the LOCK record is set to FALSE. Under this condition, an RP will not impose change control checks on subordinate CA certificates for the publication point.

Some classes of INR holders need not publish a LOCK record and INRD file. IANA, RIRs, and NIRs, are principally delegators of resources. Each of these RPKI entities SHOULD create one publication point for the resources used by the entity for its own networking needs, and a separate publication point under which all resource delegations take place. The first publication point MAY be protected by a LOCK record, so that ROAs and router certificates associated with those resources can be protected. However, the second publication point OUGHT not include a LOCK record. If this convention is followed, these classes of INR holders need not update an INRD file every time a new subordinate CA is created or modified, as a result of delegation. If an INR holder follows this convention, and includes a LOCK record in its superior publication point, that record, and the associated INRD file, conveys some degree of protection for the subordinate CA resources, even if the INR holders of these resources do not publish LOCK records.

4. Self-checking by RPs

It is easy for an INR holder, acting as an RP, to determine if any of its resource bindings have been undermined via the RPKI. It knows what resources it holds, and what assertions it has made relevant to those resources. Any conflicting RPKI objects represent a problem! It is more difficult for an RP to detect problems with another INR holder's resources, because it lacks the knowledge that the other INR holder has. The mechanisms described in Section 5 are designed to enable RPs to detect these problems. This section describes the procedures each RP executes to detect adverse changes to its own data in the RPKI repository system. Note that the procedures in this section do not require use of the LOCK record or INRD file.

When and INR downloads RPKI data, as it normally does, it SHOULD perform the checks noted below, to detect problems. To enable such checking, each INR holder's RP software MUST be configured with data about the ROAs, and other protected objects, of this INR holder. If any of these objects are missing or fail to validate, then the INR holder has detected a problem, and is notified.

The semantics of ROAs require an additional check; if other ROAs for the same or more specific prefixes are found anywhere in the RPKI repository system, this too indicates a problem, and the INR holder is notified.

The semantics of router certificates, require a separate, additional check. A router certificate binds a public key (and a router ID) to an ASN. Thus, if an INR holder discovers router certificates for its ASN, that it did not authorize, this indicates a problem.

As additional objects are protected via this mechanism, it will be necessary to perform additional checks to detect the latter sort of adverse changes, based on the semantics of the protected objects.

In any case, RP software SHOULD inform the INR holder of the apparent cause and source of the problem, e.g., a revoked or expired certificate or a manifest problem, and guide the INR holder to the responsible CAs (e.g., using Ghostbusters [RFC6493] records).

When an INR holder is alerted to a change adversely affecting its own resources, it is expected to contact the appropriate RPKI entities to rectify the error in a timely fashion. If the changes are determined to be intentional (and not authorized by the INR holder), the INR holder can inform the Internet operations community (via an out of band mechanism), which can then decide, individually, how to respond.

Remedying a problem detected by an INR holder is not likely to be instantaneous, even if the problem is just an error. To avoid adversely affecting routing, the mechanisms described in Section 5 enable RPs to detect a change that adversely affects INR holders, and to reject it, reverting to previously validated INR data. This gives the INR holder time to resolve the problem. Reverting to an earlier version of INR data is conceptually easy for RPs, because they already cache RPKI data. The mechanisms described below require augmenting the RPKI local cache maintained by each RP, to detect adverse changes, making use of information gleaned from LOCK records and INRD files. The next section describes how the LOCK and INRD data is used.

5. Detection & Remediation

The design described in this section assumes that an RP has acquired a snapshot of the RPKI repository, validated and extracted INR holding and binding data, and considers this data to be "good". The detection and remediation algorithm is initialized by acquiring a complete download of RPKI repository data, and by fetching INRD files for all publication points that contain a LOCK record. (Prior to this initialization step, it is not possible for an RP to detect and respond to adverse changes to the RPKI, using the technique described below.)

