

Secure Inter-Domain Routing
Internet-Draft
Intended status: Informational
Expires: March 11, 2016

Y. Fu
Z. Yan
X. Liu
C. Wang
CNNIC
September 8, 2015

Scenarios of unexpected resource assignment in RPKI
draft-fu-sidr-unexpected-scenarios-00

Abstract

There are some unexpected scenarios where a CA allocates resources to the sub-node caused by misoperation or malicious operation of CA in RPKI. Then some mechanisms may be needed to avoid these scenarios to happen. This document describes these scenarios and related experiments are presented.

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 March 11, 2016.

Copyright Notice

Copyright (c) 2015 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. Introduction	2
2. Terminology	3
3. The scenarios of unexpected resource assignment	3
3.1. Unauthorized resource assignment	3
3.2. Resource reassignment	3
3.3. Resource transfer	4
4. Experimental Result	4
4.1. Unauthorized resource assignment	4
4.1.1. Completely unauthorized assignment	4
4.1.2. Partially unauthorized assignment	6
4.2. Resource reassignment	8
4.2.1. Matching case	8
4.2.2. Intersection case	10
4.2.3. Subset case	12
4.3. Resource transfer	14
5. Solutions	14
5.1. The CA function enhancement	15
5.2. The RP function enhancement	15
6. Security Considerations	15
7. IANA Considerations	15
8. Acknowledgements	15
9. References	15
9.1. Normative References	15
9.2. Informative References	16
Authors' Addresses	16

1. Introduction

n RPKI, CAs (Certification Authorities) are organized in a hierarchical structure which is aligned to the existing INR (Internet Number Resources, including IP prefixes and AS numbers) allocation hierarchy. Each INR allocation requires corresponding resource certificates to attest to it. Two important resource certificates [RFC6480] are needed during this allocation process, and they are CA certificates and EE (End-entity) certificates: CA certificates attest to the INR holdings; EE certificates are primarily used for the validation of ROAs (Route Origin Authorizations) [RFC6482] which is used to verify whether an AS is permitted to originate routes to specific IP prefixes. Besides, Manifest [RFC6486] is used to ensure the integrity of the repository. So the operation of CA is very important for the RPKI deployment and applications based on it.

Considering misoperation and malicious operation are inevitable and the significant impact they may cause, this draft describes and analyzes some scenarios of the unexpected resource assignment caused by CA in RPKI deployment. Then some experiments are given to show these scenarios more clearly. Some suggestions are also proposed to avoid corresponding side-effects.

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 scenarios of unexpected resource assignment

As described in [RFC6480], the holder of resource (IP addresses and AS numbers) may reallocate portions of that block, to the sub-node, subject to contractual constraints established by the registries. But some scenarios of unexpected resource allocation may be caused by the misoperation of malicious operation of the CA as below: for simplicity, we describe some scenarios with the AS number allocation case.

3.1. Unauthorized resource assignment

For this scenario, a CA allocates a block of IP address which does not belong to him to subordinate node. That is, this CA doesn't own this block of IP Prefixes actually. So the sub-node cannot use these addresses. But in RPKI scenario, this assignment could be operated successfully. This scenario may be caused by misoperation of the CA or on purpose.

According to the resources to be allocated, we divide this case into two kinds of scenarios: Completely unauthorized assignment and Partially unauthorized assignment.

(1) Completely unauthorized assignment: the resources to be allocated to subordinate node are without the ownership of CA.

(2) Partially unauthorized assignment: the resources to be allocated to subordinate node are with the partially ownership of CA.

3.2. Resource reassignment

In this scenario, a CA reassign the resource to one sub-node which has been already assigned to another sub-node. This scenario could be sorted into three types:

(1) Matching: the block of IP address which is reassigned is the same as which has been already assigned to the other sub-node.

(2) Subset: The block of IP address which is reassigned is smaller than which has been already assigned to the other sub-node.

(3) Intersection: The block of IP address which is reassigned has overlap with which has been already assigned to the other sub-node.

3.3. Resource transfer

In practice, resource transfer between two Internet registries is reasonably achievable for most useful operational needs. For the resource transfer, a block of IP addresses will be transferred from one sub-node to the other. This scenario is described in [I-D.ymbk-sidr-transfer] in more detail. The resource reassignment may happened in this scenario by the misoperation of the CA.

4. Experimental Result

We test the scenarios described above in our testbed. In this section, we present the test results for each case and analyze these results.

4.1. Unauthorized resource assignment

4.1.1. Completely unauthorized assignment

The test scenario is described in Figure 1: APNIC allocates AS number 65551 and IP prefixes of 192.0.3.128/26 to TWNIC. But APNIC doesn't own this resource completely. We simulated this process of assignment on our testbed and the result is illustrated in Figure 2.

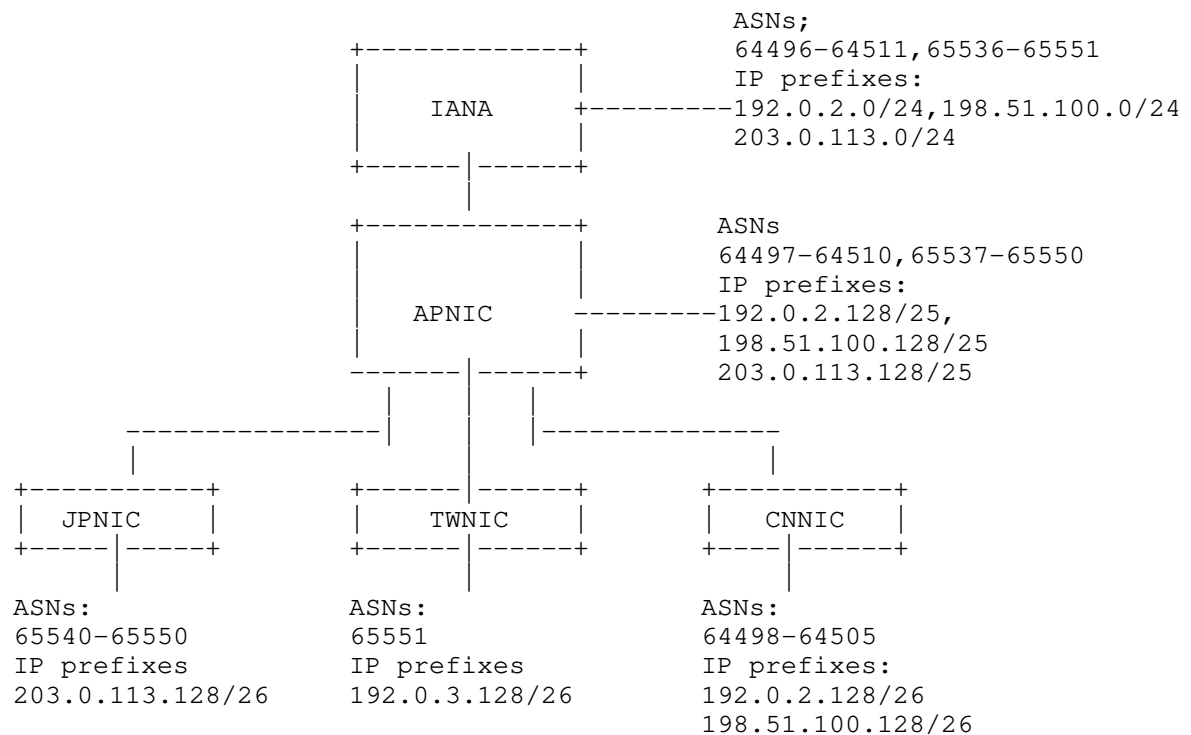


Figure 1: The network architecture of the completely unauthorized assignment

We test this case with the tools offered by <http://rpki.net/>. The result (as described in Figure 2) shows: APNIC has allocated the unauthorized resources (ASNs: 65551, IP prefixes: 192.0.3.128/26) to TWNIC successfully, but TWNIC did not receive these resources actually.

```
root@ubuntu:~#rpki -i cnnic show_received_resources
Parent:      apnic
  notBefore:  2015-07-15T15:53:25Z
  notAfter:   2016-07-14T15:36:05Z
  URI:        rsync://localhost/rpki/iana/apnic/BqHiZw817JRhXby5cljw-Iy75c4.cer
  SIA URI:    rsync://localhost/rpki/iana/apnic/cnnic/
  AIA URI:    rsync://localhost/rpki/iana/RAseYE67qlpBd34u5UqhMjwq8c0.cer
  ASN:        64498-64505
  IPv4:       192.0.2.128/26,198.51.100.128/26
  IPv6:
root@ubuntu:~# rpki -i jpnric show_received_resources
Parent:      apnic
  notBefore:  2015-07-15T15:25:54Z
  notAfter:   2016-07-14T15:20:04Z
  URI:        rsync://localhost/rpki/iana/apnic/NSt9KXs-a2py_OGZlol4fipm1lQ.cer
  SIA URI:    rsync://localhost/rpki/iana/apnic/jpnric/
  AIA URI:    rsync://localhost/rpki/iana/RAseYE67qlpBd34u5UqhMjwq8c0.cer
  ASN:        65540-65550
  IPv4:       203.0.113.128/26
  IPv6:
root@ubuntu:~# rpki -i twnic show_received_resources
root@ubuntu:~#
```

Figure 2: The test result of completely unauthorized assignment

4.1.2. Partially unauthorized assignment

The test scenario is described in Figure 3: APNIC allocates ASNs (9700-9818) to JPNIC. But APNIC only take the ownership of ASNs (9801-9818). We simulated this process of assignment on our testbed and the result is illustrated in Figure 4.

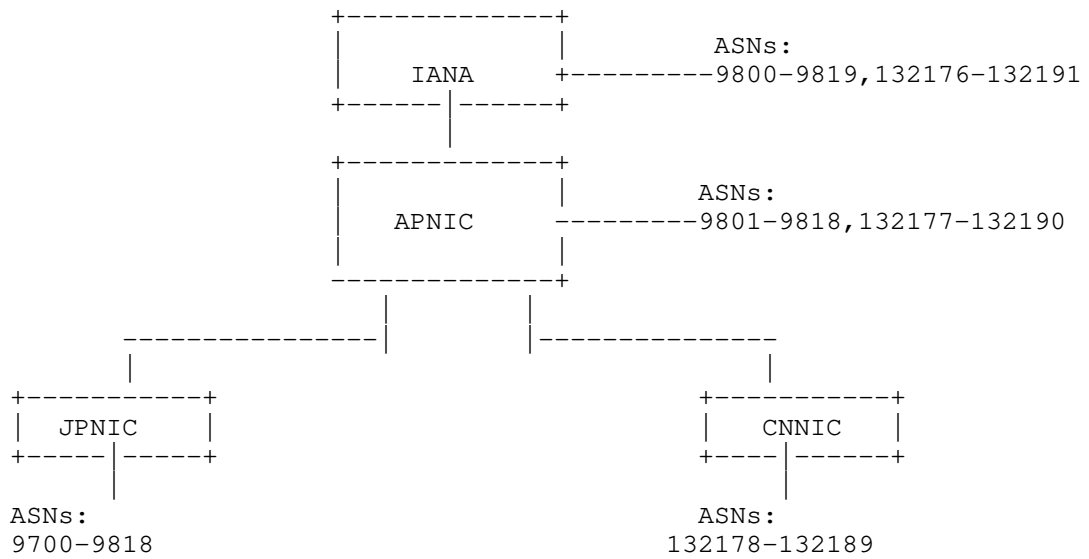


Figure 3: The network architecture of partially unauthorized assignment

Our test result for this scenario (as described in Figure 4) shows: APNIC think that it has allocated the partially unauthorized resources to JPNIC successfully, but JPNIC only received the legal part (ASNs 9801-9818), without the illegal part (ASNs 9700-9800).

```

root@ubuntu:~# rpki -i apnic show_child_resources
Child: cnnic
ASN:      132178-132179
IPv4:     203.0.113.0/26,218.214.108.0/26
Child: jpn
ASN:      9700-9818
IPv4:     218.241.104.0/26

root@ubuntu:~#rpki -i cnnic show_received_resources
Parent:    apnic
notBefore: 2015-08-31T15:45:37Z
notAfter:  2016-08-30T12:24:04Z
URI:       rsync://192.168.137.139/rpki/iana/apnic/t4jbcdikZIF9RGeLpZ0fiBd59U
w.cer
SIA URI:   rsync://192.168.137.139/rpki/iana/apnic/cnnic/
AIA URI:   rsync://192.168.137.139/rpki/iana/HFOgArj0R8dF5vZ-7beqS0CzT2w.cer
ASN:       132178-132189
IPv4:      203.0.113.0/26,218.214.108.0/26
IPv6:

root@ubuntu:~# rpki -i jpn show_received_resources
Parent:    apnic
notBefore: 2015-08-31T15:47:38Z
notAfter:  2016-08-30T12:25:25Z
URI:       rsync://192.168.137.139/rpki/iana/apnic/V4CxZVxBkwwIiBruCy7k7DESUv
g.cer
SIA URI:   rsync://192.168.137.139/rpki/iana/apnic/jpn/
AIA URI:   rsync://192.168.137.139/rpki/iana/HFOgArj0R8dF5vZ-7beqS0CzT2w.cer
ASN:       9801-9818
IPv4:      218.241.104.0/26
IPv6:

```

Figure 4: The test result of partially unauthorized assignment

4.2. Resource reassignment

In the "Resource reassignment" case, we mean that a CA allocate the resources, which have been allocated to one subordinate CA (e.g. CNNIC), to another subordinate CA (e.g. JPNIC). This case may also be caused by misoperation of CA or on purpose. And it may result in resource conflicts in the Internet.

4.2.1. Matching case

For the matching case, the test scenario is described in Figure 5: the resources to be allocated to JPNIC (ASNs 132178-132189) is identical to those allocated to CNNIC (ASNs 132178-132189).

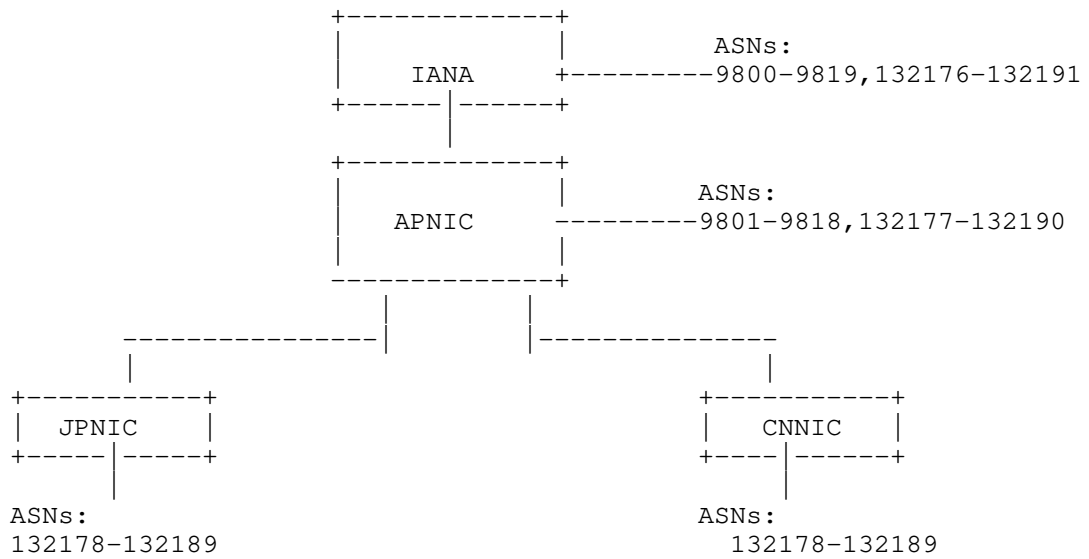


Figure 5: The network architecture of matching case

We test this case with the tools offered by <http://rpki.net/>. The result (as described in Fig 6) is that both CNNIC and JPNIC can receive these ASNs at the same time. It may result in resource confliction in the internet.

```

root@ubuntu:~# rpki -i apnic show_child_resources
Child: cnnic
  ASN: 132178-132189
  IPv4: 203.0.113.0/26
Child: jpnric
  ASN: 132178-132189
  IPv4: 203.0.113.0/26

root@ubuntu:~# rpki -i jpnric show_received_resources
Parent:      apnic
  notBefore: 2015-08-31T13:43:18Z
  notAfter:  2016-08-30T12:25:25Z
  URI:       rsync://192.168.137.139/rpki/iana/apnic/V4CxZVxBkwwIiBruCy7k7DESUv
g.cer
  SIA URI:   rsync://192.168.137.139/rpki/iana/apnic/cnnic/
  AIA URI:   rsync://192.168.137.139/rpki/iana/HFOgArj0R8dF5vZ-7beqS0CzT2w.cer
  ASN:       132178-132189
  IPv4:      203.0.113.0/26
  IPv6:

root@ubuntu:~# rpki -i cnnic show_received_resources
Parent:      apnic
  notBefore: 2015-08-31T13:39:19Z
  notAfter:  2016-08-30T12:24:04Z
  URI:       rsync://192.168.137.139/rpki/iana/apnic/t4jbcdikZIF9RGeLpZ0fiBd59U
w.cer
  SIA URI:   rsync://192.168.137.139/rpki/iana/apnic/cnnic/
  AIA URI:   rsync://192.168.137.139/rpki/iana/HFOgArj0R8dF5vZ-7beqS0CzT2w.cer
  ASN:       132178-132189
  IPv4:      203.0.113.0/26
  IPv6:

```

Figure 6: The test result of matching case

CNNIC and JPNIC receive the same resource assigned by the APNIC at the same time. So the same resource could be allocated to the different sub-node simultaneously.

4.2.2. Intersection case

As is shown in Figure 7, in the intersection case, there is an overlap between the resources to be allocated to JPNIC (ASNs 132180-132190) and those allocated to CNNIC (ASNs 132177-132185).

