

SIDROPS
Internet-Draft
Obsoletes: 6486 (if approved)
Intended status: Standards Track
Expires: 25 September 2022

R. Austein
Arrcus, Inc.
G. Huston
APNIC
S. Kent
Independent
M. Lepinski
New College Florida
24 March 2022

Manifests for the Resource Public Key Infrastructure (RPKI)
draft-ietf-sidrops-6486bis-11

Abstract

This document defines a "manifest" for use in the Resource Public Key Infrastructure (RPKI). A manifest is a signed object (file) that contains a listing of all the signed objects (files) in the repository publication point (directory) associated with an authority responsible for publishing in the repository. For each certificate, Certificate Revocation List (CRL), or other type of signed objects issued by the authority that are published at this repository publication point, the manifest contains both the name of the file containing the object and a hash of the file content. Manifests are intended to enable a relying party (RP) to detect certain forms of attacks against a repository. Specifically, if an RP checks a manifest's contents against the signed objects retrieved from a repository publication point, then the RP can detect replay attacks, and unauthorized in-flight modification or deletion of signed objects. This document obsoletes RFC 6486.

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 <https://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 25 September 2022.

Copyright Notice

Copyright (c) 2022 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 (<https://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 Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction	3
1.1. Requirements Language	3
2. Manifest Scope	4
3. Manifest Signing	4
4. Manifest Definition	4
4.1. eContentType	4
4.2. eContent	5
4.2.1. Manifest	5
4.2.2. Names in FileAndHash objects	7
4.3. Content-Type Attribute	7
4.4. Manifest Validation	7
5. Manifest Generation	8
5.1. Manifest Generation Procedure	8
5.2. Considerations for Manifest Generation	9
6. Relying Party Processing of Manifests	10
6.1. Manifest Processing Overview	11
6.2. Acquiring a Manifest for a CA	11
6.3. Detecting Stale and or Prematurely-issued Manifests	12
6.4. Acquiring Files Referenced by a Manifest	12
6.5. Matching File Names and Hashes	12
6.6. Failed Fetches	12
7. Publication Repositories	13
8. Security Considerations	13
9. IANA Considerations	13
10. Acknowledgements	14
11. References	14
11.1. Normative References	14
11.2. Informative References	15
Appendix A. ASN.1 Module	15
Appendix B. Changes since RFC 6486	16
Authors' Addresses	17

1. Introduction

The Resource Public Key Infrastructure (RPKI) [RFC6480] makes use of a distributed repository system [RFC6481] to make available a variety of objects needed by relying parties (RPs). Because all of the objects stored in the repository system are digitally signed by the entities that created them, attacks that modify these published objects are detectable by RPs. However, digital signatures alone provide no protection against attacks that substitute "stale" versions of signed objects (i.e., objects that were valid and have not yet expired, but have since been superseded), or in-flight attacks that remove an object that should be present in the repository. To assist in the detection of such attacks, RPKI repository systems make use of a signed object called a "manifest".

A manifest is a signed object that enumerates all the signed objects (files) in the repository publication point (directory) that are associated with an authority responsible for publishing at that publication point. Each manifest contains both the name of the file containing the object and a hash of the file content, for every signed object issued by an authority that is published at the authority's repository publication point. A manifest is intended to allow an RP to detect unauthorized object removal or the substitution of stale versions of objects at a publication point. A manifest also is intended to allow an RP to detect similar outcomes that may result from an on-path attack during the retrieval of objects from the repository. Manifests are intended to be used in Certification Authority (CA) publication points in repositories (directories containing files that are subordinate certificates and Certificate Revocation Lists (CRLs) issued by this CA and other signed objects that are verified by End-Entity (EE) certificates issued by this CA).

Manifests are modeled on CRLs, as the issues involved in detecting stale manifests and potential attacks using manifest replays, etc., are similar to those for CRLs. The syntax of the manifest payload differs from CRLs, since RPKI repositories contain objects not covered by CRLs, e.g., digitally signed objects, such as Route Origination Authorizations (ROAs) [RFC6482].

This document obsoletes [RFC6486].

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Manifest Scope

A manifest associated with a CA's repository publication point contains a list of:

- * the set of (non-expired, non-revoked) certificates issued and published by this CA,
- * the most recent CRL issued by this CA, and
- * all published signed objects that are verifiable using EE certificates [RFC6487] issued by this CA (other than the manifest itself).

Every RPKI signed object includes, in the Cryptographic Message Syntax (CMS) [RFC5652] wrapper of the object, the EE certificate used to verify it [RFC6488]. Thus, there is no requirement to separately publish that EE certificate at the CA's repository publication point.

Where multiple CA instances share a common publication point, as can occur when a CA performs a key-rollover operation [RFC6489], the repository publication point will contain multiple manifests. In this case, each manifest describes only the collection of published products of its associated CA instance.

3. Manifest Signing

A CA's manifest is verified using an EE certificate. The SubjectInfoAccess (SIA) field of this EE certificate contains the access method Object Identifier (OID) of id-ad-signedObject.

The CA MUST sign only one manifest with each generated private key, and MUST generate a new key pair for each new version of the manifest. This form of use of the associated EE certificate is termed a "one-time-use" EE certificate [RFC6487].

4. Manifest Definition

A manifest is an RPKI signed object, as specified in [RFC6488]. The RPKI signed object template requires specification of the following data elements in the context of the manifest structure.

4.1. eContentType

The eContentType for a manifest is defined as id-ct-rpkiManifest and has the numerical object identifier of 1.2.840.113549.1.9.16.1.26.