Each RP already maintains a cache of RPKI data [RFC6480], [RFC6481]; this document extends that cache. For every publication point that contains a LOCK record, the content of that record, and the corresponding INRD file content, become part of the data maintained by each RP.

An RP acquires and validates all changed RPKI objects as usual. An RP does not update its cache with the changes, until additional checks, described below, are performed. Before accepting any changes the RP MUST process every pub point where there is (or was) a LOCK record. For each of these pub points, if there are changes to protected objects, these changes must be confirmed by the corresponding INRD file before they are accepted. If any of these checks fail, the changes are held in escrow, waiting for a timeout (or an updated INRD file?). After all protected pub point changes have been processed, then changes for unprotected pub point can be accepted. The checks will detect pending changes that would whack or compete with protected objects, and place them in escrow.

After validating all changed objects downloaded from the RPKI repository, an RP performs the following additional checks for every publication point that has (or had) a LOCK record:

- ROA and router certificate whacking
- INR sub-delegation changes
- ROA competition
- router certificate competition
- LOCK record changes

A ROA or a router certificate has been whacked (see Appendix A) if it was valid and is now missing or invalid, and if it is not indicated as deleted in the INRD file of its issuer. Any previously valid ROA that is no longer valid (or missing) is checked against the INRD file for the ROA issuer, to determine if the ROA or (router certificate) certificate has been legitimately revoked/removed. If the INRD file confirms the action, the old ROA (or router certificate) is removed from the local cache. If not, the old ROA (or router certificate) is retained, but marked as suspicious.

Changes to INR sub-delegation occur when the INR holder issues a new CA certificate, an existing child CA certificate expires, or any other change affects the status of a child CA certificate. These changes are accepted by an RP only if they are confirmed in the INR holder's INRD file.

A newly issued ROA is in competition (see Appendix A) with an existing ROA if the new ROA specifies the same or a more specific prefix than the older ROA, the new ROA is not issued by one of the existing ROA's issuer's descendants, and the new ROA was not authorized by the INRD file of the existing ROA. A competing ROA is not accepted as valid by an RP.

A newly-issued router certificate competes with an existing router certificate, if the new certificate includes the same ASN and was not authorized by the INRD file covering the existing router certificate. A competing router certificate is not accepted as valid by an RP. (Such a certificate would be accepted if the INRD file of the issuer of the original certificate indicates that the old certificate has been deleted, and not replaced with a new router certificate associated with the same entity. In this case, the newly-issued certificate would not be in competition.)

As noted above, any change to a LOCK record is viewed as suspicious unless the outsourced flag is FALSE. If the record is for a publication point that is not outsourced, then a changed LOCK record is accepted as valid if the corresponding INRD file authorizes the new record. (If the INR holder has changed the public key for the

INRD file, it is RECOMMENDED that the URL also change. This allows the INR holder to publish a new INRD file that authorizes the new LOCK record, minimizing the potential race condition between updating an INRD file and a LOCK record.)

If the LOCK record shows that the publication point is outsourced, an RP examines the changes made to the LOCK record. If the URL has changed, but the public key and the outsource flag are unchanged, the new LOCK record may be accepted, if the new INRD file authorizes the change. If not, the new LOCK record is rejected. If the public key has been changed, a delay is imposed on accepting the new LOCK record, even if the INRD file authorizes the change. (should we establish a global delay, or should each INR holder publish its own delay preference in the INRD file?)

Remediation for all of the whacking and competition events consists of NOT making a change in the local cache when an unconfirmed change is encountered.

6. INRD Management Scenarios

Common wisdom notes that we cannot choose our parents, but we can choose our friends, and we should do so wisely. In the RPKI context, and INR holder cannot, generally choose its CA, but Suspenders allows the INR holder to choose its INRD file server. It should do so wisely.

An INRD file is published outside of the RPKI repository system, and is verified using a public key that is also independent of the RPKI. The motivation for these two measures is to insulate this part of the Suspenders system from possible manipulation by an entity to whom CA and publication point services have been outsourced. If an INR holder acts as its own CA, and manages its own publication point, it can publish its INRD file on the same machines as its publication point, but not in the publication point. In this case the independence features are not critical, but they also don't cause harm for this class of INR holder.