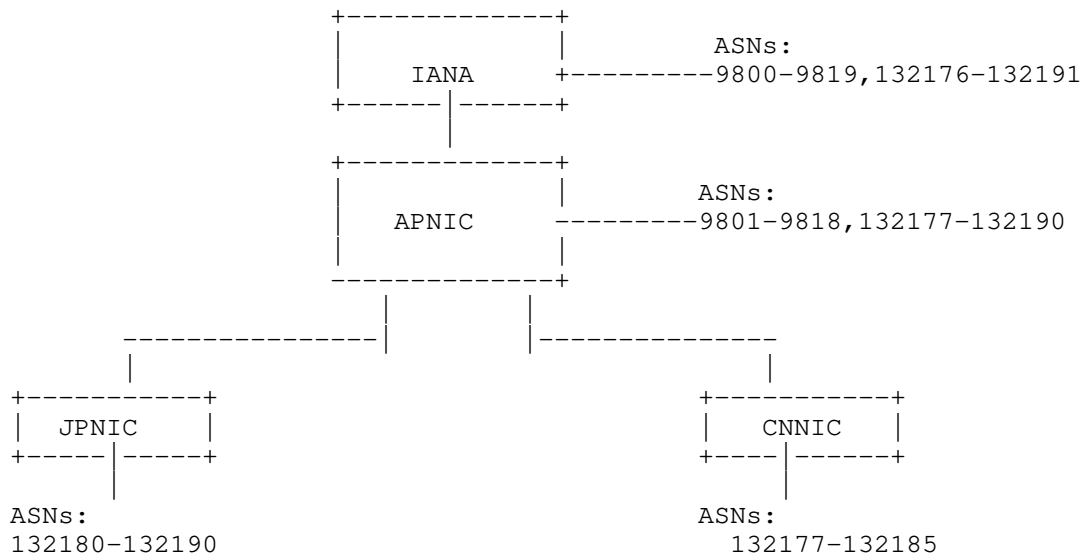


Figure 7: The network architecture of intersection case

Our test result (as described in Figure 8) shows that both CNNIC and JPNIC can receive these ASNs.

```

root@ubuntu:~# rpki -i apnic show_child_resources
Child: cnic
  ASN: 132177-132185
  IPv4: 203.0.113.0/26,218.241.108.0/26
Child: jpn
  ASN: 132180-132190
  IPv4: 218.241.104.0/26

root@ubuntu:~#rpki -i cnic show_received_resources
Parent:      apnic
  notBefore: 2015-08-31T14:58:54Z
  notAfter:  2016-08-30T12:24:04Z
  URI:       rsync://192.168.137.139/rpki/iana/apnic/t4jbcdikZIF9RGeLpZ0fiBd59U
w.cer
  SIA URI:   rsync://192.168.137.139/rpki/iana/apnic/cnic/
  AIA URI:   rsync://192.168.137.139/rpki/iana/HFOgArj0R8dF5vZ-7beqS0CzT2w.cer
  ASN:       132177-132185
  IPv4:      203.0.113.0/26,218.214.108.0/26
  IPv6:
root@ubuntu:~# rpki -i jpn show_received_resources
Parent:      apnic
  notBefore: 2015-08-31T14:58:44Z
  notAfter:  2016-08-30T12:25:25Z
  URI:       rsync://192.168.137.139/rpki/iana/apnic/V4CxZVxBkwwIiBruCy7k7DESUv
g.cer
  SIA URI:   rsync://192.168.137.139/rpki/iana/apnic/jpn/
  AIA URI:   rsync://192.168.137.139/rpki/iana/HFOgArj0R8dF5vZ-7beqS0CzT2w.cer
  ASN:       132180-132090
  IPv4:      218.241.104.0/26
  IPv6:

```

Figure 8: The test result of intersection case

4.2.3. Subset case

For the subset case, the network architecture is described in Figure 9: APNIC allocates the ASN 65540 and a block of IP address (203.0.113.128/26) to JPNIC. Then it reallocated this resources to CNNIC. These resources are the subset of CNNIC's resource.

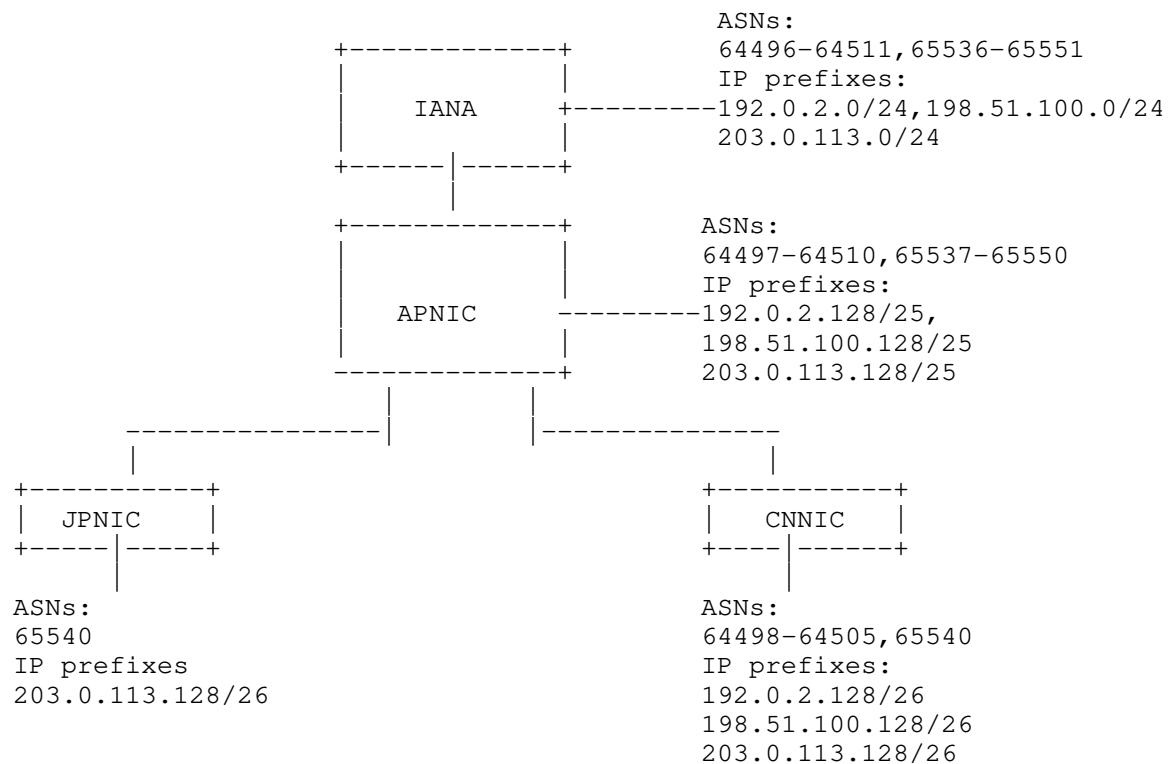


Figure 9: The network architecture of subset case

We test this case with the tools offered by <http://rpki.net/>. The result (as described in Fig 6) is that both CNNIC and JPNIC can receive these ASNs and IP Prefixes at the same time. It may result in resource confliction in the internet.

```
root@ubuntu:~# rpki -i apnic show_child_resources
Child: cnnic
ASN: 64498-64505,65540
IPv4: 192.0.2.128/26,198.51.100.128/26,203.0.113.128/26
Child: jpnice
ASN: 65540
IPv4: 203.0.113.128/26

root@ubuntu:~# rpki -i cnnic show_received_resources
Parent: apnic
notBefore: 2015-07-15T15:37:58Z
notAfter: 2016-07-14T15:36:05Z
URI: rsync://localhost/rpki/iana/apnic/BqHiZw817JRhXby5cljw-Iy75c4.cer
SIA URI: rsync://localhost/rpki/iana/apnic/cnnic/
AIA URI: rsync://localhost/rpki/iana/RAseYE67qlpBd34u5UqhMjwq8c0.cer
ASN: 64498-64505,65540
IPv4: 192.0.2.128/26,198.51.100.128/26,203.0.113.128/26
IPv6:

root@ubuntu:~# rpki -i jpnice show_received_resources
Parent: apnic
notBefore: 2015-07-15T15:25:54Z
notAfter: 2016-07-14T15:20:04Z
URI: rsync://localhost/rpki/iana/apnic/NSt9KXs-a2py_OGZl0l4fipm1lQ.cer
SIA URI: rsync://localhost/rpki/iana/apnic/jpnice/
AIA URI: rsync://localhost/rpki/iana/RAseYE67qlpBd34u5UqhMjwq8c0.cer
ASN: 65540
IPv4: 203.0.113.128/26
IPv6:
```

Figure 10: The test result of subset case

CNNIC and JPNIC could receive the same resources assigned by APNIC at the same time. So the test result verify that the same resources could be allocated to different sub-nodes simultaneously.

4.3. Resource transfer

This issue is described in [I-D.ymbk-sidr-transfer]. The most serious problem caused by it is the address reassignment described above.

5. Solutions

In order to avoid the above issues or reduce the related side-effects, the following two solutions may be needed.

5.1. The CA function enhancement

We have designed a mechanism to enhance the CA function to avoid the above misoperation or malicious operation. The detail information will be given in the future.

5.2. The RP function enhancement

The enhancement of RP function is needed to discover these resource assignment errors.

6. Security Considerations

TBD.

7. IANA Considerations

This draft does not request any IANA action.

8. Acknowledgements

The authors would like to thanks the valuable comments made by XXX and other members of sidr WG.

This document was produced using the xml2rfc tool [RFC2629].

9. References

9.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC6480] Lepinski, M. and S. Kent, "An Infrastructure to Support Secure Internet Routing", RFC 6480, DOI 10.17487/RFC6480, February 2012, <<http://www.rfc-editor.org/info/rfc6480>>.
- [RFC6482] Lepinski, M., Kent, S., and D. Kong, "A Profile for Route Origin Authorizations (ROAs)", RFC 6482, DOI 10.17487/RFC6482, February 2012, <<http://www.rfc-editor.org/info/rfc6482>>.
- [RFC6486] Austein, R., Huston, G., Kent, S., and M. Lepinski, "Manifests for the Resource Public Key Infrastructure (RPKI)", RFC 6486, DOI 10.17487/RFC6486, February 2012, <<http://www.rfc-editor.org/info/rfc6486>>.

9.2. Informative References

- [I-D.ymbk-sidr-transfer]
Austein, R., Bush, R., Huston, G., and G. Michaelson,
"Resource Transfer in the Resource Public Key
Infrastructure", draft-ymbk-sidr-transfer-01 (work in
progress), July 2015.
- [RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629,
DOI 10.17487/RFC2629, June 1999,
<<http://www.rfc-editor.org/info/rfc2629>>.
- [RFC5914] Housley, R., Ashmore, S., and C. Wallace, "Trust Anchor
Format", RFC 5914, DOI 10.17487/RFC5914, June 2010,
<<http://www.rfc-editor.org/info/rfc5914>>.

Authors' Addresses

Yu Fu
CNNIC
No.4 South 4th Street, Zhongguancun
Hai-Dian District, Beijing, 100190
P.R. China

Email: fuyu@cnnic.cn

Zhiwei Yan
CNNIC
No.4 South 4th Street, Zhongguancun
Beijing, 100190
P.R. China

Email: yanzhiwei@cnnic.cn

Xiaowei Liu
CNNIC
No.4 South 4th Street, Zhongguancun
Beijing, 100190
P.R. China

Email: liuxiaowei@cnnic.cn

Cuicui Wang
CNNIC
No.4 South 4th Street, Zhongguancun
Hai-Dian District, Beijing, 100190
P.R. China

Email: wangcuicui@cnnic.cn

Network Working Group
Internet-Draft
Updates: 6487 (if approved)
Intended status: Standards Track
Expires: April 11, 2016

G. Huston
G. Michaelson
APNIC
October 9, 2015

Update to RPKI Validation
draft-huston-sidr-validity-00.txt

Abstract

This document updates the RPKI certificate validation procedure as specified in Section 7.2 of RFC6487.

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 April 11, 2016.

Copyright Notice

Copyright (c) 2015 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. Introduction	3
2. The RPKI Certification Path Validation	3
3. Security Considerations	5
4. IANA Considerations	5
5. Acknowledgements	5
6. Normative References	5
Authors' Addresses	5

1. Introduction

This document updates the RPKI certificate validation procedure as specified in Section 7.2 of [RFC6487], by replacing the section 7.2 of [RFC6487] with the specification contained here.

2. The RPKI Certification Path Validation

Validation of signed resource data using a target resource certificate, and a specific set of number resources, consists of verifying that the digital signature of the signed resource data is valid, using the public key of the target resource certificate, and also validating the resource certificate in the context of the RPKI, using the path validation process. This path validation process verifies, among other things, that a prospective certification path (a sequence of n certificates) satisfies the following conditions:

1. for all ' x ' in $\{1, \dots, n-1\}$, the Subject of certificate ' x ' is the Issuer of certificate (' x ' + 1);
2. certificate '1' is issued by a trust anchor;
3. certificate ' n ' is the certificate to be validated; and
4. for all ' x ' in $\{1, \dots, n\}$, certificate ' x ' is valid for the specific set of resources.

RPKI validation for a specific set of resources entails verifying that all of the following conditions hold, in addition to the Certification Path Validation criteria specified in Section 6 of [RFC5280]:

1. The certificate can be verified using the Issuer's public key and the signature algorithm
2. The current time lies within the certificate's Validity From and To values.

3. The certificate contains all fields that MUST be present, as specified by [RFC6487], and contains values for selected fields that are defined as allowable values by this specification.
4. No field, or field value, that the [RFC6487] specification defines as MUST NOT be present is used in the certificate.
5. The Issuer has not revoked the certificate. A revoked certificate is identified by the certificate's serial number being listed on the Issuer's current CRL, as identified by the CRLDP of the certificate, the CRL is itself valid, and the public key used to verify the signature on the CRL is the same public key used to verify the certificate itself.
6. The resource extension data contained in this certificate "encompasses" the entirety of the resources in the specific resource set ("encompass" in this context is defined in Section 7.1 of [RFC6487]).
7. The Certification Path originates with a certificate issued by a trust anchor, and there exists a signing chain across the Certification Path where the Subject of Certificate 'x' in the Certification Path matches the Issuer in Certificate 'x + 1' in the Certification Path, and the public key in Certificate 'x' can verify the signature value in Certificate 'x+1'.

A certificate validation algorithm MAY perform these tests in any chosen order.

There exists the possibility of encountering certificate paths that are arbitrarily long, or attempting to generate paths with loops as means of creating a potential denial-of-service (DOS) attack on an Relying Party (RP). An RP executing this procedure MAY apply further heuristics to guide the certification path validation process to a halt in order to avoid some of the issues associated with attempts to validate such malformed certification path structures.

Implementations of resource certificate validation MAY halt with a validation failure if the certification path length exceeds a locally defined configuration parameter.

3. Security Considerations

This update is intended to improve the robustness of the RPKI framework by altering the procedure of the original validation path that included an "encompassing" condition applied pairwise to the certificates used in the validation path.

The intent of this update is to ensure that all certificates on a validation path encompass the resources that are included in the validation query, but to remove the "encompassing" constraint on the resources used in pairwise comparison. This change to the validation procedure reduces the criticality of strict orchestration of the sequence of certificate issuance and revocation in those circumstances, and can thereby improve the robustness of the RPKI as a consequence, without altering the underlying semantics of the association of a public key value across a collection of number resources.

4. IANA Considerations

No updates to the registries are suggested by this document.

5. Acknowledgements

Thanks

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

Authors' Addresses

Geoff Huston
Asia Pacific Network Information Centre
6 Cordelia St
South Brisbane, QLD 4101
Australia

Phone: +61 7 3858 3100
Email: gih@apnic.net

George Michaelson
Asia Pacific Network Information Centre
6 Cordelia St
South Brisbane, QLD 4101
Australia

Phone: +61 7 3858 3100
Email: ggm@apnic.net

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 21, 2016

T. Bruijnzeels
O. Muravskiy
RIPE NCC
B. Weber
Cobenian
R. Austein
Dragon Research Labs
D. Mandelberg
BBN Technologies
October 19, 2015

RPKI Repository Delta Protocol
draft-ietf-sidr-delta-protocol-01

Abstract

In the Resource Public Key Infrastructure (RPKI), certificate authorities publish certificates, including end entity certificates, and CRLs to repositories on repository servers. Relying Parties (RP) retrieve the published information from the repository and MAY store it in a cache. This document specifies a delta protocol which provides relying parties with a mechanism to query a repository for changes, thus enabling the RP to keep its state in sync with the repository.

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 April 21, 2016.

Copyright Notice

Copyright (c) 2015 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. Requirements notation	2
2. Introduction	3
3. RPKI Repository Delta Protocol Implementation	3
3.1. Informal Overview	3
3.2. Update Notification File	4
3.2.1. Purpose	4
3.2.2. Cache Concerns	5
3.2.3. File Format and Validation	5
3.2.4. Repository Server Initialisation	6
3.2.5. Publishing Updates	6
3.3. Snapshot File	7
3.3.1. Purpose	7
3.3.2. Cache Concerns	7
3.3.3. File Format and Validation	7
3.4. Delta File	8
3.4.1. Purpose	8
3.4.2. Cache Concerns	9
3.4.3. File Format and Validation	9
3.5. SIA for CA certificates	10
4. Relying Party Use	10
4.1. Full Synchronisation	11
4.2. Processing Deltas	11
5. XML Schema	11
6. Security Considerations	13
7. IANA Considerations	13
8. Acknowledgements	13
9. Normative References	13
Authors' Addresses	14

1. Requirements notation

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].

2. Introduction

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

This document specifies an alternative repository access protocol based on notification, snapshot and delta files that a RP can retrieve over HTTPS. This allows RPs to perform a full (re-)synchronisation of their local copy of the repository using snapshot files. However, typically RPs will use delta files to keep their local repository updated after initial synchronisation.