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

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

```
id-ct-rpkiManifest OBJECT IDENTIFIER ::= { id-ct 26 }
```

4.2. eContent

The content of a manifest is ASN.1 encoded using the Distinguished Encoding Rules (DER) [X.690]. The content of a manifest is defined as follows:

```
Manifest ::= SEQUENCE {
    version          [0] INTEGER DEFAULT 0,
    manifestNumber   INTEGER (0..MAX),
    thisUpdate       GeneralizedTime,
    nextUpdate       GeneralizedTime,
    fileHashAlg      OBJECT IDENTIFIER,
    fileList         SEQUENCE SIZE (0..MAX) OF FileAndHash
}
```

```
FileAndHash ::= SEQUENCE {
    file            IA5String,
    hash            BIT STRING
}
```

4.2.1. Manifest

The manifestNumber, thisUpdate, and nextUpdate fields are modeled after the corresponding fields in X.509 CRLs (see [RFC5280]). Analogous to CRLs, a manifest is nominally current until the time specified in nextUpdate or until a manifest is issued with a greater manifest number, whichever comes first.

Because a "one-time-use" EE certificate is employed to verify a manifest, the EE certificate MUST be issued with a validity period that coincides with the interval from thisUpdate to nextUpdate in the manifest, to prevent needless growth of the CA's CRL.

The data elements of the manifest structure are defined as follows:

version:

The version number of this version of the manifest specification MUST be 0.

manifestNumber:

This field is an integer that is incremented (by 1) each time a new manifest is issued for a given publication point. This field allows an RP to detect gaps in a sequence of published manifests.

As the manifest is modeled on the CRL specification, the ManifestNumber is analogous to the CRLNumber, and the guidance in [RFC5280] for CRLNumber values is appropriate as to the range of number values that can be used for the manifestNumber. Manifest numbers can be expected to contain long integers. Manifest verifiers MUST be able to process number values up to 20 octets. Conforming manifest issuers MUST NOT use number values longer than 20 octets. The issuer MUST increase the value of this field monotonically for each newly-generated Manifest. Each RP MUST verify that a purported "new" Manifest contains a higher manifestNumber than previously-validated Manifests. If the purported "new" Manifest contains an equal or lower manifestNumber than previously-validated Manifests, the RP SHOULD use locally cached versions of objects, as described in Section 6.6.

thisUpdate:

This field contains the time when the manifest was created. This field has the same format constraints as specified in [RFC5280] for the CRL field of the same name. The issuer MUST ensure that the value of this field is more recent than any previously-generated Manifest. Each RP MUST verify that this field value is greater (more recent) than the most recent Manifest it has validated. If this field in a purported "new" Manifest is smaller (less recent) than previously-validated Manifests, the RP SHOULD use locally cached versions of objects, as described in Section 6.6.

nextUpdate:

This field contains the time at which the next scheduled manifest will be issued. The value of nextUpdate MUST be later than the value of thisUpdate. The specification of the GeneralizedTime value is the same as required for the thisUpdate field.

If the authority alters any of the items that it has published in the repository publication point, then the authority MUST issue a new manifest. Even if no changes are made to objects at a publication point, a new manifest MUST be issued before the nextUpdate time. Each manifest encompasses a CRL, and the nextUpdate field of the manifest SHOULD match that of the CRL's nextUpdate field, as the manifest will be re-issued when a new CRL is published. When a new manifest is issued before the time specified in nextUpdate of the current manifest, the CA MUST also issue a new CRL that revokes the EE certificate corresponding to the old manifest.

fileHashAlg:

This field contains the OID of the hash algorithm used to hash the files that the authority has placed into the repository. The hash algorithm used MUST conform to the RPKI Algorithms and Key Size Profile specification [RFC7935].

fileList:

This field is a sequence of FileAndHash objects. There is one FileAndHash entry for each currently valid signed object that has been published by the authority (at this publication point). Each FileAndHash is an ordered pair consisting of the name of the file in the repository publication point (directory) that contains the object in question and a hash of the file's contents.

4.2.2. Names in FileAndHash objects

Names that appear in the fileList MUST consist of one or more characters chosen from the set a-z, A-Z, 0-9, - (HYPHEN), or _ (UNDERSCORE), followed by a single . (DOT), followed by a three-letter extension. The extension MUST be one of those enumerated in the "RPKI Repository Naming Scheme" registry maintained by IANA [IANA-NAMING].

As an example, 'vixxBTS_TVXQ-2pmGOT7.cer' is a valid filename.

The example above contains a mix of uppercase and lowercase characters in the filename. CAs and RPs MUST be able to perform filesystem operations in a case-sensitive, case-preserving manner.

4.3. Content-Type Attribute

The mandatory content-type attribute MUST have its attrValues field set to the same OID as eContentType. This OID is id-ct-rpkiManifest and has the numerical value of 1.2.840.113549.1.9.16.1.26.

4.4. Manifest Validation

To determine whether a manifest is valid, the RP MUST perform the following checks in addition to those specified in [RFC6488]:

1. The eContentType in the EncapsulatedContentInfo is id-ad-rpkiManifest (OID 1.2.840.113549.1.9.16.1.26).
2. The version of the rpkiManifest is 0.
3. In the rpkiManifest, thisUpdate precedes nextUpdate.

Note: Although the `thisUpdate` and `nextUpdate` fields in the Manifest `eContent` MUST match the corresponding fields in the CRL associated with the Manifest, RPs MUST NOT reject a manifest solely because these fields are not identical.