Every INR holder needs to choose a location for the INRD file that is highly available. When an INR holder has outsourced CA and publication point management, independent publication of the INRD file is critical. The INR holder needs to choose a location for the INRD file that is highly available. It also is appropriate to consider placing the file outside of the geopolitical region in which the INR holder (and its RIR) operate. Here too the motivation is to insulate the INR holder from a malicious action by the CA service provider, or, perhaps, an RIR above it.

Organizations may arise to offer hosting for INRD files, as a service for INR holders. They could offer just file storage, or they might offer more extensive services. For example, an organization might monitor an INR holder's publication point and create the INRD file data, and even sign it for the INR holder. (In this case the organization would provide the public key to the INR holder for inclusion in the LOCK record.) Various other arrangements between the INR holder and a organization that assists in managing INRD files are possible, and are a local matter between the INR holder and the organization.

A country might elect to mandate use of Suspenders, as a means to protect the INRs of its ISPs and other organizations that run BGP with in the country. The motivation is similar to that cited above, i.e., protecting INRs against errors or malicious actions by RPKI entities. In this case the country itself generally is not an INR holder per se, so the relationship is somewhat different from that discussed above. Nonetheless, the mechanisms described above apply.

For example, Elbonia might mandate that every INR holder within the country make use of Suspenders. Every Elbonian INR holder will be required to include a LOCK record in its publication point, no matter where that publication point is realized. The URL in each LOCK points to a file on a server managed by an Elbonian government organization. Each Elbonian ISP would be required to follow the procedures described in Section 5, when managing its local cache.

7. IANA Considerations

This document registers the following in the "RPKI Signed Object" registry created by [RFC6488]:

```
Name: LOCK
OID: 1.2.840.113549.1.9.16.1.XX
Reference: [RFCxxxx] (this document)
```

This document also registers the following three-letter filename extension in the "RPKI Repository Name Schemes" registry created by [RFC6481]:

```
Filename extension: lck
RPKI Object: LOCK
Reference: [RFCxxxx] (this document)
```

8. Security Considerations

This document specifies Suspenders, a set of security-focused mechanisms designed to protect INR holders against accidental and malicious changes to RPKI repository data, and to enable RPs to detect and respond to such changes. More text to be provided later.

9. Acknowledgements

Richard Barnes provided the motivation to develop Suspenders, after he identified a problem with the LTAM [I-D.ietf-sidr-ltamgmt] design.

10. References

10.1. Informative References

[I-D.ietf-sidr-ltamgmt] Reynolds, M., Kent, S., and M. Lepinski, "Local Trust Anchor Management for the Resource Public Key Infrastructure", draft-ietf-sidr-ltamgmt-08 (work in progress), April 2013.

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

[RFC6481] Huston, G., Loomans, R., and G. Michaelson, "A Profile for Resource Certificate Repository Structure", RFC 6481, February 2012.

[RFC6489] Huston, G., Michaelson, G., and S. Kent, "Certification Authority (CA) Key Rollover in the Resource Public Key Infrastructure (RPKI)", BCP 174, RFC 6489, February 2012.

[RFC6493] Bush, R., "The Resource Public Key Infrastructure (RPKI) Ghostbusters Record", RFC 6493, February 2012.

10.2. Normative References

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

[RFC3779] Lynn, C., Kent, S., and K. Seo, "X.509 Extensions for IP Addresses and AS Identifiers", RFC 3779, June 2004.

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

[RFC6485] Huston, G., "The Profile for Algorithms and Key Sizes for Use in the Resource Public Key Infrastructure (RPKI)", RFC 6485, February 2012.

[RFC6488] Lepinski, M., Chi, A., and S. Kent, "Signed Object Template for the Resource Public Key Infrastructure (RPKI)", RFC 6488, February 2012.