This protocol is designed to be consistent with the publication protocol [I-D.ietf-sidr-publication] and treats publication events of one or more repository objects as discrete events that can be communicated to relying parties. This approach helps to minimize the amount of data that traverses the network and thus helps minimize the amount of time until repository convergence occurs. This protocol also provides a standards based way to obtain consistent, point in time views of a single repository, eliminating a number of consistency related issues. Finally, this approach allows these discrete events to be communicated as immutable files, so that caching infrastructure can be used to reduce the load on a repository server when a large a number of relying parties are querying it.

3. RPKI Repository Delta Protocol Implementation

3.1. Informal Overview

Certification Authorities (CA) in the RPKI use a repository server to publish their RPKI products, such as manifests, CRLs, signed certificates and RPKI signed objects. This repository server may be remote, or embedded in the CA engine itself. Certificates in the RPKI that use a repository server that supports this delta protocol include a special Subject Information Access (SIA) pointer referring to a notification file.

The notification file includes a globally unique session_id in the form of a version 4 UUID, and serial number that can be used by the Relying Party (RP) to determine if it and the repository are synchronised. Furthermore it includes a link to the most recent

complete snapshot of current objects that are published by the repository servers, and a list of links to delta files, for each revision starting at a point determined by the repository server, up to the current revision of the repository.

A RP that learns about a notification file location for the first time can download it, and then proceed to download the latest snapshot file, and thus create a local copy of the repository that is in sync with the repository server. The RP should remember the location of this notification file, the session_id and current serial number.

RPs are encouraged to re-fetch this notification file at regular intervals, but not more often than once per minute. After re-fetching the notification file, the RP may find that there are one or more delta files available that allow it to synchronise its local repository with the current state of the repository server. If no contiguous chain of deltas from RP's serial to the latest repository serial is available, or if the session_id has changed, the RP should perform a full resynchronisation instead.

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

The repository server may use caching infrastructure to reduce its load. It should be noted that snapshots and deltas for any given session_id and serial number contain an immutable record of the state of the repository server at a certain point in time. For this reason these files can be cached indefinitely. Notification files are polled by RPs to discover if updates exist, and for this reason notification files may not be cached for longer than one minute.

3.2. Update Notification File

3.2.1. Purpose

The update notification file is used by RPs to discover whether any changes exist between the state of the repository and the RP's cache. It describes the location of the files containing the snapshot and incremental deltas which can be used by the RP to synchronise with the repository.

3.2.2. Cache Concerns

A repository server MAY use caching infrastructure to cache the notification file and reduce the load of HTTPS requests. However, since this file is used by RPs to determine whether any updates are available the repository server MUST ensure that this file is not cached for longer than 1 minute. An exception to this rule is that it is better to serve a stale notification file, then no notification file.

How this is achieved exactly depends on the caching infrastructure used. In general a repository server may find certain http headers to be useful, such as: Cache-Control: max-age=60. Another approach can be to have the repository server push out new versions of the notification file to the caching infrastructure when appropriate.

Relying Parties SHOULD not cache the notification file for longer than 1 minute, regardless of the headers set by the repository server or CDN.

3.2.3. File Format and Validation

Example notification file:

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

The following validation rules must be observed when creating or parsing notification files:

- o A RP MUST NOT process any update notification file that is not well formed, or which does not conform to the RELAX NG schema outlined in Section 5 of this document.
- o The XML namespace MUST be `http://www.ripe.net/rpki/rrdp`
- o The encoding MUST be `us-ascii`
- o The version attribute in the notification root element MUST be 1
- o The session_id attribute MUST be a random version 4 UUID unique to this session

- o The serial attribute must be an unbounded, unsigned positive integer indicating the current version of the repository.
- o The notification file MUST contain exactly one 'snapshot' element for the current repository version.
- o If delta elements are included they MUST form a contiguous sequence of serial numbers starting at a revision determined by the repository server, up to the serial number mentioned in the notification element.

3.2.4. Repository Server Initialisation

When the repository server (re-) initialises it MUST generate a new random version 4 UUID to be used as the session_id. Furthermore it MUST then generate a snapshot file for serial number ONE for this new session that includes all currently known published objects that the repository server is responsible for. This snapshot file MUST be made available at a URL that is unique to this session and version, so that it can be cached indefinitely. The format and caching concerns for snapshot files are explained in more detail below in Section 3.3. After the snapshot file has been published the repository server MUST publish a new notification file that contains the new session_id, has serial number ONE, has one reference to the snapshot file that was just published, and that contains no delta references.

3.2.5. Publishing Updates

Whenever the repository server receives updates from a CA it SHOULD generate new snapshot and delta files and publish a new notification file as follows:

- o The new repository serial MUST be one greater than the current repository serial.
- o A new delta file MUST be generated for this new serial. This delta file MUST include all new, replaced and withdrawn objects, as a single change set.
- o This delta file MUST be made available at a URL that is unique to this session and version, so that it can be cached indefinitely.
- o The repository server MUST also generate a new snapshot file for this new serial. This file MUST contain all publish elements for all current objects.

- o The snapshot file MUST be made available at a URL that is unique to this session and new version, so that it can be cached indefinitely.
- o A new notification file MUST be created by the repository server. This new notification file MUST include a reference to the new snapshot file. The file SHOULD also include available delta files for this and previous updates. However, the server MUST NOT include more delta files than, when combined, exceed the size of the current snapshot.

If the repository server is not capable of performing the above for some reason, then it MUST perform a full re-initialisation, as explained above in Section 3.2.4.

3.3. Snapshot File

3.3.1. Purpose

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

3.3.2. Cache Concerns

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

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

Repository servers MAY delete old snapshot files one hour after they are no longer included in the notification file.

3.3.3. File Format and Validation

Example snapshot file:

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

The following rules must be observed when creating or parsing snapshot files:

- o A RP MUST NOT process any snapshot file that is not well formed, or which does not conform to the RELAX NG schema outlined in Section 5 of this document.
- o The XML namespace MUST be `http://www.ripe.net/rpki/rrdp`.
- o The encoding MUST be `us-ascii`.
- o The version attribute in the notification root element MUST be 1
- o The `session_id` attribute MUST match the expected `session_id` in the reference in the notification file.
- o The `serial` attribute MUST match the expected `serial` in the reference in the notification file.
- o Note that the `publish` element is defined in the publication protocol [I-D.ietf-sidr-publication]

3.4. Delta File

3.4.1. Purpose

An incremental delta file contains all changes for exactly one serial increment of the repository server. In other words a single delta will typically include all the new objects, updated objects and withdrawn objects that a Certification Authority sent to the repository server. In its simplest form the update could concern only a single object, but it is recommended that CAs send all changes

for one of their key pairs: i.e. updated objects as well as a new manifest and CRL as one atomic update message.

3.4.2. Cache Concerns

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

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

Repository servers **MAY** delete old delta files one hour after they are no longer included in the notification file.

3.4.3. File Format and Validation

Example delta file:

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

Note that a formal RELAX NG specification of this file format is included later in this document. A RP **MUST NOT** process any delta file that is incomplete or not well formed.

The following validation rules must be observed when creating or parsing delta files:

- o A RP **MUST NOT** process any delta file that is not well formed, or which does not conform to the RELAX NG schema outlined in Section 5 of this document.

- o The XML namespace MUST be `http://www.ripe.net/rpki/rrdp`.
- o The encoding MUST be `us-ascii`.
- o The version attribute in the delta root element MUST be 1
- o The `session_id` attribute MUST be a random version 4 UUID unique to this session
- o The `session_id` attribute MUST match the expected `session_id` in the reference in the notification file.
- o The serial attribute MUST match the expected serial in the reference in the notification file.
- o Note that the `publish` and `withdraw` elements are defined in the publication protocol [I-D.ietf-sidr-publication]

3.5. SIA for CA certificates

Certificate Authorities that use this delta protocol MUST have an instance of an SIA `AccessDescription` in addition to the ones defined in [RFC6487],

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

This extension MUST use an `accessMethod` of `id-ad-rpkiNotify`, see: [IANA-AD-NUMBERS],

```
id-ad OBJECT IDENTIFIER ::= { id-pkix 48 }  
id-ad-rpkiNotify OBJECT IDENTIFIER ::= { id-ad 13 }
```

The `accessLocation` MUST be a URI [RFC3986], using the 'https' scheme, that will point to the update notification file for the repository server that publishes the products of this CA certificate.

Relying Parties that do not support this delta protocol MUST NOT reject a CA certificate merely because it has an SIA extension containing this new kind of `AccessDescription`.

4. Relying Party Use

4.1. Full Synchronisation

When a Relying Party first encounters a notification file URI as an SIA of a certificate that it has validated it SHOULD retrieve the notification file and download the latest snapshot to get in sync with the current version of the repository server.

4.2. Processing Deltas

It is RECOMMENDED that the RP notes the URI, session_id and serial number when it first learns about a notification file. The RP MAY then poll the file to discover updates, but SHOULD NOT poll more frequently than once per minute.

If the RP finds that the session_id has changed, or if it cannot find a contiguous chain of links to delta files from its current serial to repository server's current serial, then it MUST perform a full synchronisation instead of continuing to process deltas.

If the RP finds a contiguous chain of links to delta files from its current serial to the repository server's current serial, and the session_id has not changed, it should download all missing delta files. If any delta file cannot be downloaded, or if no such chain of deltas is available, or the session_id has changed, then the RP MUST perform a full synchronisation instead.

New objects found in delta files can be added to the RPs local copy of the repository. However, it is RECOMMENDED that the RP treats object updates and withdraws with some skepticism. A compromised repository server may not have access to the certification authorities' keys, but it can pretend valid objects have been withdrawn. Therefore it may be preferred to use a strategy where local copies of objects are only discarded when the RP is sure that they are no longer relevant, e.g. the CA has explicitly removed the objects in a recent valid manifest and revoked them in a recent valid CRL (unless they have expired).

5. XML Schema

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

```
#  
# RelaxNG schema for RPKI Repository Delta Protocol (RRDP).  
#  
  
default namespace = "http://www.ripe.net/rpki/rrdp"
```

```

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

```

Notification file: lists current snapshots and deltas

```

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

```

Snapshot segment: think DNS AXFR.

```

start |= element snapshot {
  attribute version    { version },
  attribute session_id { uuid },
  attribute serial     { serial },
  element publish     {
    attribute uri { uri },
    base64
  }*
}

```

Delta segment: think DNS IXFR.

```

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

```

```

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

```

```
    base64
  }

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

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

6. Security Considerations

TBD

7. IANA Considerations

This document has no actions for IANA.

8. Acknowledgements

TBD

9. Normative References

[I-D.ietf-sidr-publication]

Weiler, S., Sonalker, A., and R. Austein, "A Publication Protocol for the Resource Public Key Infrastructure (RPKI)", draft-ietf-sidr-publication-07 (work in progress), September 2015.

[IANA-AD-NUMBERS]

"SMI Security for PKIX Access Descriptor",
<<http://www.iana.org/assignments/smi-numbers/smi-numbers.xhtml#smi-numbers-1.3.6.1.5.5.7.48>>.

[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>>.

[RFC3986]

Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005,
<<http://www.rfc-editor.org/info/rfc3986>>.

- [RFC6481] Huston, G., Loomans, R., and G. Michaelson, "A Profile for Resource Certificate Repository Structure", RFC 6481, DOI 10.17487/RFC6481, February 2012, <<http://www.rfc-editor.org/info/rfc6481>>.
- [RFC6486] Austein, R., Huston, G., Kent, S., and M. Lepinski, "Manifests for the Resource Public Key Infrastructure (RPKI)", RFC 6486, DOI 10.17487/RFC6486, February 2012, <<http://www.rfc-editor.org/info/rfc6486>>.
- [RFC6487] Huston, G., Michaelson, G., and R. Loomans, "A Profile for X.509 PKIX Resource Certificates", RFC 6487, DOI 10.17487/RFC6487, February 2012, <<http://www.rfc-editor.org/info/rfc6487>>.
- [RFC6488] Lepinski, M., Chi, A., and S. Kent, "Signed Object Template for the Resource Public Key Infrastructure (RPKI)", RFC 6488, DOI 10.17487/RFC6488, February 2012, <<http://www.rfc-editor.org/info/rfc6488>>.

Authors' Addresses

Tim Bruijnzeels
RIPE NCC

Email: tim@ripe.net

Oleg Muravskiy
RIPE NCC

Email: oleg@ripe.net

Bryan Weber
Cobenian

Email: bryan@cobenian.com

Rob Austein
Dragon Research Labs

Email: sra@hactrn.net

David Mandelberg
BBN Technologies

Email: david@mandelberg.org

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 21, 2016

R. Austein
Dragon Research Labs
October 19, 2015

An Out-Of-Band Setup Protocol For RPKI Production Services
draft-ietf-sidr-rpki-oob-setup-03

Abstract

This note describes a simple out-of-band protocol to ease setup of the 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 secure means.

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 BPKI keying material.

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 April 21, 2016.

Copyright Notice

Copyright (c) 2015 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. Introduction	2
2. Overview of the BPKI	3
3. Protocol Elements	4
3.1. Nomenclature	4
3.2. Common Protocol Elements	5
3.3. Protocol Messages	5
3.3.1. <child_request/>	5
3.3.2. <parent_response/>	6
3.3.3. <publisher_request/>	8
3.3.4. <repository_response/>	9
3.4. <authorization/>	10
3.5. <error/>	11
4. Protocol Walk-Through	12
5. IANA Considerations	17
6. Security Considerations	17
7. Acknowledgements	17
8. Normative References	17
Appendix A. RelaxNG Schema	18
Author's Address	19

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 ([I-D.ietf-sidr-publication]).

The basic function of this protocol is public key exchange, in the form of self-signed BPKI 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 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. Overview of the BPKI

Several protocols related to RPKI provisioning use signed 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 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]
```

[[Need some text detailing required and allowed values in the certificates: 2048-bit RSA, what extensions, But once we go there we also have to provide a path for algorithm agility.]]

3. Protocol Elements

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

3.1. Nomenclature

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 [I-D.ietf-sidr-publication].

Repository: An entity acting in the server role of the publication protocol defined in [I-D.ietf-sidr-publication].

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.

3.2. Common Protocol Elements

The first XML attribute in each message is a version field. This document describes version 1 of the protocol.

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 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.

3.3. Protocol Messages

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

3.3.1. <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.

child_handle: The child_handle attribute is what the child calls itself. This is just a hint from the child to the parent, 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
  child_handle="Bob"
  version="1"
  xmlns="http://www.hactrn.net/uris/rpki/rpki-setup/">
  <child_bpki_ta>
    R29kIGlzIHJlYWwgdW5sZXNzIGRlY2xhcmVkaGludGVnZXI=
  </child_bpki_ta>
</child_request>
-----
```

3.3.2. <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.

service_uri: The service_uri attribute contains an HTTP URL 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 might or might not 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 a `<referral/>` element is present, it suggests a third-party publication services that the child might use, and contains:

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 3.4 for details on the authorization token.

```
-----
<parent_response
  child_handle="Bob-42"
  parent_handle="Alice"
  service_uri="http://a.example/up-down/Alice/Bob-42"
  version="1"
  xmlns="http://www.hactrn.net/uris/rpki/rpki-setup/">
  <parent_bpki_ta>
    WW91IGNhbiBoYWNRIGFueXRoaW5nIHlvdSB3YW50IHdpdGggVEVDTyBhbmQgRERU
  </parent_bpki_ta>
  <offer/>
</parent_response>
-----
```

```
-----
<parent_response
  child_handle="Carol"
  parent_handle="Bob"
  service_uri="http://bob.example/up-down/Bob/Carol"
  version="1"
  xmlns="http://www.hactrn.net/uris/rpki/rpki-setup/">
  <parent_bpki_ta>
    R29kIGlzIHJlYWwgYW5sZXNzIGRlY2xhcmVkaGludGVnZXI=
  </parent_bpki_ta>
  <referral
    referrer="Alice/Bob-42">
    R28sIGxlbWlpbmdzLCBnbyE=
  </referral>
</parent_response>
-----
```

3.3.3. <publisher_request/>

The <publisher_request/> message is a setup request from a publisher to a repository.

Fields in the <publisher_request/> message:

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

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 3.4 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 3.3.2). 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
  publisher_handle="Bob"
  version="1"
  xmlns="http://www.hactrn.net/uris/rpki/rpki-setup/">
  <publisher_bpki_ta>
    R29kIGlzIHJlYWgdW5sZXNzIGRlY2xhcmVklGludGVnZXI=
  </publisher_bpki_ta>
</publisher_request>
-----
```

3.3.4. <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.

service_uri: The service_uri attribute contains an HTTP URL that the publisher should contact for publication service ([I-D.ietf-sidr-publication]).

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 RRDP notification file covering the publication space allocated to the publisher ([I-D.ietf-sidr-delta-protocol]).

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

```

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

```

3.4. <authorization/>

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

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

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 <bpki_ta/> 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. The identity is represented as a self-signed X.509 BPKI certificate.

```

-----
<authorization
  authorized_sia_base="rsync://a.example/rpki/Alice/Bob-42/Carol/"
  version="1"
  xmlns="http://www.hactrn.net/uris/rpki/rpki-setup/">
    SSd2ZSB0YWQgZnVuIGJlZm9yZS4gIFRoXMGaXNuJ3QgaXQu
  </authorization>
-----

```

3.5. <error/>

The <error/> element is an optional message which can be used in response to any of the core protocol messages described in Section 3.3.