If the above procedure indicates that the manifest is invalid, then the manifest MUST be discarded and treated as though no manifest were present.

5. Manifest Generation

5.1. Manifest Generation Procedure

For a CA publication point in the RPKI repository system, a CA MUST perform the following steps to generate a manifest:

1. Generate a new key pair for use in a "one-time-use" EE certificate.
2. Issue an EE certificate for this key pair. The CA MUST revoke the EE certificate used for the manifest being replaced.

This EE certificate MUST have an SIA extension access description field with an `accessMethod` OID value of `id-ad-signedobject`, where the associated `accessLocation` references the publication point of the manifest as an object URL. (RPs are required to verify both of these syntactic constraints.)

This EE certificate MUST describe its Internet Number Resources (INRs) using the "inherit" attribute, rather than explicit description of a resource set (see [RFC3779]). (RPs are required to verify this.)

The validity interval of the EE certificate MUST exactly match the `thisUpdate` and `nextUpdate` times specified in the manifest's `eContent`. (An RP MUST NOT consider misalignment of the validity interval misalignment in and of itself to be an error.)

3. The EE certificate MUST NOT be published in the authority's repository publication point.
4. Construct the manifest content.

The manifest content is described in Section 4.2.1. The manifest's `fileList` includes the file name and hash pair for each object issued by this CA that has been published at this repository publication point (directory). The collection of objects to be included in the manifest includes all certificates

issued by this CA that are published at the CA's repository publication point, the most recent CRL issued by the CA, and all objects verified by EE certificates that were issued by this CA that are published at this repository publication point. (Sections 6.1-5 describes the checks that an RP MUST perform in support of the manifest content noted here.)

Note that the manifest does not include a self reference (i.e., its own file name and hash), since it would be impossible to compute the hash of the manifest itself prior to it being signed.

5. Encapsulate the manifest content using the CMS SignedData content type (as specified Section 4), sign the manifest using the private key corresponding to the subject key contained in the EE certificate, and publish the manifest in the repository system publication point that is described by the manifest. (RPs are required to verify the CMS signature.)
6. Because the key pair is to be used only once, the private key associated with this key pair MUST now be destroyed.

5.2. Considerations for Manifest Generation

A new manifest MUST be issued and published before the nextUpdate time.

An authority MUST issue a new manifest in conjunction with the finalization of changes made to objects in the publication point. If any named objects in the publication point are replaced, the authority MUST ensure that the file hash for each replaced object is updated accordingly in the new manifest. Additionally, the authority MUST revoke the certificate associated with each replaced object (other than a CRL), if it is not expired. An authority MAY perform a number of object operations on a publication repository within the scope of a repository change before issuing a single manifest that covers all the operations within the scope of this change. Repository operators MUST implement some form of repository update procedure that mitigates, to the extent possible, the risk that RPs that are performing retrieval operations on the repository are exposed to inconsistent, transient, intermediate states during updates to the repository publication point (directory) and the associated manifest.

Since the manifest object URL is included in the SIA of issued certificates, a new manifest MUST NOT invalidate the manifest object URL of previously issued certificates. This implies that the manifest's publication name in the repository, in the form of an object URL, is unchanged across manifest generation cycles.

When a CA entity is performing a key rollover, the entity MAY choose to have two CA instances simultaneously publishing into the same repository publication point. In this case, there will be one manifest associated with each active CA instance that is publishing into the common repository publication point (directory).

6. Relying Party Processing of Manifests

Each RP MUST use the current manifest of a CA to control addition of listed files to the set of signed objects the RP employs for validating basic RPKI objects: certificates, ROAs, and CRLs. Any files not listed on the manifest MUST NOT be used for validation of these objects. However, files not listed on a manifest MAY be employed to validate other signed objects, if the profile of the object type explicitly states that such behavior is allowed (or required). Note that relying on files not listed in a manifest may allow an attacker to effect substitution attacks against such objects.

As noted earlier, manifests are designed to allow an RP to detect manipulation of repository data, errors by a CA or repository manager, and/or active attacks on the communication channel between an RP and a repository. Unless all of the files enumerated in a manifest can be obtained by an RP during a fetch operation, the fetch is considered to have failed and the RP MUST retry the fetch later.

[RFC6480] suggests (but does not mandate) that the RPKI model employ fetches that are incremental, e.g., an RP transfers files from a publication point only if they are new/changed since the previous, successful, fetch represented in the RP's local cache. This document avoids language that relies on details of the underlying file transfer mechanism employed by an RP and a publication point to effect this operation. Thus the term "fetch" refers to an operation that attempts to acquire the full set of files at a publication point, consistent with the id-ad-rpkiManifest URI extracted from a CA certificate's SIA (see below).

If a fetch fails, it is assumed that a subsequent fetch will resolve problems encountered during the fetch. Until such time as a successful fetch is executed, an RP SHOULD use cached data from a previous, successful fetch. This response is intended to prevent an RP from misinterpreting data associated with a publication point, and thus possibly treating invalid routes as valid, or vice versa.

The processing described below is designed to cause all RPs with access to the same local cache and RPKI repository data to acquire the same set of validated repository files. It does not ensure that the RPs will achieve the same results with regard to validation of

RPKI data, since that depends on how each RP resolves any conflicts that may arise in processing the retrieved files. Moreover, in operation, different RPs will access repositories at different times, and some RPs may experience local cache failures, so there is no guarantee that all RPs will achieve the same results with regard to acquisition or validation of RPKI data.