[RFC6916] Gagliano, R., Kent, S., and S. Turner, "Algorithm Agility Procedure for the Resource Public Key Infrastructure (RPKI)", BCP 182, RFC 6916, April 2013.

Appendix A. RPKI Object Whacking and Competition

There are two ways that an RPKI object can be adversely affected. We term these actions "whacking" and "competition."

Any object in the RPKI can become invalid or inaccessible (to RPs) via various actions by CAs and/or publication point maintainers along the certificate path from the object's EE certificate to a trust anchor (TA). Any action that causes an object to become invalid or inaccessible is termed "whacking". Revocation of the EE certificate for an object whacks it. Revocation of any CA certificate along the certificate path for the object (without reissuance) has the same effect. Reissuance of any CA certificate along the certificate path, with changes in [RFC3779] extensions in any of these certificates to exclude the resources cited in a targeted object, also constitutes whacking. Changing a manifest along the certificate path might whack an object (depending on how RPs deal with manifest changes), and removing an object from the RPKI repository system also potentially whacks it. Unless an action that causes an object to be whacked is authorized by the creator of an object, whacking is an attack against the INR holder that created the whacked object.

A different form of attack is termed object "competition". The details of object competition are determined by the semantics of the object. In the general case, one object competes with another object (of the same type), if the newer object creates a binding that adversely affects the binding expressed in the original object. So, for example, a newly issued ROA competes with an existing ROA if the new ROA contains the same or more specific prefixes than the older ROA. Competition does not always indicate an attack; the transfer of resources in a "make before break" model implies ROA competition. A newly issued router certificate competes with a previously issued one if the new certificate binds the same ASN to a public key issued by a different entity. (If key rollover or algorithm transition is in progress, such competition is explicitly authorized via the INRD file.)

Competition that is not authorized by the issuer of the original router certificate is viewed as an attack against that certificate.

Appendix B. Design Criteria (do we still need this section?)

Several criteria were employed in developing the mechanisms described in this document.

1. It is anticipated that object whacking and competition, and analogous forms of errors that adversely impact INR holders, will be infrequent. Thus the detection mechanisms employed by RPs to detect such anomalies ought to be efficient (in terms of data fetching, processing, and storage) for the normal case.

2. RPs may elect to ignore/reject adverse changes to objects if they perceive such changes as suspicious. If an RP elects to reject a change to an object it must have access to previously validated objects for the INR holder question.

3. Transfers of "live" address space will occur, although not frequently. INR holders engaged in such transfers must be able to signal to RPs that such transfers are authorized, so that the transfers are not rejected as suspicious.

4. Routes for a prefix may be legitimately originated by more than one AS (MOA). The design MUST enable an INR holder to inform RPs when this situation is authorized.

5. Many INR holders may choose to outsource CA and publication point management functions. INR holders who choose to outsource these functions should be offered equivalent protection against ROA invalidation and competition as INR holders who perform these functions for themselves.

6. Any new RPKI repository objects used with the mechanisms defined here MUST conform to the format specified in [RFC6488].

7. The decision to process any additional data associated with the mechanisms described in this document is local to each RP. RPs that choose to not implement these mechanisms will incur minimal additional data fetching, storage, and processing burdens.

8. The decision to employ the mechanisms described here to protect INR holdings and binding is a local one made by each INR holder. (INR holders who outsource CA and publication point management functions will require the providers of these services to support creation and publication of one new RPKI object. As a result, all such providers must support generation and maintenance of the

new RPKI object so that their clients have the option to utilize these capabilities.)

9. Revocation and expiration of RPKI object MUST continue to work as they do currently, for all objects that have not been adversely affected.

Authors' Addresses

Stephen Kent
BBN Technologies
10 Moulton St.
Camridge, MA 02138
US

Email: kent@bbn.com

David Mandelberg
BBN Technologies
10 Moulton St.
Camridge, MA 02138
US

Email: david@mandelberg.org