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 (email, 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
  reason="refused"
  version="1"
  xmlns="http://www.hactrn.net/uris/rpki/rpki-setup/">
  <child_request
    child_handle="Carol">
    <child_bpki_ta>
      SSd2ZSBoYWQgZnVuIGJlZm9yZS4gIFRoaxMgaXNuJ3QgaXQu
    </child_bpki_ta>
  </child_request>
</error>
-----
```

4. 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 a RPKI tree, Bob wants to get address and 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 generates his or her own self-signed BPKI certificate.

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

```
-----
<child_request
  child_handle="Bob"
  version="1"
  xmlns="http://www.hactrn.net/uris/rpki/rpki-setup/">
  <child_bpki_ta>
    R29kIGlziIHJlYWwgZD5sZXNzIGRlY2xhcmVklGludGVnZXI=
  </child_bpki_ta>
</child_request>
-----
```

- o Bob's preferred handle is "Bob", so Bob uses that when setting child_handle.
- o <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
  child_handle="Bob-42"
  parent_handle="Alice"
  service_uri="http://a.example/up-down/Alice/Bob-42"
  version="1"
  xmlns="http://www.hactrn.net/uris/rpki/rpki-setup/">
  <parent_bpki_ta>
    WW91IGNhbiBoYWNRIGFueXRoaW5nIHlvdSB3YW50IHdpdGggVEVDYyBhbmQgRERU
  </parent_bpki_ta>
  <offer/>
</parent_response>
-----
```

o <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
  publisher_handle="Bob"
  version="1"
  xmlns="http://www.hactrn.net/uris/rpki/rpki-setup/">
  <publisher_bpki_ta>
    R29kIGlzIHJlYWwgYW5sZXNzIGRlY2xhcmVkaW50IGludGVnZXI=
  </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
  publisher_handle="Alice/Bob-42"
  rrdp_notification_uri="https://rpki.example/rrdp/notify.xml"
  service_uri="http://a.example/publication/Alice/Bob-42"
  sia_base="rsync://a.example/rpki/Alice/Bob-42/"
  version="1"
  xmlns="http://www.hactrn.net/uris/rpki/rpki-setup/">
  <repository_bpki_ta>
    WW91IGNhbiBoYWNRIGFueXRoaW5nIHlvdSB3YW50IHdpdGggVEVDTyBhbmQgRERU
  </repository_bpki_ta>
</repository_response>
-----
```

Bob should now have everything he needs to talk to Alice both for provisioning and for 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
  child_handle="Carol"
  version="1"
  xmlns="http://www.hactrn.net/uris/rpki/rpki-setup/">
  <child_bpki_ta>
    SSd2ZSB0YWQgZnVuIGJlZm9yZS4gIFRoXMGaXNuJ3QgaXQu
  </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
  child_handle="Carol"
  parent_handle="Bob"
  service_uri="http://bob.example/up-down/Bob/Carol"
  version="1"
  xmlns="http://www.hactrn.net/uris/rpki/rpki-setup/">
  <parent_bpki_ta>
    R29kIGlzIHJlYWwgdW5sZXNzIGRlY2xhcmVkJGluZGVnZXI=
  </parent_bpki_ta>
  <referral
    referrer="Alice/Bob-42">
    R28sIGxlbWlpbmdzLCBnbyE=
  </referral>
</parent_response>
-----
```

Bob's response includes a <referral/> element with a referrer attribute of "Alice/Bob-42", since that's Bob's name to 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
  authorized_sia_base="rsync://a.example/rpki/Alice/Bob-42/Carol/"
  version="1"
  xmlns="http://www.hactrn.net/uris/rpki/rpki-setup/">
  SSd2ZSBoYWQgZnVuIGJlZm9yZS4gIFRoXMGaXNuJ3QgaXQu
</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
  publisher_handle="Carol"
  version="1"
  xmlns="http://www.hactrn.net/uris/rpki/rpki-setup/">
  <publisher_bpki_ta>
    SSd2ZSBoYWQgZnVuIGJlZm9yZS4gIFRoXMaXNuJ3QgaXQu
  </publisher_bpki_ta>
  <referral
    referrer="Alice/Bob-42">
      R28sIGxlbWlpbmdzLCBnbyE=
    </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 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
  publisher_handle="Alice/Bob-42/Carol"
  service_uri="http://a.example/publication/Alice/Bob-42/Carol"
  sia_base="rsync://a.example/rpki/Alice/Bob-42/Carol/"
  version="1"
  xmlns="http://www.hactrn.net/uris/rpki/rpki-setup/">
  <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.

5. IANA Considerations

Blah.

6. 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.

7. 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 whose name the author has temporarily forgotten.

8. Normative References

[I-D.ietf-sidr-delta-protocol]

Bruijnzeels, T., Muravskiy, O., Weber, B., Austein, R., and D. Mandelberg, "RPKI Repository Delta Protocol", draft-ietf-sidr-delta-protocol-01 (work in progress), October 2015.

[I-D.ietf-sidr-publication]

Weiler, S., Sonalker, A., and R. Austein, "A Publication Protocol for the Resource Public Key Infrastructure (RPKI)", draft-ietf-sidr-publication-07 (work in progress), September 2015.

[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, May 2008.

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

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

Appendix A. RelaxNG Schema

Here is a RelaxNG schema describing the protocol elements.

```
# $Id: rpki-setup.rnc 3429 2015-10-14 23:46:50Z sra $

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 )* }

authorization_token = base64
bpki_ta = base64

start |= element child_request {
  attribute version { version },
  attribute child_handle { handle },
  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 },
  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 },
  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 }?,
  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?
}
```

Author's Address

Rob Austein
Dragon Research Labs

Email: sra@hactrn.net

Network Working Group
Internet-Draft
Intended status: Informational
Expires: April 11, 2016

G. Huston
G. Michaelson
APNIC
C. Martinez
LACNIC
T. Bruijnzeels
RIPE NCC
A. Newton
ARIN
A. Aina
AFRINIC
October 9, 2015

RPKI Validation Reconsidered
draft-ietf-sidr-rpki-validation-reconsidered-02.txt

Abstract

This document reviews the certificate validation procedure specified in RFC6487 and highlights aspects of operational fragility in the management of certificates in the RPKI.

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 April 11, 2016.

Copyright Notice

Copyright (c) 2015 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. Introduction	3
2. Certificate Validation in the RPKI	3
3. Operational Considerations	4
4. Alternative Approaches	7
5. An Amended RPKI Certification Validation Process	7
6. Security Considerations	9
7. IANA Considerations	9
8. Acknowledgements	10
9. References	10
9.1. Normative References	10
9.2. Informative References	10
Authors' Addresses	10

1. Introduction

This document reviews the certificate validation procedure specified in RFC6487 and highlights aspects of operational fragility in the management of certificates.

2. Certificate Validation in the RPKI

As currently defined in section 7.2 of [RFC6487], validation of PKIX certificates that conform to the RPKI profile relies on the use of a path validation process where each certificate in the validation path is required to meet the certificate validation criteria. This is a recursively defined validation process where, in the context of an ordered sequence of certificates, as defined by each pair of certificates in this sequence having a common Issuer and Subject Name respectively, a certificate is defined as valid if it satisfies basic validation criteria relating to the syntactic correctness, currency of validity dates and similar properties of the certificate itself, as described in [RFC5280], and also that it satisfies certain additional criteria with respect to the previous certificate in the sequence (the Issuer part of the pair), and that this previous certificate is itself a valid certificate using the same criteria. This process is applied to all certificates in the sequence apart from the initial sequence element, which is required to be a Trust Anchor.

For RPKI certificates, the additional criteria relating to the previous certificate in this sequence is that the certificate's number resource set, as defined in [RFC3779], is "encompassed" by the number resource set contained in the previous certificate.

Because [RFC6487] validation demands that all resources in a certificate be valid under the parent (and recursively, to the root), a digitally signed attestation, such as a Route Origin Authorization (ROA) object [RFC6482], which refers only to a subset of RFC3779-specified resources from that certificate validation chain can be concluded to be invalid, but not by virtue of the relationship between the RFC3779 extensions of the certificates on the putative certificate validation path and the resources in the ROA, but by other resources described in these certificates where the "encompassing" relationship of the resources does not hold. Any such invalidity along the certificate validation chain can cause this outcome, not just at the immediate parent of the end entity certificate that attests to the key used to sign the ROA.

For example, in the certificate sequence:

Certificate 1:

Issuer A, Subject B, Resources 192.0.2.0/24, AS64496-AS64500

Certificate 2:

Issuer B, Subject C, Resources 192.0.2.0/24/24, AS64496-AS64511

Certificate 3:

Issuer C, Subject D, Resources 192.0.2.0/24

Certificate 3 is considered to be an invalid certificate, because the resources in Certificate 2 are not encompassed by the resources in Certificate 1, by virtue of certificate 2 describing the resources of the range AS64501 - AS64511 in this RFC3779 resource extension. Obviously, these Autonomous Systems numbers are not related to the IPv4 resources contained in Certificate 3.

Any non-encompassed resource set can cause this form or validation failure, whether it is an ASN, IPv4 or IPv6 resource, if it is not encompassed by the resource set in the parent (Issuer) certificate.

The underlying observation here is that this definition of certificate validation treats a collection of resources as inseparable, so that a single certificate containing a bundle of number resources is semantically distinct from an equivalent set of certificates where each certificate contains a single number resource. This semantic distinction between the whole and the sum of its parts is an artifice introduced by the particular choice of a certificate validation procedure used by the RPKI, as distinct from meeting any particular operational requirement, and the result is the introduction of operational fragility into the handling of RPKI certificates, particularly in the case where number resources are moved between the corresponding registries, as described here.

3. Operational Considerations

There are two areas of operational concern with the current RPKI validation definition.

The first is that of the robustness of the operational management procedures in the issuance of certificates. If a subordinate Certification Authority (CA) issues a certificate that contains an Internet Number Resource (INR) collection that is not either exactly equal to, or a strict subset of, its parent CA, then this issued certificate, and all subordinate certificates of this issued certificate are invalid. These certificates are not only defined as invalid when being considered to validate an INR that is not in the parent CA certificate, but are defined as invalid for all INRs in the

certificate.

This constraint creates a degree of operational fragility in the issuance of certificates, as all CA's are now required to exercise extreme care in the issuance and reissuance of certificates to ensure that at no time do they overclaim on the resources described in the parent CA, as the consequences of an operational lapse or oversight implies that all the subordinate certificates from the point of INR mismatch are invalid. It would be preferred if the consequences of such an operational lapse were limited in scope to the specific INRs that formed the mismatch, rather than including the entire set of INRs within the scope of damage from this point of mismatch downward across the entire sub-tree of descendant certificates in the RPKI certificate hierarchy.

The second operational consideration described here relates to the situation where a registry withdraws a resource from the current holder, and the resource is transferred to another registry, to be registered to a new holder in that registry. The reason why this is a consideration in operational deployments of the RPKI lies in the movement of the "home" registry of number resources during cases of mergers, acquisitions, business re-alignments, and resource transfers and the desire to ensure that during this movement all other resources can continue to be validated.

If the original registry's certification actions are simply to issue a new certificate for the current holder with a reduced resource set, and to revoke the original certificate, then there is a distinct possibility of encountering the situation illustrated by the example in the previous section. This is a result of an operational process for certificate issuance by the parent CA being de-coupled from the certificate operations of child CA.

This de-coupled operation of CAs introduces a risk of unintended third party damage: since a CA certificate can refer to holdings which relate to two or more unrelated subordinate certificates, if this CA certificate becomes invalid due to the reduction in the resources allocated to this CA relating to one subordinate resource set, all other subordinate certificates are invalid until the CA certificate is reissued with a reduced resource set.

In the example provided in the previous section, all subordinate certificates issued by CA B are invalid, including all certificates issued by CA C, until CA A issues a new certificate for CA B with a reduced resource set.

At the lower levels of the RPKI hierarchy the resource sets affected by such movements of resources may not encompass significantly large

pools of resources. However, as one ascends through this certification hierarchy towards the apex, the larger the resource set that is going to be affected by a period of invalidity by virtue of such uncoordinated certificate management actions. In the case of a Regional Internet Registry (RIR) or National Internet Registry (NIR), the potential risk arising from uncoordinated certification actions relating to a transfer of resources is that the entire set of subordinate certificates that refer to resources administered by the RIR or the NIR cannot be validated during this period.

Avoiding such situations requires that CA's adhere to a very specific ordering of certificate issuance. In this framework, the common registry CA that describes (directly or indirectly) the resources being shifted from one registry to the other, and also contains in subordinate certificates (direct or indirect) the certificates for both registries who are parties to the resource transfer has to coordinate a specific sequence of actions.

This common registry CA has to first issue a new certificate towards the "receiving" registry that adds to the RFC3779 extension resource set the specific resource being transferred into this receiving registry. The common registry CA then has to wait until all registries in the subordinate certificate chain to the receiving registry have also performed a similar issuance of new certificates, and in each case a registry must await the issuance of the immediate superior certificate with the augmented resource set before it, in turn, can issue its own augmented certificate to its subordinate CA. This is a "top down" issuance sequence."

It is possible for the common registry to issue a certificate to the "sending" registry with the reduced resource set at any time, but it should not revoke the previously issued certificate, nor overwrite this previously issued certificate in its repository publication point without specific coordination. Only when the common registry is assured that the top down certificate issuance process to the receiving registry CA chain has been completed can the common registry commence the revocation of the original certificate for the sending registry. However, it should not so until it is assured that the immediate subordinate registry CA in the path to the sending registry has issued a certificate with a reduced resource set, and so on. This implies that on the sending side the certificate issuance and revocation is a "bottom up" process.

If this process is not carefully followed, then the risk is that some or all of the subordinate certificates of this common registry CA will be unable to be validated until the entire process of certificate issuance and revocation has been completed. While this sequenced process is intended to preserve validity of certificates in

the RPKI, it is a complex, fragile and operationally cumbersome process.

The underlying consideration here is that the operational coordination of these certificate issuance and revocation actions to effect a smooth resource transfer across registries is mandated by the nature of the particular choice of certificate validation process described in [RFC6487].

4. Alternative Approaches

If the current definition of the RPKI certificate validation procedure is considered to introduce unacceptable levels of fragility and risk into the operational environment, what alternatives exist?

One approach is to remove the semantic requirement to consider the collection of resources in the extension field of the RPKI certificate as an indivisible bundle. This would allow for a certificate to be considered as valid for some subset of the resources listed in this extension, without necessarily being considered as valid for all such described resources. The implications of this approach is that any mismatch between parent and subordinate over resources where the subordinate certificate lists resources that are not contained in the parent certificate would affect validity questions relating to only those particular resources, rather than invalidating the subordinate certificate for all resources, and all of its subordinate products. This would appear to offer a relatively precise alignment to the defined problem space, and limits the scope of consequent third party damage in the event of a INR mismatch within the RPKI certification hierarchy.

Another approach may involve the alteration of the RPKI provisioning protocol [RFC6492] to include a specific signal from child to parent ("bottom up") relating to readiness for certificate revocation. At this stage it is entirely unclear how this signalling mechanism would operate, nor is it clear that it would alter the elements of operational fragility nor mitigate to any meaningful extent the risks of failure to ensure strict INR consistency at all times. This is a topic for further study.

5. An Amended RPKI Certification Validation Process

The following is a amended specification of certificate validation as described in [RFC6487] that describes an amended RPKI certificate validation process that was informally outlined in the previous section.

Validation of signed resource data using a signing key that is certified in a resource certificate, coupled with a specific set of number resources, consists of verifying that the digital signature of the signed resource data is valid, using the public key that is certified by the resource certificate, and also validating the resource certificate in the context of the RPKI, using the path validation process.

This path validation process verifies, among other things, that a prospective certification path (a sequence of n certificates) satisfies the following conditions:

1. for all ' x ' in $\{1, \dots, n-1\}$, the Subject of certificate ' x ' is the Issuer of certificate ' $x + 1$;
2. certificate ' 1 ' is issued by a trust anchor;
3. certificate ' n ' is the certificate to be validated; and
4. for all ' x ' in $\{1, \dots, n\}$, certificate ' x ' is valid for the specific set of resources.

RPKI validation for a specific set of resources entails verifying that all of the following conditions hold, in addition to the Certification Path Validation criteria specified in Section 6 of [RFC5280]:

1. The certificate can be verified using the Issuer's public key and the signature algorithm
2. The current time lies within the certificate's Validity From and To values.
3. The certificate contains all fields that MUST be present, as specified by [RFC6487], and contains values for selected fields that are defined as allowable values by this specification.

4. No field, or field value, that the [RFC6487] specification defines as MUST NOT be present is used in the certificate.
5. The Issuer has not revoked the certificate. A revoked certificate is identified by the certificate's serial number being listed on the Issuer's current CRL, as identified by the CRLDP of the certificate, the CRL is itself valid, and the public key used to verify the signature on the CRL is the same public key used to verify the certificate itself.
6. The resource extension data contained in this certificate "encompasses" the entirety of the resources in the specific resource set ("encompass" in this context is defined in Section 7.1 of [RFC6487]).
7. The Certification Path originates with a certificate issued by a trust anchor, and there exists a signing chain across the Certification Path where the Subject of Certificate 'x' in the Certification Path matches the Issuer in Certificate 'x + 1' in the Certification Path, and the public key in Certificate 'x' can verify the signature value in Certificate 'x+1'.

A certificate validation algorithm MAY perform these tests in any chosen order.

6. Security Considerations

The Security Considerations of [RFC6487] and [RFC6492] do not address the topic described here. Obviously, within the current RPKI validation procedure, any inconsistency in certificates located towards the apex of the RPKI hierarchy would invalidate the entirety of the sub-tree located below the point of this inconsistency. If the RPKI was used to control inter-domain routing in the context of a secure routing protocol, then the implications of this large scale invalidation of certificates would have a corresponding massive impact on the stability of routing. This appears to be a serious situation.

7. IANA Considerations

No updates to the registries are suggested by this document.

8. Acknowledgements

TBA.

9. References

9.1. Normative References

- [RFC3779] Lynn, C., Kent, S., and K. Seo, "X.509 Extensions for IP Addresses and AS Identifiers", RFC 3779, DOI 10.17487/RFC3779, June 2004, <<http://www.rfc-editor.org/info/rfc3779>>.
- [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>>.
- [RFC6487] Huston, G., Michaelson, G., and R. Loomans, "A Profile for X.509 PKIX Resource Certificates", RFC 6487, DOI 10.17487/RFC6487, February 2012, <<http://www.rfc-editor.org/info/rfc6487>>.