Note also that there is a "chicken and egg" relationship between the manifest and the CRL for a given CA instance. If the EE certificate for the current manifest is revoked, i.e., it appears in the current CRL, then the CA or publication point manager has made a serious error. In this case the fetch has failed; proceed to Section 6.6. Similarly, if the CRL is not listed on a valid, current manifest, acquired during a fetch, the fetch has failed; proceed to Section 6.6, because the CRL is considered missing.

6.1. Manifest Processing Overview

For a given publication point, an RP MUST perform a series of tests to determine which signed object files at the publication point are acceptable. The tests described below (Section 6.2 to Section 6.5) are to be performed using the manifest identified by the id-ad-rpkiManifest URI extracted from a CA certificate's SIA. All of the files referenced by the manifest MUST be located at the publication point specified by the id-ad-caRepository URI from the (same) CA certificate's SIA. The manifest and the files it references MUST reside at the same publication point. If an RP encounters any files that appear on a manifest but do not reside at the same publication point as the manifest the RP MUST treat the fetch as failed, and a warning MUST be issued (see Section 6.6 below).

Note that, during CA key rollover [RFC6489], signed objects for two or more different CA instances will appear at the same publication point. Manifest processing is to be performed separately for each CA instance, guided by the SIA id-ad-rpkiManifest URI in each CA certificate.

6.2. Acquiring a Manifest for a CA

The RP MUST fetch the manifest identified by the SIA id-ad-rpkiManifest URI in the CA certificate. If an RP cannot retrieve a manifest using this URI, or if the manifest is not valid (Section 4.4), an RP MUST treat this as a failed fetch and proceed to Section 6.6; otherwise proceed to Section 6.3.

6.3. Detecting Stale and or Prematurely-issued Manifests

The RP MUST check that the current time (translated to UTC) is between thisUpdate and nextUpdate. If the current time lies within this interval, proceed to Section 6.4. If the current time is earlier than thisUpdate, the CA may have made an error or the RP's local notion of time may be in error; the RP MUST treat this as a failed fetch and proceed to Section 6.6. If the current time is later than nextUpdate, then the manifest is stale; this is a failed fetch and RP MUST proceed to Section 6.6; otherwise proceed to Section 6.4.

6.4. Acquiring Files Referenced by a Manifest

The RP MUST acquire all of the files enumerated in the manifest (fileList) from the publication point. If there are files listed in the manifest that cannot be retrieved from the publication point, the fetch has failed and the RP MUST proceed to Section 6.6; otherwise, proceed to Section 6.5.

6.5. Matching File Names and Hashes

The RP MUST verify that the hash value of each file listed in the manifest matches the value obtained by hashing the file acquired from the publication point. If the computed hash value of a file listed on the manifest does not match the hash value contained in the manifest, then the fetch has failed and the RP MUST proceed to Section 6.6.

6.6. Failed Fetches

If a fetch fails for any of the reasons cited in 6.2-6.5, the RP MUST issue a warning indicating the reason(s) for termination of processing with regard to this CA instance. It is RECOMMENDED that a human operator be notified of this warning.

Termination of processing means that the RP SHOULD continue to use cached versions of the objects associated with this CA instance, until such time as they become stale or they can be replaced by objects from a successful fetch. This implies that the RP MUST NOT try to acquire and validate subordinate signed objects, e.g., subordinate CA certificates, until the next interval when the RP is scheduled to fetch and process data for this CA instance.

7. Publication Repositories

The RPKI publication system model requires that every publication point be associated with one or more CAs, and be non-empty. Upon creation of the publication point associated with a CA, the CA MUST create and publish a manifest as well as a CRL. A CA's manifest will always contain at least one entry, i.e., a CRL issued by the CA [RFC6481], corresponding to the scope of this manifest.

Every published signed object in the RPKI [RFC6488] is published in the repository publication point of the CA that issued the EE certificate, and is listed in the manifest associated with that CA certificate.

8. Security Considerations

Manifests provide an additional level of protection for RPKI RPs. Manifests can assist an RP to determine if a repository object has been deleted, occluded, or otherwise removed from view, or if a publication of a newer version of an object has been suppressed (and an older version of the object has been substituted).

Manifests cannot repair the effects of such forms of corruption of repository retrieval operations. However, a manifest enables an RP to determine if a locally maintained copy of a repository is a complete and up-to-date copy, even when the repository retrieval operation is conducted over an insecure channel. In cases where the manifest and the retrieved repository contents differ, the manifest can assist in determining which repository objects form the difference set in terms of missing, extraneous, or superseded objects.

The signing structure of a manifest and the use of the nextUpdate value allows an RP to determine if the manifest itself is the subject of attempted alteration. The requirement for every repository publication point to contain at least one manifest allows an RP to determine if the manifest itself has been occluded from view. Such attacks against the manifest are detectable within the time frame of the regular schedule of manifest updates. Forms of replay attack within finer-grained time frames are not necessarily detectable by the manifest structure.

9. IANA Considerations

As [RFC6488] created and populated the registries "RPKI Signed Object" and three-letter filename extensions for "RPKI Repository Name Schemes," no new action is requested of the IANA.

10. Acknowledgements

The authors would like to acknowledge the contributions from George Michelson and Randy Bush in the preparation of the manifest specification. Additionally, the authors would like to thank Mark Reynolds and Christopher Small for assistance in clarifying manifest validation and RP behavior. The authors also wish to thank Tim Bruijnzeels, Job Snijders, Oleg Muravskiy, Sean Turner, Adianto Wibisono, Murray Kucherawy, Francesca Palombini, Roman Danyliw, Lars Eggert, Robert Wilton, and Benjamin Kaduk for their helpful review of this document.