9.2. Informative References

- [RFC6482] Lepinski, M., Kent, S., and D. Kong, "A Profile for Route Origin Authorizations (ROAs)", RFC 6482, DOI 10.17487/RFC6482, February 2012, <<http://www.rfc-editor.org/info/rfc6482>>.
- [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>>.

Authors' Addresses

Geoff Huston
Asia Pacific Network Information Centre
6 Cordelia St
South Brisbane, QLD 4101
Australia

Phone: +61 7 3858 3100
Email: gih@apnic.net

George Michaelson
Asia Pacific Network Information Centre
6 Cordelia St
South Brisbane, QLD 4101
Australia

Phone: +61 7 3858 3100
Email: ggm@apnic.net

Carlos M. Martinez
Latin American and Caribbean IP Address Regional Registry
Rambla Mexico 6125
Montevideo 11400
Uruguay

Phone: +598 2604 2222
Email: carlos@lacnic.net

Tim Bruijnzeels
RIPE Network Coordination Centre
Singel 258
Amsterdam 1016 AB
The Netherlands

Email: tim@ripe.net

Andrew Lee Newton
American Registry for Internet Numbers
3635 Concorde Parkway
Chantilly, VA 20151
USA

Email: andy@arin.net

Alain Aina
African Network Information Centre (AFRINIC)
11th Floor, Raffles Tower
Cybercity, Ebene
Mauritius

Phone: +230 403 51 00
Email: aalain@afnic.net

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 5, 2016

R. Bush
IIJ Lab / Dragon Research Lab
S. Turner
IECA, Inc.
K. Patel
Cisco Systems
November 2, 2015

Router Keying for BGPsec
draft-ietf-sidr-rtr-keying-10

Abstract

BGPsec-speaking routers are provisioned with private keys in order to sign BGPsec announcements. The corresponding public keys are published in the global Resource Public Key Infrastructure, enabling verification of BGPsec messages. This document describes two methods of generating the public-private key-pairs: router-driven and operator-driven.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in RFC 2119 [RFC2119] only when they appear in all upper case. They may also appear in lower or mixed case as English words, without normative meaning.

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 May 5, 2016.

Copyright Notice

Copyright (c) 2015 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. Introduction	2
2. Management / Router Communication	3
3. Exchanging Certificates	4
4. Set-Up	4
5. PKCS#10 Generation	4
5.1. Router-Generated Keys	4
5.2. Operator-Generated Keys	5
6. Installing Signed Keys	5
7. Key Management	6
7.1. Key Validity	7
7.2. Key Roll-Over	7
7.3. Key Revocation	8
7.4. Router Replacement	8
8. Security Considerations	9
9. IANA Considerations	10
10. References	10
10.1. Normative References	10
10.2. Informative References	11
Appendix A. Management/Router Channel Security	12
Authors' Addresses	13

1. Introduction

BGPsec-speaking routers are provisioned with private keys, which allow them to digitally sign BGPsec announcements. To verify the signature, the public key, in the form of a certificate [I-D.ietf-sidr-bgpsec-pki-profiles], is published in the Resource Public Key Infrastructure (RPKI). This document describes provisioning of BGPsec-speaking routers with the appropriate public-private key-pairs. There are two sub-methods, router-driven and operator-driven.

These two sub-methods differ in where the keys are generated: on the router in the router-driven method, and elsewhere in the operator-driven method. Routers are required to support at least one of the methods in order to work in various deployment environments. Some routers may not allow the private key to be off-loaded while others may. While off-loading private keys would ease swapping of routing engines, exposure of private keys is a well known security risk.

In the operator-driven method, the operator generates the private/public key-pair and sends it to the router, perhaps in a PKCS#8 package [RFC5958].

In the router-driven method, the router generates its own public/private key-pair, uses the private key to sign a PKCS#10 certification request [I-D.ietf-sidr-bgpsec-pki-profiles], which includes the public key), and returns the certification request to the operator to be forwarded to the RPKI Certification Authority (CA). The CA returns a PKCS#7, which includes the certified public key in the form of a certificate, to the operator for loading into the router; and the CA also publishes the certificate in the RPKI.

The router-driven model mirrors the model used by traditional PKI subscribers; the private key never leaves trusted storage (e.g., Hardware Security Module). This is by design and supports classic PKI Certification Policies for (often human) subscribers which require the private key only ever be controlled by the subscriber to ensure that no one can impersonate the subscriber. For non-humans, this model does not always work. For example, when an operator wants to support hot-swappable routers the same private key needs to be installed in the soon-to-be online router that was used by the the soon-to-be offline router. This motivated the operator-driven model.

The remainder of this document describes how operators can use the two methods to provision new and existing routers.

Useful References: [I-D.ietf-sidr-bgpsec-overview] gives an overview of the BGPsec protocol, [I-D.ietf-sidr-bgpsec-protocol] gives the gritty details, [I-D.ietf-sidr-bgpsec-pki-profiles] specifies the format for the PKCS #10 request, and [I-D.ietf-sidr-bgpsec-algs] specifies the algorithms used to generate the signature.

2. Management / Router Communication

Operators are free to use either the router-driven or operator-driven method as supported by the platform. Regardless of the method chosen, operators first establish a secure communication channel between the management system and the router. How this channel is established is router-specific and is beyond scope of this document.

Though other configuration mechanisms might be used, e.g. NetConf (see [RFC6470]); for simplicity, in this document, the communication channel between the management platform and the router is assumed to be an SSH-protected CLI. See Appendix A for security considerations for this channel.

3. Exchanging Certificates

The operator management station can exchange certificate requests and certificates with routers and with the RPKI CA infrastructure using the application/pkcs10 media type [RFC5967] and application/pkcs7-mime [RFC5751], respectively, and may use FTP or HTTP per [RFC2585], or the Enrollment over Secure Transport [RFC7030].

4. Set-Up

To start, the operator uses the communication channel to install the appropriate RPKI Trust Anchor' Certificate (TA Cert) in the router. This will later enable the router to validate the router certificate returned in the PKCS#7.

The operator also configures the Autonomous System (AS) number to be used in the generated router certificate. This may be the sole AS configured on the router, or an operator choice if the router is configured with multiple ASs.

The operator configures or extracts from the router the BGP RouterID to be used in the generated certificate. In the case where the operator has chosen not to use unique per-router certificates, a RouterID of 0 may be used.

5. PKCS#10 Generation

The private key, and hence the PKCS#10 request may be generated by the router or by the operator.

5.1. Router-Generated Keys

In the router-generated method, once the protected session is established and the initial Set-Up (Section 4) performed, the operator issues a command or commands for the router to generate the public/private key pair, to generate the PKCS#10 request, and to sign the PKCS#10 with the private key. Once generated, the PKCS#10 is returned to the operator over the protected channel.

If a router was to communicate directly with a CA to have the CA certify the PKCS#10, there would be no way for the CA to authenticate

the router. As the operator knows the authenticity of the router, the operator must mediate the communication with the CA.

The operator adds the chosen AS number and the RouterID to send to the RPKI CA for the CA to certify.

5.2. Operator-Generated Keys

In the operator-generated method, the operator generates the public/private key pair on a management station and installs the private key into the router over the protected channel. Beware that experience has shown that copy and paste from a management station to a router can be unreliable for long texts.

Alternatively, the private key may be encapsulated in a PKCS #8 [RFC5958], the PKCS#8 is further encapsulated in Cryptographic Message Syntax (CMS) SignedData [RFC5652], and signed by the AS's End Entity (EE) certificate.

The router SHOULD verify the signature of the encapsulated PKCS#8 to ensure the returned private key did in fact come from the operator, but this requires that the operator also provision via the CLI or include in the SignedData the RPKI CA certificate and relevant AS's EE certificate(s). The router should inform the operator whether or not the signature validates to a trust anchor; this notification mechanism is out of scope.

The operator then creates and signs the PKCS#10 with the private key, and adds the chosen AS number and RouterID to be sent to the RPKI CA for the CA to certify.

6. Installing Signed Keys

The operator uses RPKI management tools to communicate with the global RPKI system to have the appropriate CA validate the PKCS#10 request, sign the key in the PKCS#10 and generated PKCS#7 response, as well as publishing the certificate in the Global RPKI. External network connectivity may be needed if the certificate is to be published in the Global RPKI.

After the CA certifies the key, it does two things:

1. Publishes the certificate in the Global RPKI. The CA must have connectivity to the relevant publication point, which in turn must have external network connectivity as it is part of the Global RPKI.

2. Returns the certificate to the operator's management station, packaged in a PKCS#7, using the corresponding method by which it received the certificate request. It SHOULD include the certificate chain below the TA Certificate so that the router can validate the router certificate.

In the operator-generated method, the operator SHOULD extract the certificate from the PKCS#7, and verify that the private key it holds corresponds to the returned public key.

In the operator-generated method, the operator has already installed the private key in the router (see Section 5.2).

The operator provisions the PKCS#7 into the router over the secure channel.

The router SHOULD extract the certificate from the PKCS#7 and verify that the private key corresponds to the returned public key. The router SHOULD inform the operator whether it successfully received the certificate and whether or not the keys correspond; the mechanism is out of scope.

The router SHOULD also verify that the returned certificate validates back to the installed TA Certificate, i.e., the entire chain from the installed TA Certificate through subordinate CAs to the BGPsec certificate validate. To perform this verification the CA certificate chain needs to be returned along with the router's certificate in the PKCS#7. The router SHOULD inform the operator whether or not the signature validates to a trust anchor; this notification mechanism is out of scope.

Note: The signature on the PKCS#8 and Certificate need not be made by the same entity. Signing the PKCS#8, permits more advanced configurations where the entity that generates the keys is not the direct CA.

Even if the operator cannot extract the private key from the router, this signature still provides a linkage between a private key and a router. That is the server can verify the proof of possession (POP), as required by [RFC6484].

7. Key Management

An operator's responsibilities do not end after key generation, key provisioning, certificate issuance, and certificate distribution. They persist for as long as the operator wishes to operate the BGPsec-speaking router.

7.1. Key Validity

It is critical that a BGPsec speaking router ensures that it is signing with a valid certificate at all times. To this end, the operator needs to ensure the router always has a non-expired certificate. I.e. the key used to sign BGPsec announcements always has an associated certificate whose expiry time is after the current time.

Ensuring this is not terribly difficult but requires that either:

1. The router has a mechanism to notify the operator that the certificate has an impending expiration, and/or
2. The operator notes the expiry time of the certificate and uses a calendaring program to remind them of the expiry time, and/or
3. The RPKI CA warns the operator of pending expiration, and/or
4. Use some other kind of automated process to search for and track the expiry times of router certificates.

It is advisable that expiration warnings happen well in advance of the actual expiry time.

Regardless of the technique used to track router certificate expiry times, it is advisable to notify additional operators in the same organization as the expiry time approaches thereby ensuring that the forgetfulness of one operator does not affect the entire organization.

Depending on inter-operator relationship, it may be helpful to notify a peer operator that one or more of their certificates are about to expire.

7.2. Key Roll-Over

Routers that support multiple private keys also greatly increase the chance that routers can continuously speak BGPsec because the new private key and certificate can be obtained and distributed prior to expiration of the operational key. Obviously, the router needs to know when to start using the new key. Once the new key is being used, having the already distributed certificate ensures continuous operation.

Whether the certificate is re-keyed (i.e., different key in the certificate with a new expiry time) or renewed (i.e., the same key in the certificate with a new expiry time) depends on the key's lifetime

and operational use. Arguably, re-keying the router's BGPsec certificate every time the certificate expires is more secure than renewal because it limits the private key's exposure. However, if the key is not compromised the certificate could be renewed as many times as allowed by the operator's security policy. Routers that support only one key can use renewal to ensure continuous operation, assuming the certificate is renewed and distributed well in advance of the operational certificate's expiry time.

7.3. Key Revocation

Certain unfortunate circumstances may occur causing a need to revoke a router's BGPsec certificate. When this occurs, the operator needs to use the RPKI CA system to revoke the certificate by placing the router's BGPsec certificate on the Certificate Revocation List (CRL) as well as re-keying the router's certificate.

When an active router key is to be revoked, the process of requesting the CA to revoke, the process of the CA actually revoking the router's certificate, and then the process of re-keying/renewing the router's certificate, (possibly distributing a new key and certificate to the router), and distributing the status takes time during which the operator must decide how they wish to maintain continuity of operations, with or without the compromised private key, or whether they wish to bring the router offline to address the compromise.

Keeping the router operational and BGPsec-speaking is the ideal goal, but if operational practices do not allow this then reconfiguring the router to disabling BGPsec is likely preferred to bringing the router offline.

Routers which support more than one private key, where one is operational and other(s) are soon-to-be-operational, facilitate revocation events because the operator can configure the router to make a soon-to-be-operational key operational, request revocation of the compromised key, and then make a next generation soon-to-be-operational key, all hopefully without needing to take offline or reboot the router. For routers which support only one operational key, the operators should create or install the new private key, and then request revocation of the compromised private key.

7.4. Router Replacement

Currently routers often generate private keys for uses such as SSH, and the private keys may not be seen or off-loaded from the router. While this is good security, it creates difficulties when a routing engine or whole router must be replaced in the field and all software

which accesses the router must be updated with the new keys. Also, any network based initial contact with a new routing engine requires trust in the public key presented on first contact.

To allow operators to quickly replace routers without requiring update and distribution of the corresponding public keys in the RPKI, routers SHOULD allow the private BGPsec key to be off-loaded via a protected session, e.g. SSH, NetConf (see [RFC6470]), SNMP, etc. This lets the operator upload the old private key via the mechanism used for operator-generated keys, see Section 5.2.

8. Security Considerations

The router's manual will describe whether the router supports one, the other, or both of the key generation options discussed in the earlier sections of this draft as well as other important security-related information (e.g., how to SSH to the router). After familiarizing one's self with the capabilities of the router, operators are encouraged to ensure that the router is patched with the latest software updates available from the manufacturer.

This document defines no protocols so in some sense introduces no new security considerations. However, it relies on many others and the security considerations in the referenced documents should be consulted; notably, those document listed in Section 1 should be consulted first. PKI-relying protocols, of which BGPsec is one, have many issues to consider so many in fact entire books have been written to address them; so listing all PKI-related security considerations is neither useful nor helpful; regardless, some bootstrapping-related issues are listed here that are worth repeating:

Public-Private key pair generation: Mistakes here are for all practical purposes catastrophic because PKIs rely on the pairing of a difficult to generate public-private key pair with a signer; all key pairs MUST be generated from a good source of non-deterministic random input [RFC4086].

Private key protection at rest: Mistakes here are for all practical purposes catastrophic because disclosure of the private key allows another entity to masquerade as (i.e., impersonate) the signer; all private keys MUST be protected when at rest in a secure fashion. Obviously, how each router protects private keys is implementation specific. Likewise, the local storage format for the private key is just that, a local matter.

Private key protection in transit: Mistakes here are for all practical purposes catastrophic because disclosure of the private key allows another entity to masquerade as (i.e., impersonate) the

signer; transport security is therefore strongly RECOMMENDED. The level of security provided by the transport layer's security mechanism SHOULD be commensurate with the strength of the BGPsec key; there's no point in spending time and energy to generate an excellent public-private key pair and then transmit the private key in the clear or with a known-to-be-broken algorithm, as it just undermines trust that the private key has been kept private. Additionally, operators SHOULD ensure the transport security mechanism is up to date, in order to addresses all known implementation bugs.

SSH key management is known, in some cases, to be lax [I-D.ylonen-sshkeybcp]; employees that no longer need access to routers SHOULD be removed the router to ensure only those authorized have access to a router.

Though the CA's certificate is installed on the router and used to verify that the returned certificate is in fact signed by the CA, the revocation status of the CA's certificate is rarely checked as the router may not have global connectivity or CRL-aware software. The operator MUST ensure that installed CA certificate is valid.

9. IANA Considerations

This document has no IANA Considerations.

10. References

10.1. Normative References

- [I-D.ietf-sidr-bgpsec-algs]
Turner, S., "BGP Algorithms, Key Formats, & Signature Formats", draft-ietf-sidr-bgpsec-algs-04 (work in progress), March 2013.
- [I-D.ietf-sidr-bgpsec-pki-profiles]
Reynolds, M., Turner, S., and S. Kent, "A Profile for BGPSEC Router Certificates, Certificate Revocation Lists, and Certification Requests", draft-ietf-sidr-bgpsec-pki-profiles-04 (work in progress), October 2012.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC4086] Eastlake, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, June 2005.

- [RFC4253] Ylonen, T. and C. Lonvick, "The Secure Shell (SSH) Transport Layer Protocol", RFC 4253, January 2006.
- [RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", RFC 5652, September 2009.
- [RFC5958] Turner, S., "Asymmetric Key Packages", RFC 5958, August 2010.

10.2. Informative References

- [I-D.ietf-sidr-bgpsec-overview]
Lepinski, M. and S. Turner, "An Overview of BGPSEC", draft-ietf-sidr-bgpsec-overview-02 (work in progress), May 2012.
- [I-D.ietf-sidr-bgpsec-protocol]
Lepinski, M., "BGPSEC Protocol Specification", draft-ietf-sidr-bgpsec-protocol-07 (work in progress), February 2013.
- [I-D.ylonen-sshkeybcp]
Ylonen, T. and G. Kent, "Managing SSH Keys for Automated Access - Current Recommended Practice", draft-ylonen-sshkeybcp-01 (work in progress), April 2013.
- [RFC2585] Housley, R. and P. Hoffman, "Internet X.509 Public Key Infrastructure Operational Protocols: FTP and HTTP", RFC 2585, May 1999.
- [RFC3766] Orman, H. and P. Hoffman, "Determining Strengths For Public Keys Used For Exchanging Symmetric Keys", BCP 86, RFC 3766, April 2004.
- [RFC5480] Turner, S., Brown, D., Yiu, K., Housley, R., and T. Polk, "Elliptic Curve Cryptography Subject Public Key Information", RFC 5480, March 2009.
- [RFC5647] Igoe, K. and J. Solinas, "AES Galois Counter Mode for the Secure Shell Transport Layer Protocol", RFC 5647, August 2009.
- [RFC5656] Stebila, D. and J. Green, "Elliptic Curve Algorithm Integration in the Secure Shell Transport Layer", RFC 5656, December 2009.
- [RFC5751] Ramsdell, B. and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification", RFC 5751, January 2010.

- [RFC5967] Turner, S., "The application/pkcs10 Media Type", RFC 5967, August 2010.
- [RFC6187] Igoe, K. and D. Stebila, "X.509v3 Certificates for Secure Shell Authentication", RFC 6187, March 2011.
- [RFC6470] Bierman, A., "Network Configuration Protocol (NETCONF) Base Notifications", RFC 6470, February 2012.
- [RFC6484] Kent, S., Kong, D., Seo, K., and R. Watro, "Certificate Policy (CP) for the Resource Public Key Infrastructure (RPKI)", BCP 173, RFC 6484, February 2012.
- [RFC6668] Bider, D. and M. Baushke, "SHA-2 Data Integrity Verification for the Secure Shell (SSH) Transport Layer Protocol", RFC 6668, July 2012.
- [RFC7030] Pritikin, M., Ed., Yee, P., Ed., and D. Harkins, Ed., "Enrollment over Secure Transport", RFC 7030, DOI 10.17487/RFC7030, October 2013, <<http://www.rfc-editor.org/info/rfc7030>>.

Appendix A. Management/Router Channel Security

Encryption, integrity, authentication, and key exchange algorithms used by the secure communication channel SHOULD be of equal or greater strength than the BGPsec keys they protect, which for the algorithm specified in [I-D.ietf-sidr-bgpsec-algs] is 128-bit; see [RFC5480] and by reference [SP800-57] for information about this strength claim as well as [RFC3766] for "how to determine the length of an asymmetric key as a function of a symmetric key strength requirement." In other words, for the encryption algorithm, do not use export grade crypto (40-56 bits of security), do not use Triple DES (112 bits of security). Suggested minimum algorithms would be AES-128: aes128-cbc [RFC4253] and AEAD_AES_128_GCM [RFC5647] for encryption, hmac-sha2-256 [RFC6668] or AESAD_AES_128_GCM [RFC5647] for integrity, ecdsa-sha2-nistp256 [RFC5656] for authentication, and ecdh-sha2-nistp256 [RFC5656] for key exchange.

Some routers support the use of public key certificates and SSH. The certificates used for the SSH session are different than the certificates used for BGPsec. The certificates used with SSH should also enable a level of security commensurate with BGPsec keys; x509v3-ecdsa-sha2-nistp256 [RFC6187] could be used for authentication.

Authors' Addresses

Randy Bush
IIJ / Dragon Research Labs
5147 Crystal Springs
Bainbridge Island, Washington 98110
US

Email: randy@psg.com

Sean Turner
IECA, Inc.
3057 Nutley Street, Suite 106
Fairfax, Virginia 22031
US

Email: sean@sn3rd.com

Keyur Patel
Cisco Systems
170 W. Tasman Drive
San Jose, CA 95134
USA

Email: keyupate@cisco.com

SIDR
Internet-Draft
Intended status: Informational
Expires: April 15, 2016

S. Kent
BBN
D. Ma
ZDNS
October 13, 2015

Adverse Actions by a Certification Authority (CA) or Repository Manager
in the Resource Public Key Infrastructure (RPKI)
draft-kent-sidr-adverse-actions-01

Abstract

This document analyzes actions by or against a CA or independent repository manager in the RPKI that can adversely affect the Internet Number Resources (INRs) associated with that CA or its subordinate CAs. The analysis is based on examination of the data items in the RPKI repository, as controlled by a CA (or independent repository manager) and fetched by Relying Parties (RPs). The analysis is performed from the perspective of an affected INR holder.

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 April 15, 2016.

Copyright Notice

Copyright (c) 2015 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. Introduction	2
1.1. Terminology	3
2. Analysis of RPKI Repository Objects	3
2.1. ROA	5
2.2. Manifest	7
2.3. Ghostbusters Record	9
2.4. Certificate Revocation List	10
2.5. CA Certificates	12
2.6. Router Certificates	15
3. Analysis of Actions Relative to Scenarios	16
3.1. Scenario A	17
3.2. Scenario B	18
3.3. Scenario C	18
3.4. Scenario D	19
4. Detection and Remediation	19
5. Security Considerations	21
6. IANA Considerations	21
7. Acknowledgements	21
8. References	21
8.1. Normative References	21
8.2. Informative References	23
Authors' Addresses	23

1. Introduction

Both Suspenders [I-D.kent-sidr-suspenders] and RPKI Validation Reconsidered [I-D.ietf-sidr-rpki-validation-reconsidered] address mistakes by Resource Public Key Infrastructure (RPKI) [RFC6480] Certification Authorities (CAs) (with respect to subordinate CAs). However, mistakes are not the only way that adverse changes to RPKI data can arise. A CA or repository operator might be subject to an attack [RFC7132]. For a CA, if an attack allows an adversary to use the private keys of that CA to sign RPKI objects, then the effect is analogous to the CA making mistakes. There is also the possibility that a CA or repository operator may be subject to legal measures that compel them to generate "bogus" signed objects or remove legitimate repository data. In many cases, such actions may be hard to distinguish from non-malicious mistakes, other than with respect to the time required to remedy the adverse action. Thus this document examines the implications of adverse actions with respect to Internet Number Resources (INRs) irrespective of the cause of the

actions. The document proposes mitigation strategies that take into account the nature of adverse actions, e.g., distinguishing malicious vs. erroneous actions.

This document analyzes the various types of actions by a CA (or independent repository manager) that can adversely affect the Internet Number Resources (INRs) associated with that CA, as well as the INRs of subordinate CAs. The analysis is based on examination of the data items in the RPKI repository, as controlled by a CA (or independent repository manager) and fetched by Relying Parties (RPs). The analysis is done from the perspective of an affected INR holder.

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].

2. Analysis of RPKI Repository Objects

This section enumerates the RPKI repository system objects and examines how changes to them affect Route Origination Authorizations (ROAs) and router certificate validation. Identifiers are assigned to errors for reference by later sections of this document.

The RPKI repository [RFC6481] contains a number of (digitally signed) objects that are fetched and processed by RPs. Until the deployment of BGPsec [I-D.ietf-sidr-bgpsec-overview], the principal goal of the RPKI is to enable an RP to validate ROAs [RFC6482]. A ROA binds address space to an Autonomous System Number (ASN). A ROA can be used to verify BGP announcements with respect to route origin [RFC6483]. The most important objects in the RPKI for origin validation are ROAs; all of the other RPKI objects exist to enable the validation of ROAs in a fashion consistent with the INR allocation system. Thus errors that result in changes to a ROA, or to RPKI objects needed to validate a ROA, can cause RPs to reach different (from what was intended) conclusions about the validity of the bindings expressed in a ROA.

When BGPsec is deployed, router certificates [I-D.ietf-sidr-bgpsec-pki-profiles] will be added to repository publication points. These are End-Entity (EE) certificates used to verify signatures applied to BGP update data, to enable path validation [I-D.ietf-sidr-bgpsec-protocol]. Router certificates are as important to path validation as ROAs are to origin validation.

The objects contained in the RPKI repository are of two types: conventional PKI objects (certificates and Certificate Revocation

Lists (CRLs)) and RPKI-specific signed objects. The latter make use of a common encapsulation format [RFC6488] based on the Cryptographic Message Syntax (CMS) [RFC5652]. A syntax error in this common format will cause an RP to reject the object as invalid. In turn, this may cause a ROA at a publication point to be considered invalid.

Adverse actions take several forms:

- * Deletion (D) is defined as removing an object from a publication point, without the permission of the INR holder.
- * Suppression (S) is defined as not deleting an object, or not publishing an object, as intended by an INR holder. This action also includes retaining a prior version of an object in a publication point when a newer version is available for publication.
- * Corruption (C) is defined as modification of a signed object in a fashion not requiring access to the private key used to sign the object. Thus a corrupted object will not carry a valid signature. Implicitly, the corrupted object replaces the legitimate version.
- * Modification (M) is defined as publishing a syntactically valid, verifiable version of an object that differs from the (existing) version authorized by the INR holder. Implicitly, the legitimate version of the affected object is deleted and replaced by the modified object.
- * Revocation (R) is defined as revoking a certificate (EE or CA) by placing its serial number on the appropriate CRL, without authorization of the INR holder.
- * Injection (I) is defined as introducing an instance of a signed object into a publication point (without authorization of the INR holder). It assumes that the signature on the object will be viewed as valid by RPs.

The first three of these actions (deletion, suppression, and corruption) can be effected by any entity that manages the publication point of the affected INR holder. Also, an entity with the ability to act as a man-in-the-middle between an RP and a repository can effect these actions with respect to the RP in question.

The latter three actions (modification, revocation, and injection) nominally require access to the private key of the INR holder.

All six of these actions also can be effected by a parent CA. A parent CA could reissue the INR holder's CA certificate, but with a different public key, matching a private key to which the parent CA has access. The CA could generate new signed objects using the private key associated with the reissued certificate, and publish these objects at a location of its choosing.

Most of these actions may be performed independently or in combination with one another. For example, a ROA may be revoked and deleted or revoked and replaced with a modified ROA. Where appropriate, the analysis of adverse actions will distinguish which of these individual actions, or combinations thereof, yield different outcomes for RPs. Recall that the focus of the analysis is the impact on ROAs and router certificates, with respect to RP processing.

The following sections examine how the actions enumerated above affect objects in the RPKI repository system. Each action is addressed in order (Deletion, Suppression, Corruption, Modification, Revocation, and Injection) for each object, making it easy to see how every action has been considered with regard to each object. (For the GhostBusters record we condensed the discussion of the actions because the impact is the same in each case.)

2.1. ROA

In addition to the generic RPKI object syntax checks, ROA validation requires that the signature on the ROA can be validated using the public key from the EE certificate embedded in the ROA [RFC6482]. It also requires that the EE certificate be validated consistently with the procedures described in [RFC6482] and [RFC6487]. Adverse actions against a ROA can cause the following errors:

- A-1.1 A ROA may be deleted from the indicated publication point. The result is to void the binding between the prefix(es) and the AS number in the ROA. An RP that previously viewed this binding as authentic will now not have any evidence about its validity. For origin validation, this means that a legitimate route will be treated as NotFound (if there are no other ROAs for the same prefix) or Invalid (if there is another ROA for the same prefix, but with a different AS number).

A-1.2 Suppression

- A-1.2.1 Publication of a newer ROA may be suppressed. If the INR holder intended to change the binding between the prefix(es) and the AS number in the ROA, this change will not be effected. As a result, RPs may continue to believe an old prefix/ASN binding that is no longer what the INR holder intended.
- A-1.2.2 If an INR holder intends to issue and publish two (or more) new ROAs for the same address space, one (or more) of the new ROAs may be suppressed while the other is published. In this case, RPs will de-preference the suppressed prefix/ASN binding. Suppression of the new ROA might cause traffic to flow to an ASN other than the one(s) intended by the INR holder.
- A-1.2.3 If an INR holder intends to delete all ROAs for the same address space, some of them may be retained while the others are deleted. Preventing the deletion of some ROAs can cause traffic to continue to be delivered to the ASNs that were advertised by these ROAs. Deletion of all ROAs is consistent with a transfer of address space to a different INR holder, in a phased fashion. Thus this sort of attack could interfere with the successful transfer of the affected address space (until such time as the prefixes are removed from the previous INR holder's CA certificate).
- A-1.3 A ROA may be corrupted. A corrupted ROA will be ignored by an RP, so the effect is essentially the same as for A-1.1 and 1.5. A possible difference is that an RP may be alerted to the fact that the ROA was corrupted, which might attract attention to the attack.
- A-1.4 A ROA may be modified so that the Autonomous System Number (ASN) or one or more of the address blocks in a ROA is different from the values the INR holder intended for this ROA. (This action assumes that the modified ROA's ASN and address ranges are authorized for use by the INR holder.) This attack will cause RPs to de-preference the legitimate prefix/ASN binding intended by the INR holder.

- A-1.5 A ROA may be revoked (by placing its EE certificate on the CRL for the publication point). This has the same effect as A-1.1.
- A-1.6 A ROA expressing different bindings than those published by the INR holder may be injected into a publication point. This action could authorize an additional ASN to advertise the specified prefix, allowing that ASN to originate routes for the prefix, thus enabling route origin spoofing. The injected ROA might express a different prefix for an ASN already authorized to originate a route, e.g., a longer prefix, which could enable that ASN to override other advertisements using shorter prefixes.

2.2. Manifest

Each repository publication point contains a manifest [RFC6486]. The RPKI incorporates manifests to enable RPs to detect suppression and/or substitution of (more recent) publication point objects, as the result of a mistake or attack. A manifest enumerates (by filename) all of the other signed objects at the publication point. The manifest also contains a hash of each enumerated file, to enable an RP to determine if the named file content matches what the INR holder identified in the manifest.

A manifest is an RPKI signed object, so it is validated as per [RFC6488]. If a manifest is modified in a way that causes any of these checks to fail, the manifest will be considered invalid. Suppression of a manifest itself (indicated by a stale manifest) also can cause an RP to not detect suppression of other signed objects at the publication point. (Note that if a Manifest's EE certificate expires at the time that the Manifest is scheduled to be replaced, a delay in publication will cause the Manifest to become invalid, not merely stale. This very serious outcome should be avoided, e.g., by making the Manifest EE certificate's notAfter value the same as that of the CA certificate under which it was issued). If a signed object at a publication point can be validated (using the rules applicable for that object type), then an RP MAY accept that object, even if there is no matching entry for it on the manifest. However, it appears that most RP software ignores publication point data that fails to match Manifest entries (at the time this document was written).

Corruption, suppression, modification, or deletion of a manifest might not affect RP processing of other publication point objects, as specified in [RFC6486]. However, as noted above, many RP implementations ignore objects that are present at a publication

point but not listed in a valid Manifest. Thus the following actions against a manifest can impact RP processing:

- A-2.1 A Manifest may be deleted from the indicated publication point. In this circumstance an RP may elect to use the previous Manifest that it has, and may ignore any new/changed objects at the publication point. The implications of this action are equivalent to suppression of publication of the objects that are not recognized by RPs because the new objects are not present in the old Manifest. For example, a new ROA could be ignored (A-1.2). A newly issued CA certificate might be ignored (A-5.1). A subordinate CA certificate that was revoked might still be viewed as valid by RPs (A-4.1). A new or changed router certificate might be ignored (A-6.2) as would a revised Ghostbusters record (A-3.1).
- A-2.2 Publication of a newer Manifest may be suppressed. Suppression of a newer Manifest probably will cause an RP to rely on a previous (cached) Manifest. The older Manifest would not enumerate newly added objects, and thus those objects might be ignored by an RP, equivalent to deletion of those objects (A-1.1, A-3.1, A-4.1, A-5.1, A-6.1).
- A-2.3 A Manifest may be corrupted. A corrupted Manifest will be rejected by RPs. This may cause RPs to rely on a previous manifest, with the same impact as A-2.2. If an RP does not revert to using a cached Manifest, the impact of this action is very severe, i.e., all publication point objects will be viewed as invalid, including subordinate tree objects. This is equivalent to revoking or deleting an entire subtree (see A-4.4.2).
- A-2.4 Modification
 - A-2.4.1 A Manifest may be modified to remove one or more objects. Because the modified Manifest is viewed as valid by RPs, any objects that were removed may be ignored by RPs. This is equivalent to deleting these objects from the repository. The impact of this action will vary, depending on which objects are (effectively) removed. However, the impact is equivalent to deletion of the object in question, (A-1.1, A-3.1, A-4.1, A-5.1, A-6.1).

- A-2.4.2 A Manifest may be modified to add one or more objects. If an added object has a valid signature (and is non-expired), it will be accepted by RPs and processed accordingly. If the added object was previously deleted by the INR holder, this action is equivalent to suppressing deletion of that object. If the object is newly created, or modified, it is equivalent to a modification or injection action for the type of object in question, and thus is discussed in the relevant section for those actions for the object type.
- A-2.4.3 A Manifest may be modified to list an incorrect hash for one or more objects. An object with an incorrect hash may be ignored by an RP. Thus the effect may be equivalent to corrupting the object in question, although the error reported by RP software would differ from that reported for a corrupted object. (The Manifest specifications do not require an RP to ignore an object that has a valid signature and that is not revoked or expired, but for which the hash doesn't match the object. However, an RP may elect to do so.)
- A-2.5 A Manifest may be revoked (by including its EE certificate on the CRL for the publication point). A revoked Manifest will be ignored by an RP, which probably would revert to an older (cached) Manifest. The implications for RPs are equivalent to A-2.1, with regard to new/changed objects.
- A-2.6 A Manifest representing different objects may be injected into a publication point. The effects are the same as for a modified Manifest (see above). The impact will depend on the type of the affected object(s), and thus is discussed in the relevant section(s) for each object type.

2.3. Ghostbusters Record

The Ghostbusters record [RFC6493] is a signed object that MAY be included at a publication point, at the discretion of the INR holder or publication point operator. The record is validated according to [RFC6488]. Additionally, the syntax of the record is verified based on the vCard profile from Section 5 of [RFC6493]. Errors in this record do not affect RP processing. However, if an RP encounters a problem with objects at a publication point, the RP may use information from the record to contact the publication point operator.