11. References

11.1. Normative References

- [IANA-NAMING] "RPKI Repository Name Schemes",
<<https://www.iana.org/assignments/rpki/rpki.xhtml#name-schemes>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://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, <<https://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, <<https://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, <<https://www.rfc-editor.org/info/rfc6482>>.
- [RFC7935] Huston, G. and G. Michaelson, Ed., "The Profile for Algorithms and Key Sizes for Use in the Resource Public Key Infrastructure", RFC 7935, DOI 10.17487/RFC7935, August 2016, <<https://www.rfc-editor.org/info/rfc7935>>.

- [RFC6487] Huston, G., Michaelson, G., and R. Loomans, "A Profile for X.509 PKIX Resource Certificates", RFC 6487, DOI 10.17487/RFC6487, February 2012, <<https://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, <<https://www.rfc-editor.org/info/rfc6488>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [X.690] "X.690", <<https://www.itu.int/rec/T-REC-X.690-199511-S!Cor1>>.

11.2. Informative References

- [RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <<https://www.rfc-editor.org/info/rfc5652>>.
- [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, <<https://www.rfc-editor.org/info/rfc3779>>.
- [RFC6480] Lepinski, M. and S. Kent, "An Infrastructure to Support Secure Internet Routing", RFC 6480, DOI 10.17487/RFC6480, February 2012, <<https://www.rfc-editor.org/info/rfc6480>>.
- [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, <<https://www.rfc-editor.org/info/rfc6486>>.
- [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, <<https://www.rfc-editor.org/info/rfc6489>>.

Appendix A. ASN.1 Module

```
RPKIManifest { iso(1) member-body(2) us(840) rsadsi(113549)
               pkcs(1) pkcs9(9) smime(16) mod(0) TBD }
```

```

DEFINITIONS EXPLICIT TAGS ::=
BEGIN

-- EXPORTS ALL --

IMPORTS

CONTENT-TYPE
FROM CryptographicMessageSyntax-2010 -- in [RFC6268]
{ iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
  pkcs-9(9) smime(16) modules(0) id-mod-cms-2009(58) } ;

-- Manifest Content Type

ct-rpkiManifest CONTENT-TYPE ::=
{ TYPE Manifest IDENTIFIED BY id-ct-rpkiManifest }

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

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

id-ct-rpkiManifest OBJECT IDENTIFIER ::= { id-ct 26 }

Manifest ::= SEQUENCE {
  version          [0] INTEGER DEFAULT 0,
  manifestNumber   INTEGER (0..MAX),
  thisUpdate       GeneralizedTime,
  nextUpdate       GeneralizedTime,
  fileHashAlg      OBJECT IDENTIFIER,
  fileList         SEQUENCE SIZE (0..MAX) OF FileAndHash
}

FileAndHash ::= SEQUENCE {
  file  IA5String,
  hash  BIT STRING
}

END

```

Appendix B. Changes since RFC 6486

In 2019, it came to light that multiple Relying Party implementations were in a vulnerable position, possibly due to perceived ambiguity in the original [RFC6486] specification. This document attempts to clarify the innovative concept and application of RPKI Manifests in light of real-world deployment experience in the global Internet routing system, to avoid future problematic cases.

The following list summarizes the changes between RFC 6486 and this document:

- * Forbid "sequential-use" EE certificates, instead mandate "one-time-use" EE certificates.
- * Clarify that Manifest EE certificates are to be issued with a validity period which coincides with the interval specified in the Manifest eContent, which coincides with the CRL's thisUpdate and nextUpdate.
- * Clarify the manifestNumber is monotonically incremented in steps of 1.
- * Recommend CA issuers to coincidence the applicable CRL's nextUpdate with the Manifest's nextUpdate.
- * The set of valid characters in FileAndHash filenames was constrained.
- * Clarifications that an RP unable to obtain the full set of files listed on a Manifest is considered a failure state, in which case cached data from a previous attempt should be used (if available).
- * Clarifications on the requirement for a current CRL to be present, listed, and verified.
- * Removed the notion of 'local policy'.

Authors' Addresses

Rob Austein
Arrcus, Inc.
Email: sra@hactrn.net

Geoff Huston
APNIC
6 Cordelia St
South Brisbane QLD 4101
Australia
Email: gih@apnic.net

Stephen Kent
Independent
Email: kent@alum.mit.edu

Matt Lepinski
New College Florida
5800 Bay Shore Rd.
Sarasota, FL 34243
USA
Email: mlepinski@ncf.edu

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 16 August 2022

J. Snijders
Fastly
T. Harrison
APNIC
B. Maddison
Workonline
12 February 2022

Resource Public Key Infrastructure (RPKI) object profile for Signed
Checklist (RSC)
draft-ietf-sidrops-rpki-rsc-06

Abstract

This document defines a Cryptographic Message Syntax (CMS) profile for a general purpose listing of checksums (a 'checklist'), for use with the Resource Public Key Infrastructure (RPKI). The objective is to allow an attestation, in the form of a listing of one or more checksums of arbitrary digital objects (files), to be signed "with resources", and for validation to provide a means to confirm a specific Internet Resource Holder produced the Signed Checklist. The profile is intended to provide for the signing of an arbitrary checksum listing with a specific set of Internet Number Resources.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

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 <https://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 16 August 2022.

Copyright Notice

Copyright (c) 2022 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 (<https://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 Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction	3
2. RSC Profile and Distribution	3
2.1. RSC End-Entity Certificates	4
3. The RSC ContentType	4
4. The RSC eContent	4
4.1. version	5
4.2. resources	5
4.3. digestAlgorithm	5
4.4. checkList	6
5. RSC Validation	6
6. Operational Considerations	7
7. Security Considerations	7
8. Implementation status	8
9. IANA Considerations	9
9.1. SMI Security for S/MIME CMS Content Type (1.2.840.113549.1.9.16.1)	9
9.2. RPKI Signed Objects sub-registry	9
9.3. File Extension	10
9.4. SMI Security for S/MIME Module Identifier (1.2.840.113549.1.9.16.0)	10
9.5. Media Type	10
10. References	11
10.1. Normative References	11
10.2. Informative References	12
Appendix A. Acknowledgements	13
Appendix B. Document changelog	13
B.1. changes from -05 -> -06	13
B.2. changes from -04 -> -05	13
B.3. changes from -03 -> -04	13
B.4. changes from -02 -> -03	14

B.5. changes from -01 -> -02	14
B.6. changes from -00 -> -01	14
B.7. individual submission phase	14
Authors' Addresses	14

1. Introduction

This document defines a Cryptographic Message Syntax (CMS) [RFC5652] profile for a general purpose listing of checksums (a 'checklist'), for use with the Resource Public Key Infrastructure (RPKI) [RFC6480]. The objective is to allow an attestation, in the form of a listing of one or more checksums of arbitrary files, to be signed "with resources", and for validation to provide a means to confirm a given Internet Resource Holder produced the RPKI Signed Checklist (RSC). The profile is intended to provide for the signing of a checksum listing with a specific set of Internet Number Resources.

Signed Checklists are expected to facilitate inter-domain business use-cases which depend on an ability to verify resource holdership. RPKI-based validation processes are expected to become the industry norm for automated Bring Your Own IP (BYOIP) on-boarding or establishment of physical interconnection between Autonomous Systems.

The RSC concept borrows heavily from RTA [I-D.ietf-sidrops-rpki-rta], Manifests [RFC6486], and OpenBSD's [signify] utility. The main difference between RSC and RTA is that the RTA profile allows multiple signers to attest a single digital object through a checksum of its content, while the RSC profile allows a single signer to attest the existence of multiple digital objects. A single signer profile is considered a simplification for both implementers and operators.

2. RSC Profile and Distribution

RSC follows the Signed Object Template for the RPKI [RFC6488] with one exception. Because RSCs MUST NOT be distributed through the global RPKI Repository system, the Subject Information Access (SIA) extension MUST be omitted from the RSC's X.509 End-Entity (EE) certificate.

What constitutes suitable transport for RSC files 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.

2.1. RSC End-Entity Certificates

The CA MUST only sign one RSC with each EE Certificate, and MUST generate a new key pair for each new RSC. This form of use of the associated EE Certificate is termed a "one-time-use" EE certificate Section 3 of [RFC6487].

3. The RSC ContentType

The ContentType for an RSC is defined as `rpkiSignedChecklist`, and has the numerical value of 1.2.840.113549.1.9.16.1.48.

This OID MUST appear both within the `eContentType` in the `encapContentInfo` object as well as the `ContentType` signed attribute in the `signerInfo` object (see [RFC6488]).

4. The RSC eContent

The content of an RSC indicates that a checklist for arbitrary digital objects has been signed "with resources". An RSC is formally defined as:

```
RpkiSignedChecklist-2021
{ iso(1) member-body(2) us(840) rsadsi(113549)
  pkcs(1) pkcs9(9) smime(16) mod(0) TBD }

DEFINITIONS EXPLICIT TAGS ::=
BEGIN

IMPORTS
    CONTENT-TYPE, Digest, DigestAlgorithmIdentifier
    FROM CryptographicMessageSyntax-2010 -- in [RFC6268]
    { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
      pkcs-9(9) smime(16) modules(0) id-mod-cms-2009(58) }

    ASIdOrRange, IPAddressOrRange
    FROM IPAddrAndASCertExtn -- in [RFC3779]
    { iso(1) identified-organization(3) dod(6) internet(1)
      security(5) mechanisms(5) pkix(7) mod(0)
      id-mod-ip-addr-and-as-ident(30) } ;

ct-rpkiSignedChecklist CONTENT-TYPE ::=
{ TYPE RpkiSignedChecklist IDENTIFIED BY
  id-ct-signedChecklist }

id-ct-signedChecklist OBJECT IDENTIFIER ::=
{ iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
  pkcs-9(9) id-smime(16) id-ct(1) 48 }
```

```
RpkiSignedChecklist ::= SEQUENCE {  
    version [0]          INTEGER DEFAULT 0,  
    resources             ResourceBlock,  
    digestAlgorithm       DigestAlgorithmIdentifier,  
    checkList             SEQUENCE SIZE (1..MAX) OF FileNameAndHash }  
  
FileNameAndHash ::= SEQUENCE {  
    fileName             IA5String OPTIONAL,  
    hash                 Digest }  
  
ResourceBlock ::= SEQUENCE {  
    asID [0]             AsList OPTIONAL,  
    ipAddrBlocks [1]     IPList OPTIONAL }  
-- at least one of asID or ipAddrBlocks MUST be present  
( WITH COMPONENTS { ..., asID PRESENT } |  
  WITH COMPONENTS { ..., ipAddrBlocks PRESENT } )  
  
AsList ::= SEQUENCE (SIZE(1..MAX)) OF ASIdOrRange  
  
IPList ::= SEQUENCE (SIZE(1..MAX)) OF IPAddressFamilyItem  
  
IPAddressFamilyItem ::= SEQUENCE {      -- AFI & optional SAFI --  
    addressFamily        OCTET STRING (SIZE (2..3)),  
    ipAddressOrRange     IPAddressOrRange }
```

END

4.1. version

The version number of the RpkiSignedChecklist MUST be 0.