Adverse actions against a Ghostbusters record can cause the following error:

- A-3.1 Deletion, suppression, corruption, or revocation of a Ghostbusters record could prevent an RP from contacting the appropriate entity when a problem is detected by the RP. Modification or injection of a Ghostbusters record could cause an RP to contact the wrong entity, thus delaying remediation of a detected anomaly. All of these actions are viewed as equivalent from an RP processing perspective; they do not alter RP validation of ROAs or router certificates. However, these actions can interfere with remediation of a problem when detected by an RP.

2.4. Certificate Revocation List

Each publication point contains a CRL that enumerates revoked (not yet expired) certificates issued by the CA associated with the publication point [RFC6481].

Adverse actions against a CRL can cause the following errors:

A-4.1 Deletion

- A-4.1.1 If a CRL is deleted, RPs will continue to use an older, previously fetched Certificate Revocation List. As a result, they will not be informed of any changes in revocation status of subordinate CA or router certificates or the EE certificates of signed objects, e.g., ROAs. This action is equivalent to corruption of a CRL, since a corrupted CRL will not be accepted by an RP.
- A-4.1.2 Deletion of a CRL could cause an RP to continue to accept a ROA that no longer expresses the intent of an INR holder. As a result, an announcement for the affected prefixes would be viewed as Valid, instead of NotFound or Invalid. In this case, the effect is analogous to A-1.2.
- A-4.1.3 If a router certificate were revoked, and the CRL were deleted, RPs would not be aware of the revocation. They might continue to accept the old, revoked, router certificate. If the

certificate had been revoked due to a compromise of the router's private key, RPs would be vulnerable to accepting routes signed by an unauthorized entity.

- A-4.1.4 If a subordinate CA certificate were revoked on the deleted CRL, the revocation would not take effect. This could interfere with a transfer of address space from the subordinate CA, adversely affecting routing to the new holder of the space.
- A-4.2 If publication of the most recent CRL is suppressed, an RP will not be informed of the most recent revocation status of subordinate CA or router certificates or the EE certificates of signed objects. If an EE certificate has been revoked and the associated signed object is still present in the publication point, an RP might mistakenly treat that object as valid. (This would happen if the object is still in the manifest or the RP is configured to process valid objects that are not on the manifest.) This type of action is of special concern if the affected object is a ROA, a router certificate, or a subordinate CA certificate. The effects here are equivalent to CRL deletion (A-4.1), but suppression of a new CRL may not even be reported as an error, i.e., if the suppressed CRL were issued before the NextUpdate time (of the previous CRL).
- A-4.3 If a CRL is corrupted, an RP will reject it. If a prior CRL has not yet exceeded its NextUpdate time, an RP will continue to use the prior CRL. Even if the prior CRL has passed the NextUpdate time, an RP may choose to continue to rely on the prior CRL. The effects are essentially equivalent to suppression or deletion of a CRL (A-4.1, A-4.2)
- A-4.4 Modification
 - A-4.4.1 If a CRL is modified to erroneously list a signed object's EE certificate as revoked, the corresponding object will be treated as invalid by RPs, even if it is present in a publication point. If this object is a ROA, the (legitimate) binding expressed by the ROA will be ignored by an RP (see A-1.5). If a CRL is modified to erroneously list a router certificate as revoked, a path signature associated with that certificate will be treated as Not Valid by RPs (see A-6.5).

- A-4.4.2 If a CRL is modified to erroneously list a CA certificate as revoked, that CA and all subordinate signed objects will be treated as invalid by RPs. Depending on the location of the affected CA in the hierarchy, these effects could be very substantial, causing routes that should be Valid to be treated as NotFound.
- A-4.4.3 If a CRL is modified to omit a revoked EE, router, or CA certificate, RPs likely will continue to accept the revoked, signed object as valid. This contravenes the intent of the INR holder. If an RP continues to accept a revoked ROA, it may make routing decisions on now-invalid data. This could cause valid routes to be de-preferenced and invalid routes to continue to be accepted.
- A-4.5 A CRL cannot be revoked, per se, but it will fail validation if the CA certificate under which it was issued is revoked. See A-5.5 for a discussion of that action.
- A-4.6 Insertion of a bogus CRL can have the same effects as listed above for a modified CRL, depending on the form of the inserted CRL.

2.5. CA Certificates

Every INR holder is represented by one or more CA certificates. An INR holder has multiple CA certificates if it holds resources acquired from different sources. Also, every INR holder has more than one CA certificate during key rollover [RFC6489] and algorithm rollover [RFC6916].

If a publication point is not a leaf in the RPKI hierarchy, then the publication point will contain one or more CA certificates, each representing a subordinate CA. Each subordinate CA certificate contains a pointer (SIA) to the publication point where the signed objects associated with that CA can be found [RFC6487].

A CA certificate is a complex data structure and thus errors in that structure may have different implications for RPs depending on the specific data that is in error.

Adverse actions against a CA certificate can cause the following errors:

- A-5.1 Deletion of a CA certificate would cause an RP to not be able to locate signed objects generated by that CA, except those that have been cached by the RP. Thus an RP would be unaware of changed or new (issued after the cached data) INR bindings asserted in subordinate ROAs, and the RP would be unable to validate new or changed router certificates. If the missed objects were intended to replace ROAs or router certificates prior to expiration, then when those objects expire, RPs may cease to view them as valid. As a result, valid routes may be viewed as NotFound or Invalid.
- A-5.2 If publication of a CA certificate is suppressed, the impact depends on what changes appeared in the suppressed certificate. If the SIA value changed, the effect would be the same as in A-5.1 or 5.4.2. If the 3779 extensions in the suppressed certificate changed, the impact would be the same as in 5.4.1. If the AIA extension changed in the suppressed certificate, the impact would be the same as in 5.4.3.
- A-5.3 Corruption of a CA certificate will cause it to be rejected by RPs. In turn, this may cause subordinate signed objects to become invalid. An RP that has cached the subtree under the affected CA certificate may continue to view it as valid, until objects expire. But changed or new objects might not be retrieved, depending on details of the design of the RP software. Thus this action may be equivalent to suppressing changes to the affected subtree.
- A-5.4 Modification
- A-5.4.1 If a CA certificate is modified, but still conforms to the RPKI certificate profile [RFC6485], it will be accepted by RPs. If an [RFC3779] extension in this certificate is changed to exclude INRs that were previously present, then subordinate signed objects will become invalid if they rely on the excised INRs. If these objects are CA certificates, their subordinate signed objects will be treated as invalid. If the objects are ROAs, the binding expressed by the affected ROAs will be ignored by RPs. If the objects are router certificates, BGPsec_Path attributes [I-D.ietf-sidr-bgpsec-protocol] verifiable under these certificates will be considered invalid.

- A-5.4.2 If the SIA extension of a CA certificate is modified to refer to another publication point, this will cause an RP to look at another location for subordinate objects. This could cause RPs to not acquire the objects that the INR holder intended to be retrieved - manifests, ROAs, router certificates, Ghostbuster records, or any subordinate CA certificates associated with that CA. If the objects at this new location contain invalid signatures or appear to be corrupted, they may be rejected. In this case, cached versions of the objects may be viewed as valid by an RP, until they expire. If the objects at the new location have valid signatures and pass path validation checks, they will replace the cached objects, effectively replacing the INR holder's objects.
- A-5.4.3 If the AIA extension in a CA certificate is modified, it would point to a different CA certificate, not the parent CA certificate. This extension is used only for path discovery, not path validation. Path discovery in the RPKI is usually performed on a top-down basis, starting with TAs and recursively descending the RPKI hierarchy. Thus there may be no impact on the ability of clients to acquire and validate certificates if the AIA is modified.
- A-5.4.4 If the Subject Public Key Info (and Subject Key Identifier extension) in a CA certificate is modified to contain a public key corresponding to a private key held by the parent, the parent could sign objects as children of the affected CA certificate. With this capability, the parent could replace the INR holder, issuing new signed objects that would be accepted by RPs (as long as they do not violate the path validation criteria). This would enable the parent to effect modification, revocation, and injection actions against all of the objects under the affected CA certificate, including subordinate CA certificates.
- A-5.5 If a CA certificate is revoked an RP will treat as invalid all subordinate signed objects, both immediate and transitively. The effects are essentially the same as described in A-4.4.2.

- A-5.6 If a CA certificate is injected the impact will depend on the data contained in the injected certificate. Changes will generally be equivalent to modification actions as described in A-5.4.

2.6. Router Certificates

Router certificates are used by RPs to verify signatures on BGPsec_Path attributes carried in Update messages.

Each AS is free to determine the granularity at which router certificates are managed [I-D.ietf-sidr-bgpsec-pki-profiles]. Each participating AS is represented by one or more router certificates. During key or algorithm rollover, multiple router certificates will be present in a publication point, even if the AS is normally represented by just one such certificate.

Adverse actions against router certificates can cause the following errors:

- A-6.1 Deletion of a router certificate would cause an RP to not be able to verify signatures applied to BGPsec_Path attributes on behalf of the AS in question. In turn, this would cause the route to be treated with lower preference than competing routes that have valid BGPsec_Path attribute signatures. (However, if another router certificate for the affected AS is valid and contains the same AS number and public key, and is in use by that AS, there would be no effect on routing. This scenario will arise if a router certificate is renewed, i.e., issued with a new validity interval.)
- A-6.2 Suppression of a router certificate could have the same impact as deletion of a certificate of this type, i.e., if no router certificate was available, BGPsec attributes that should be verified using the certificate would fail validation. If an older certificate existed, and had not expired, it would be used by RPs. If the older certificate contained a different ASN, the impact would be the same as in A-6.4.
- A-6.3 Corruption of a router certificate will result in the certificate being rejected by RPs. Absent a valid router certificate, BGPsec_Path attributes associated with that certificate will be unverifiable. In turn, this would cause the route to be treated with lower preference than

competing routes that have valid BGPsec_Path attribute signatures.

- A-6.4 If a router certificate is modified to represent a different ASN, but it still passes syntax checks, then this action could cause signatures on BGPsec_Path attributes to be associated with the wrong AS. This could cause signed routes to be inconsistent with the intent of the INR holder, e.g., traffic might be routed via a different AS than intended.
- A-6.5 If a router certificate were revoked, BGPsec_Path attributes verifiable using that certificate would no longer be considered valid. The impact would be the same as for a deleted certificate, as described in A-6.1
- A-6.6 Insertion of a router certificate could authorize additional routers to sign BGPsec traffic for the targeted ASN, and thus undermine fundamental BGPsec security guarantees.

3. Analysis of Actions Relative to Scenarios

This section examines the types of problems that can arise in four scenarios described below. We consider mistakes, (successful) attacks against a CA or a publication point, and situations in which a CA or publication point manager is compelled to take action by a law enforcement authority.

We explore the following four scenarios:

- A. An INR holder operates its own CA and manages its own repository publication point.
- B. An INR holder operates its own CA, but outsources management of its repository publication point to its parent or another entity.
- C. An INR holder outsources management of its CA to its parent, but manages its own repository publication point.
- D. An INR holder outsources management of its CA and its publication point to its parent.

Note that these scenarios focus on the affected INR holder as the party directly affected by an adverse action. The most serious cases

arise when the INR holder appears as a high-tier CA in the RPKI hierarchy; in such situations subordinate INR holders may be affected as a result of an action. A mistake by or an attack against a "leaf" has more limited impact because all of the affected INRs belong to the INR holder itself.

In Scenario A, actions by the INR holder can adversely affect all of its resources and, transitively, resources of any subordinate CAs. (If the CA is a "leaf" in the RPKI, then it has no subordinate CAs and the damage is limited to its own INRs.)

In Scenario B, actions by the (outsourced) repository operator also can adversely affect the resources of the INR holder, and those of any subordinates CAs. (If the CA is a "leaf" in the RPKI, then it has no subordinate CAs and the damage is limited, as in Scenario A.) The range of adverse effects here includes those in Scenario A, and adds a new potential source of adverse actions, i.e., the outsourced repository operator.

In Scenario C, all signed objects associated with the INR holder are generated by the parent CA but are self-hosted. (We expect this scenario to be rare, because an INR holder that elects to outsource CA operation seems unlikely to manage its own repository publication point.) Because that CA has the private key used to sign them, it can generate alternative signed objects---ones not authorized by the INR holder. However, erroneous objects created by the parent CA will not be published by the INR holder IF the holder checks them first. Because the parent CA is acting on behalf of the INR holder, mistakes by or attacks against that entity are equivalent to ones effected by the INR holder in Scenario A.

The INR holder is most vulnerable in Scenario D. Actions by the parent CA, acting on behalf of the INR holder, can adversely affect all signed objects associated with that INR holder, including any subordinate CA certificates. These actions will presumably translate directly into publication point changes, because the parent CA is managing the publication point for the INR holder. The range of adverse effects here includes those in Scenarios A, B, and C.

3.1. Scenario A

In this scenario, the INR holder acts as its own CA and it manages its own publication point. Actions by the INR holder can adversely affect all of its resources and, transitively, resources of any subordinate CAs. (If the CA is a "leaf" in the RPKI, then it has no subordinate CAs and the damage is limited to its own INRs.) Mistakes by the INR holder can cause any of the actions noted in Section 2. A successful attack against this CA can effect all of the modification,

revocation, or injection actions noted in that section. (We assume that objects generated by the CA are automatically published). An attack against the publication point can effect all of the deletion, suppression, or corruption actions noted in that section.

3.2. Scenario B

In this scenario, the INR holder acts as its own CA and but it delegates management of its own publication point to a third party. Mistakes by the INR holder can cause any of the modification, revocation, or injection actions described in Section 2. Actions by the repository operator can adversely affect the resources of the INR holder, and those of any subordinate CAs. (If the CA is a "leaf" in the RPKI, then it has no subordinate CAs and the damage is limited, as in Scenario A.) The range of adverse effects here includes those in Scenario A, and adds a new potential source of adverse actions, i.e., the third party repository operator. A successful attack against the CA can effect all of the modification, revocation, or injection actions noted in that section (assuming that objects generated by the CA are automatically published). Here, actions by the publication point manager (or attacks against that entity) can effect all of the deletion, suppression, or corruption actions noted in Section 2.

3.3. Scenario C

In this scenario, the INR holder outsources management of its CA to its parent, but manages its own repository publication point. All signed objects associated with the INR holder are generated by the parent CA but are self-hosted. (We expect this scenario to be rare, because an INR holder that elects to outsource CA operation seems unlikely to manage its own repository publication point.) Because that CA has the private key used to sign them, it can generate alternative signed objects -- ones not authorized by the INR holder. However, erroneous objects created by the parent CA will not be published by the INR holder IF the holder checks them first. Because the parent CA is acting on behalf of the INR holder, mistakes by or attacks against that entity are equivalent to ones effected by the INR holder in Scenario A. Mistakes by the INR holder, acted upon by the parent CA, can cause any of the actions noted in Section 2. Actions unilaterally undertaken by the parent CA also can have the same effect, unless the INR holder checks the signed objects before publishing them. A successful attack against the parent CA can effect all of the modification, revocation, or injection actions noted in Section 2, unless the INR holder checks the signed objects before publishing them. An attack against the INR holder (in its role as repository operator) can effect all of the deletion, suppression, or corruption actions noted in Section 2 (because the

INR holder is managing its publication point), unless the INR holder checks the signed objects before publishing them. (An attack against the INR holder implies that the path it uses to direct the parent CA to issue and publish objects has been compromised.)

3.4. Scenario D

In this scenario an INR holder outsources management of both its CA and its publication point to its parent. The INR holder is most vulnerable in this scenario. Actions by the parent CA, acting on behalf of the INR holder, can adversely affect all signed objects associated with that INR holder, including any subordinate CA certificates. These actions will presumably translate directly into publication point changes, because the parent CA is managing the publication point for the INR holder. The range of adverse effects here includes those in Scenarios A, B, and C. Mistakes by the INR holder, acted upon by the parent CA, can cause any of the actions noted in Section 2. Actions unilaterally undertaken by the parent CA also can have the same effect. A successful attack against the parent CA can effect all of the modification, revocation, or injection actions noted in Section 2. An attack against the parent CA can also effect all of the deletion, suppression, or corruption actions noted in Section 2 (because the parent CA is managing the INR holder's publication point).

4. Detection and Remediation

To detect problems, each INR holder SHOULD check the signed objects available in its publication point on a regular basis. This document RECOMMENDS that each INR holder perform such checks as a side-effect of acquiring RPKI data for local processing, e.g., 3-4 times a day. Third parties also can perform checks on behalf of RPs, to detect adverse actions. This is consistent with the outsourcing options noted in the use cases in Section 1. In either situation, detection of adverse actions requires that the cognizant party have available a reference set of RPKI data for the INR holder. The reference set includes all signed objects in the publication point(s) of the INR holder plus the CA certificate for each publication point.

If an adverse action is the result of a mistake by, or an attack against, a superior CA or a repository manager, the INR holder SHOULD contact the relevant entity as soon as possible. It is expected that a mistake by these entities normally will be corrected and new objects published within 24-72 hours. An attack against a superior CA or repository manager also should be remedied by the affected parties, but the time to repair the damage may be longer, because of the additional activities that may accompany responding to an attack,

e.g., assessing the extent of the attack, identifying and remediating the vulnerabilities that allowed the attack, etc.

In both of these situations, the harmful effects of adverse actions can be mitigated if RPs delay acting on changes that might be attributable to adverse actions. This observation motivates the introduction of hysteresis into the RPKI validation process, at least with respect to changes that are indicative of an adverse action. The idea is that an RP would continue to accept as valid objects that were previously valid (based on local cache history), and ignore recent changes, for some interval.

Sometimes an INR holder may wish to make a change that might be viewed as an adverse action, without encountering such a delay. To accommodate this situation, the RPKI can be extended to allow an INR holder to provide independent confirmation of such changes. A mechanism of this sort could prevent mistakes by a superior CA or repository manager from having an immediate, adverse effect. If the independent confirmation is not completely dependent on the RPKI repository and CA system, it can be immune to mistakes by or attacks against superior CAs and repository managers, and mistakes by or attacks against third-party CAs and repository managers.