4.2. resources

The resources contained here are the resources used to mark the attestation, and MUST match the set of resources listed by the EE Certificate carried in the CMS certificates field.

4.3. digestAlgorithm

The digest algorithm used to create the message digest of the attested digital object. This algorithm MUST be a hashing algorithm defined in [RFC7935].

4.4. checkList

This field is a sequence of FileNameAndHash objects. There is one FileNameAndHash entry for each arbitrary object referenced on the Signed Checklist. Each FileNameAndHash is an ordered pair of the name of the directory entry containing the digital object and the message digest of the digital object. The filename field is OPTIONAL.

5. RSC Validation

Before a Relying Party can use an RSC to validate a set of digital objects, the Relying Party MUST first validate the RSC. To validate an RSC, the Relying Party MUST perform all the validation checks specified in [RFC6488] (except checking for the presence of a SIA extension), and perform the following additional RSC-specific validation steps:

1. The RSC specification deviates from Section 4.8.8.2 of [RFC6487], the Subject Information Access extension MUST NOT be present on the End-Entity (EE) certificate signing the RSC.
2. The IP Addresses and AS Identifiers extension [RFC3779] is present in the end-entity (EE) certificate (contained within the RSC), and each IP address prefix(es) and/or AS Identifier(s) in the RSC is contained within the set of IP addresses specified by the EE Certificate's IP Addresses and AS Identifiers delegation extension.
3. For each FileNameAndHash entry in the RSC, if a filename field is present, the field's content MUST contain only characters specified in the Portable Filename Character Set as defined in [POSIX].

To verify a set of digital objects with an RSC:

- * The message digest of each referenced digital object, using the digest algorithm specified in the the digestAlgorithm field, MUST be calculated and MUST match the value given in the messageDigest field of the associated FileNameAndHash, for the digital object to be considered as verified with the RSC.

6. Operational Considerations

When creating digital objects of a plain-text nature (such as ASCII, UTF-8, HTML, Javascript, XML, etc) it is RECOMMENDED to convert such objects into a lossless compressed form. Distributing plain-text objects within a compression envelope (such as GZIP [RFC1952]) might help avoid unexpected canonicalization at intermediate systems (which in turn would lead to checksum verification errors). Validator implementations are expected to treat a checksummed digital object as string of arbitrary single octets.

If a filename field is present, but no referenced digital object has a filename that matches the content of that field, a validator implementation SHOULD compare the message digest of each digital object to the value from the messageDigest field of the associated FileNameAndHash, and report matches to the client for further consideration.

7. Security Considerations

Relying parties are hereby warned that the data in a RPKI Signed Checklist is self-asserted. When determining the meaning of any data contained in an RPKI Signed Checklist, Relying Parties MUST NOT make any assumptions about the signer beyond the fact that it had sufficient control of the issuing CA to create the object. These data have not been verified by the Certificate Authority (CA) that issued the CA certificate to the entity that issued the EE Certificate used to validate the Signed Checklist.

RPKI Certificates are not bound to real world identities, see [I-D.ymbk-sidrops-rpki-has-no-identity] for an elaboration. Relying Parties can only associate real world entities to Internet Number Resources by additionally consulting an exogenous authority. Signed Checklists are a tool to communicate assertions 'signed with Internet Number Resources', not about any other aspect of the resource holder's business operations such as the identity of the resource holder itself.

RSC objects are not distributed through the RPKI Repository system. From this, it follows that third parties who do not have a copy of a given RSC, may not be aware of the existence of that RSC. Since RSC objects use EE Certificates, but all other currently defined types of RPKI object profiles are published in public CA repositories, an observer may infer from discrepancies in the Repository that RSC object(s) may exist. For example, if a CA does not use random serial numbers for Certificates, an observer could detect gaps between the serial numbers of the published EE Certificates. Similarly, if the CA includes a serial number on a CRL that does not match any published object, an observer could postulate an RSC EE Certificate was revoked.

Conversely, a gap in serial numbers does not imply that an RSC exists. Nor does an arbitrary (to the RP unknown) serial in a CRL imply an RSC object exists: the implicitly referenced object might not be a RSC, it might never have been published, or was revoked before it was visible to RPs. In general, it is not possible to confidently infer the existence or non-existence of RSCs from the Repository state without access to a given RSC.

While an one-time-use EE Certificate must only be used to generate and sign a single RSC object, CAs technically are not restricted from generating and signing multiple different RSC objects with a single keypair. Any RSC objects sharing the same EE Certificate can not be revoked individually.

8. Implementation status

This section is to be removed before publishing as an RFC.

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in RFC 7942. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to RFC 7942, "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

- * A signer and validator implementation [rpki-rsc-demo] written in Perl based on OpenSSL was provided by Tom Harrison from APNIC.
- * A signer implementation [rpkimancer] written in Python was developed by Ben Maddison.
- * Example .sig files were created by Job Snijders with the use of OpenSSL.
- * A validator implementation based on OpenBSD rpki-client and LibreSSL was developed by Job Snijders.
- * A validator implementation [FORT] based on the FORT validator was developed by Alberto Leiva.