The effects of an attack on an INR holder itself may not be countered by a mechanism of this sort; an adversary who can attack a CA and/or repository management function of an INR holder may be able to attack the independent confirmation mechanism at the same time. The use of a separate mechanism does create the potential for additional safeguards against such attacks. However, the extent to which this potential is achieved will depend on how an INR holder implements and manages this mechanism.

If a superior CA or repository manager is compelled to engage in an adverse action against an INR holder, e.g., by a law enforcement agency, use of an independent confirmation mechanism also may be able to counter such an action. In this situation, the actions are not likely to be reversed quickly, unlike a mistake or attack. This situation might argue for a longer period of delay. An INR holder and a subordinate INR holder may disagree about an action that invalidates the holdings of the subordinate. The addition of hysteresis and an independent confirmation mechanism to the RPKI ought not to deprive an INR holder of the legitimate ability to take such actions. This argues for a time limit on the hysteresis and confirmation mechanism, consistent with the business practices of RPKI CAs. This time limit should be long enough to allow affected entities to remedy mistakes and recover from attacks. It also should be short enough to not impede the resolution of legitimate actions by an INR holder relative to subordinate INR holders.

5. Security Considerations

This informational document describes a threat model for the RPKI, focusing on mistakes by or attacks against CAs and independent repository managers. It is intended to support the design of future RPKI security mechanisms that seek to address the concerns associated with such actions.

6. IANA Considerations

This document has no actions for IANA.

7. Acknowledgements

The authors thank Richard Hansen and David Mandelberg for their review feedback and editorial assistance.

8. References

8.1. Normative References

- [I-D.ietf-sidr-bgpsec-overview]
Lepinski, M., "An Overview of BGPsec", draft-ietf-sidr-bgpsec-overview-07 (work in progress), June 2015.
- [I-D.ietf-sidr-bgpsec-pki-profiles]
Reynolds, M. and S. Kent, "A Profile for BGPsec Router Certificates, Certificate Revocation Lists, and Certification Requests", draft-ietf-sidr-bgpsec-pki-profiles-11 (work in progress), August 2015.
- [I-D.ietf-sidr-bgpsec-protocol]
Lepinski, M., "BGPsec Protocol Specification", draft-ietf-sidr-bgpsec-protocol-13 (work in progress), July 2015.
- [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>>.
- [RFC3779] Lynn, C., Kent, S., and K. Seo, "X.509 Extensions for IP Addresses and AS Identifiers", RFC 3779, DOI 10.17487/RFC3779, June 2004, <<http://www.rfc-editor.org/info/rfc3779>>.
- [RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <<http://www.rfc-editor.org/info/rfc5652>>.

- [RFC6480] Lepinski, M. and S. Kent, "An Infrastructure to Support Secure Internet Routing", RFC 6480, DOI 10.17487/RFC6480, February 2012, <<http://www.rfc-editor.org/info/rfc6480>>.
- [RFC6481] Huston, G., Loomans, R., and G. Michaelson, "A Profile for Resource Certificate Repository Structure", RFC 6481, DOI 10.17487/RFC6481, February 2012, <<http://www.rfc-editor.org/info/rfc6481>>.
- [RFC6482] Lepinski, M., Kent, S., and D. Kong, "A Profile for Route Origin Authorizations (ROAs)", RFC 6482, DOI 10.17487/RFC6482, February 2012, <<http://www.rfc-editor.org/info/rfc6482>>.
- [RFC6483] Huston, G. and G. Michaelson, "Validation of Route Origination Using the Resource Certificate Public Key Infrastructure (PKI) and Route Origin Authorizations (ROAs)", RFC 6483, DOI 10.17487/RFC6483, February 2012, <<http://www.rfc-editor.org/info/rfc6483>>.
- [RFC6485] Huston, G., "The Profile for Algorithms and Key Sizes for Use in the Resource Public Key Infrastructure (RPKI)", RFC 6485, DOI 10.17487/RFC6485, February 2012, <<http://www.rfc-editor.org/info/rfc6485>>.
- [RFC6486] Austein, R., Huston, G., Kent, S., and M. Lepinski, "Manifests for the Resource Public Key Infrastructure (RPKI)", RFC 6486, DOI 10.17487/RFC6486, February 2012, <<http://www.rfc-editor.org/info/rfc6486>>.
- [RFC6487] Huston, G., Michaelson, G., and R. Loomans, "A Profile for X.509 PKIX Resource Certificates", RFC 6487, DOI 10.17487/RFC6487, February 2012, <<http://www.rfc-editor.org/info/rfc6487>>.
- [RFC6488] Lepinski, M., Chi, A., and S. Kent, "Signed Object Template for the Resource Public Key Infrastructure (RPKI)", RFC 6488, DOI 10.17487/RFC6488, February 2012, <<http://www.rfc-editor.org/info/rfc6488>>.
- [RFC6489] Huston, G., Michaelson, G., and S. Kent, "Certification Authority (CA) Key Rollover in the Resource Public Key Infrastructure (RPKI)", BCP 174, RFC 6489, DOI 10.17487/RFC6489, February 2012, <<http://www.rfc-editor.org/info/rfc6489>>.

- [RFC6493] Bush, R., "The Resource Public Key Infrastructure (RPKI) Ghostbusters Record", RFC 6493, DOI 10.17487/RFC6493, February 2012, <<http://www.rfc-editor.org/info/rfc6493>>.
- [RFC6916] Gagliano, R., Kent, S., and S. Turner, "Algorithm Agility Procedure for the Resource Public Key Infrastructure (RPKI)", BCP 182, RFC 6916, DOI 10.17487/RFC6916, April 2013, <<http://www.rfc-editor.org/info/rfc6916>>.
- [RFC7132] Kent, S. and A. Chi, "Threat Model for BGP Path Security", RFC 7132, DOI 10.17487/RFC7132, February 2014, <<http://www.rfc-editor.org/info/rfc7132>>.

8.2. Informative References

- [I-D.ietf-sidr-rpki-validation-reconsidered]
Huston, G., Michaelson, G., Martinez, C., Bruijnzeels, T., Newton, A., and A. Aina, "RPKI Validation Reconsidered", draft-ietf-sidr-rpki-validation-reconsidered-02 (work in progress), October 2015.
- [I-D.kent-sidr-suspenders]
Kent, S. and D. Mandelberg, "Suspenders: A Fail-safe Mechanism for the RPKI", draft-kent-sidr-suspenders-03 (work in progress), April 2015.

Authors' Addresses

Stephen Kent
BBN Technologies
10 Moulton St
Cambridge, MA 02138-1119
USA

EMail: kent@bbn.com

Di Ma
ZDNS
4 South 4th St.
Zhongguancun
Haidian, Beijing 100190
China

EMail: madi@zdns.cn

SIDR
Internet-Draft
Intended status: Informational
Expires: April 21, 2016

T. Bruijnzeels
O. Muravskiy
RIPE NCC
October 19, 2015

RPKI Repository Validation Using Local Cache
draft-tbruijnzeels-sidr-validation-local-cache-02

Abstract

This document describes the approach to validate the content of the RPKI repository, which is independent of a particular object retrieval mechanism. This allows it to be used with repositories available over rsync protocol (see Section 3 of[RFC6481]), and delta protocol ([I-D.tbruijnzeels-sidr-delta-protocol]), as well as repositories that use a mix of both.

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 April 21, 2016.

Copyright Notice

Copyright (c) 2015 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. Introduction	2
2. Top-down Validation of a Single Repository	2
2.1. Fetching Trust Anchor Certificate Using Trust Anchor Locator	3
2.2. Resource Certificate Validation	3
2.2.1. Finding most recent valid manifest and CRL	4
2.2.2. Manifest entries validation	5
2.3. Store Cleanup	5
3. Remote Objects Fetcher	5
3.1. Fetcher Operations	5
3.1.1. Fetch repository objects	5
3.1.2. Fetch single repository object	6
4. Local Object Store	7
4.1. Store Operations	7
4.1.1. Store Repository Object	7
4.1.2. Update object's last fetch time	7
4.1.3. Get objects by hash	7
4.1.4. Get certificate objects by URI	7
4.1.5. Get manifest objects by AKI	7
4.1.6. Delete objects for URI	7
4.1.7. Delete outdated objects	7
4.1.8. Update object's validation time	7
5. Acknowledgements	8
6. IANA Considerations	8
7. Security Considerations	8
8. References	8
8.1. Normative References	8
8.2. Informative References	9
Authors' Addresses	9

1. Introduction

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].

2. Top-down Validation of a Single Repository

The validation of one repository is independent from any other repository, and thus, multiple repositories could be validated concurrently.

The validation of a repository starts from its Trust Anchor (TA) certificate. To retrieve the TA, the Trust Anchor Locator (TAL) object is used, as described in Section 2.1.

If the TA certificate is retrieved, it is validated according to the Section 2.2 of [RFC6490].

Then the TA certificate is validated as a resource certificate, as described in Section 2.2.

For all repository objects that were validated during this validation run, their validation timestamp is updated in the local store (see Section 4.1.8).

Outdated objects are removed from the store as described in Section 2.3. This completes the validation of a repository.

2.1. Fetching Trust Anchor Certificate Using Trust Anchor Locator

The following steps are performed in order to fetch the Trust Anchor Certificate:

- o If the Trust Anchor Locator contains "prefetch.uris" field, pass the URIs contained there to the fetcher (see Section 3.1.1).
- o Pass to the fetcher (Section 3.1.2) the URI from the TAL (see Section 2.1 of [RFC6490]).
- o Retrieve from the local store (see Section 4.1.4) all certificate objects, for which the URI matches the URI extracted from the TAL in the previous step, and the public key matches the subjectPublicKeyInfo field of the TAL (Section 2.1 of [RFC6490]).
- o If no, or more than one such objects are found, issue an error and stop validation process. Otherwise, use that object as a Trust Anchor certificate.

2.2. Resource Certificate Validation

The following steps describe the validation of a single resource certificate:

- o If both the caRepository (Section 4.8.8.1 of [RFC6487]), and the id-ad-rpkiNotify (Section 3.5 of [I-D.tbuijnzeels-sidr-delta-protocol]) SIA pointers are present in the given resource certificate, use a local policy to determine which pointer to use. Extract the URI from the selected pointer and pass it to the fetcher (see Section 3.1.1).

- o For a given resource certificate, find it's manifest and certificate revocation list (CRL), using the procedure described in Section 2.2.1. If no such manifest and CRL could be found, issue an error and stop processing current certificate.
- o Compare given resource certificate's manifest URI with the URI of the manifest found in the previous step. If they are different, issue a warning.
- o Get from the local store and validate repository objects that correspond to the manifest entries, using the procedure described in the Section 2.2.2.
- o Validate all resource certificate objects found on the manifest, using the CRL object found on the manifest, according to Section 7 of [RFC6487].
- o Validate all ROA objects found on the manifest, using the CRL object found on the manifest, according to the Section 4 of [RFC6482].
- o Validate all Ghostbusters Record objects found on the manifest, using the CRL object found on the manifest, according to the Section 7 of [RFC6493].
- o For every valid resource certificate object found on the manifest, apply the procedure described in this section (Section 2.2), recursively, provided that this resource certificate (identified by it's SKI) has not yet been validated during current repository validation run.

2.2.1. Finding most recent valid manifest and CRL

Fetch from the store (see Section 4.1.5) all objects of type manifest, whose certificate's AKI field matches the SKI of the current CA certificate.

Find the manifest object with the highest manifest number, for which all following conditions are met:

- o There is only one entry in the manifest for which the store contains exactly one object of type CRL, whose hash matches the hash of the entry.
- o The manifest's certificate AKI equals the above CRL's AKI
- o The above CRL is a valid object according to Section 6.3 of [RFC5280]

- o The manifest is a valid object according to Section 4.4 of [RFC6486], using the CRL found above

Report an error for every invalid manifest with the number higher than the number of the valid manifest.

2.2.2. Manifest entries validation

For every entry in the manifest object:

- o Construct an entry's URI by appending the entry name to the current CA's publication point URI.
- o Get all objects from the store whose hash attribute equals entry's hash (see Section 4.1.3).
- o If no such objects found, issue an error.
- o For every found object, compare it's URI with the URI of the manifest entry. If they do not match, issue a warning.
- o If no objects with matching URI found, issue a warning.
- o If some objects with non-matching URI found, issue a warning.

2.3. Store Cleanup

At the end of repository validation, the store cleanup is performed. Given all objects that were validated during current validation run, it removes from the store (Section 4.1.7) all objects whose URI attribute matches URI of validated object(s), but the hash attribute is different.

3. Remote Objects Fetcher

The fetcher is responsible for downloading objects from remote repositories. Currently rsync and RRDP repositories are supported.

3.1. Fetcher Operations

3.1.1. Fetch repository objects

This operation receives one parameter - a URI. For rsync protocol this URI points to a directory in a remote rsync repository. For RRDP repository it points to the repository's notification file.

The fetcher performs following steps:

- o If the given URI has been downloaded recently (as specified by the local policy), do nothing.
- o Download remote objects using the URI provided (for rsync repository use recursive mode).
- o For every new object that is downloaded, try to parse it as an object of specific RPKI type (certificate, manifest, CRL, ROA, Ghostbusters record), based on the object's filename extension (.cer, .mft, .crl, .roa, and .gbr, respectively), and perform basic RPKI object validation, as specified in [RFC6487] and [RFC6488].
- o For every downloaded valid object, record it in the local store (Section 4.1.1), and set it's last fetch time to the time it was downloaded (Section 4.1.2).

3.1.2. Fetch single repository object

This operation receives one parameter - a URI that points to an object in a remote repository.

The fetcher performs following operations:

- o If the given URI has been downloaded recently (as specified by the local policy), do nothing.
- o Download the remote object using the URI provided.
- o Try to parse downloaded object as an object of a specific RPKI type (certificate, manifest, CRL, ROA, Ghostbusters record), based on the object's filename extension (.cer, .mft, .crl, .roa, and .gbr, respectively), and perform basic RPKI object validation, as specified in [RFC6487] and [RFC6488].
- o If the downloaded object is not valid, issue an error and skip further steps.
- o Delete objects from the local store (Section 4.1.6) using given URI.
- o Put validated object in the local store (Section 4.1.1), and set it's last fetch time to the time it was downloaded (Section 4.1.2).

4. Local Object Store

4.1. Store Operations

4.1.1. Store Repository Object

Put given object in the store, along with it's type, URI, hash, and AKI, if there is no record with the same hash and URI fields.

4.1.2. Update object's last fetch time

For all objects in the store whose URI matches the given URI, set the last fetch time attribute to the given timestamp.

4.1.3. Get objects by hash

Retrieve all objects from the store whose hash attribute matches the given hash.

4.1.4. Get certificate objects by URI

Retrieve from the store all objects of type certificate, whose URI attribute matches the given URI.

4.1.5. Get manifest objects by AKI

Retrieve from the store all objects of type manifest, whose AKI attribute matches the given AKI.

4.1.6. Delete objects for URI

For a given URI, delete all objects in the store with matching URI attribute.

4.1.7. Delete outdated objects

For a given URI and a list of hashes, delete all objects in the store with matching URI, whose hash attribute is not in the given list of hashes.

4.1.8. Update object's validation time

For all objects in the store whose hash attribute matches the given hash, set the last validation time attribute to the given timestamp.

5. Acknowledgements

6. IANA Considerations

7. Security Considerations

8. References

8.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [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>>.
- [RFC6481] Huston, G., Loomans, R., and G. Michaelson, "A Profile for Resource Certificate Repository Structure", RFC 6481, DOI 10.17487/RFC6481, February 2012, <<http://www.rfc-editor.org/info/rfc6481>>.
- [RFC6482] Lepinski, M., Kent, S., and D. Kong, "A Profile for Route Origin Authorizations (ROAs)", RFC 6482, DOI 10.17487/RFC6482, February 2012, <<http://www.rfc-editor.org/info/rfc6482>>.
- [RFC6486] Austein, R., Huston, G., Kent, S., and M. Lepinski, "Manifests for the Resource Public Key Infrastructure (RPKI)", RFC 6486, DOI 10.17487/RFC6486, February 2012, <<http://www.rfc-editor.org/info/rfc6486>>.
- [RFC6487] Huston, G., Michaelson, G., and R. Loomans, "A Profile for X.509 PKIX Resource Certificates", RFC 6487, DOI 10.17487/RFC6487, February 2012, <<http://www.rfc-editor.org/info/rfc6487>>.
- [RFC6488] Lepinski, M., Chi, A., and S. Kent, "Signed Object Template for the Resource Public Key Infrastructure (RPKI)", RFC 6488, DOI 10.17487/RFC6488, February 2012, <<http://www.rfc-editor.org/info/rfc6488>>.

- [RFC6490] Huston, G., Weiler, S., Michaelson, G., and S. Kent,
 "Resource Public Key Infrastructure (RPKI) Trust Anchor
 Locator", RFC 6490, DOI 10.17487/RFC6490, February 2012,
 <<http://www.rfc-editor.org/info/rfc6490>>.
- [RFC6493] Bush, R., "The Resource Public Key Infrastructure (RPKI)
 Ghostbusters Record", RFC 6493, DOI 10.17487/RFC6493,
 February 2012, <<http://www.rfc-editor.org/info/rfc6493>>.

8.2. Informative References

- [I-D.tbuijnzeels-sidr-delta-protocol]
 Bruijnzeels, T., Muravskiy, O., Weber, B., Austein, R.,
 and D. Mandelberg, "RPKI Repository Delta Protocol",
 draft-tbuijnzeels-sidr-delta-protocol-03 (work in
 progress), December 2014.

Authors' Addresses

Tim Bruijnzeels
RIPE NCC

Email: tim@ripe.net

Oleg Muravskiy
RIPE NCC

Email: oleg@ripe.net