9. IANA Considerations

9.1. SMI Security for S/MIME CMS Content Type (1.2.840.113549.1.9.16.1)

The IANA has permanently allocated for this document in the SMI Security for S/MIME CMS Content Type (1.2.840.113549.1.9.16.1) registry:

Decimal	Description	References
48	id-ct-signedChecklist	[draft-ietf-sidrops-rpki-rsc]

Upon publication of this document, IANA is requested to reference the RFC publication instead of this draft.

9.2. RPKI Signed Objects sub-registry

The IANA is requested to register the OID for the RPKI Signed Checklist in the registry created by [RFC6488] as following:

Name	OID	Specification
Signed Checklist	1.2.840.113549.1.9.16.1.48	[draft-ietf-sidrops-rpki-rsc]

9.3. File Extension

The IANA is requested to add an item for the Signed Checklist file extension to the "RPKI Repository Name Scheme" registry created by [RFC6481] as follows:

Filename Extension	RPKI Object	Reference
.sig	Signed Checklist	[draft-ietf-sidrops-rpki-rsc]

9.4. SMI Security for S/MIME Module Identifier (1.2.840.113549.1.9.16.0)

The IANA is requested to add an item to the "SMI Security for S/MIME Module Identifier" registry as follows:

Decimal	Description	References
TBD	id-mod-rpkiSignedChecklist-2021	[draft-ietf-sidrops-rpki-rsc]

9.5. Media Type

The IANA is requested to register the media type application/rpki-checklist in the Provisional Standard Media Type registry as follows:

Type name: application
Subtype name: rpki-checklist
Required parameters: None
Optional parameters: None
Encoding considerations: binary
Security considerations: Carries an RPKI Signed Checklist [RFC-TBD].
Interoperability considerations: None
Published specification: This document.
Applications that use this media type: RPKI operators.
Additional information:
 Content: This media type is a signed object, as defined in [RFC6488], which contains a payload of a list of checksums as defined above in this document.
 Magic number(s): None
 File extension(s): .sig
 Macintosh file type code(s):
Person & email address to contact for further information:
 Job Snijders <job@fastly.com>
Intended usage: COMMON
Restrictions on usage: None
Author: Job Snijders <job@fastly.com>
Change controller: Job Snijders <job@fastly.com>

10. References

10.1. 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, <<https://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, <<https://www.rfc-editor.org/info/rfc3779>>.
- [RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <<https://www.rfc-editor.org/info/rfc5652>>.
- [RFC6481] Huston, G., Loomans, R., and G. Michaelson, "A Profile for Resource Certificate Repository Structure", RFC 6481, DOI 10.17487/RFC6481, February 2012, <<https://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, <<https://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, <<https://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, <<https://www.rfc-editor.org/info/rfc6488>>.
- [RFC7935] Huston, G. and G. Michaelson, Ed., "The Profile for Algorithms and Key Sizes for Use in the Resource Public Key Infrastructure", RFC 7935, DOI 10.17487/RFC7935, August 2016, <<https://www.rfc-editor.org/info/rfc7935>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

10.2. Informative References

- [FORT] LACNIC and NIC.MX, "FORT", May 2021, <<https://github.com/NICMx/FORT-validator>>.
- [I-D.ietf-sidrops-rpki-rta] Michaelson, G., Huston, G., Harrison, T., Bruijnzeels, T., and M. Hoffmann, "A profile for Resource Tagged Attestations (RTAs)", Work in Progress, Internet-Draft, draft-ietf-sidrops-rpki-rta-00, 21 January 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-sidrops-rpki-rta-00>>.
- [I-D.ymbk-sidrops-rpki-has-no-identity] Bush, R. and R. Housley, "The I in RPKI does not stand for Identity", Work in Progress, Internet-Draft, draft-ymbk-sidrops-rpki-has-no-identity-00, 16 March 2021, <<https://datatracker.ietf.org/doc/html/draft-ymbk-sidrops-rpki-has-no-identity-00>>.
- [POSIX] IEEE and The Open Group, "The Open Group's Base Specifications, Issue 7", 2016, <<https://publications.opengroup.org/standards/unix/c165>>.

- [RFC1952] Deutsch, P., "GZIP file format specification version 4.3", RFC 1952, DOI 10.17487/RFC1952, May 1996, <<https://www.rfc-editor.org/info/rfc1952>>.
- [RFC6480] Lepinski, M. and S. Kent, "An Infrastructure to Support Secure Internet Routing", RFC 6480, DOI 10.17487/RFC6480, February 2012, <<https://www.rfc-editor.org/info/rfc6480>>.
- [rpki-rsc-demo] Harrison, T., "A proof-of-concept for constructing and validating RPKI Signed Checklists (RSCs).", February 2021, <<https://github.com/APNIC-net/rpki-rsc-demo>>.
- [rpkimancer] Maddison, B., "rpkimancer", May 2021, <<https://github.com/benmaddison/rpkimancer>>.
- [signify] Unangst, T. and M. Espie, "signify - cryptographically sign and verify files", May 2014, <<https://man.openbsd.org/signify>>.

Appendix A. Acknowledgements

The authors wish to thank George Michaelson, Tom Harrison, Geoff Huston, Randy Bush, Stephen Kent, Matt Lepinski, Rob Austein, Ted Unangst, and Marc Espie for prior art. The authors thank Russ Housley for reviewing the ASN.1 notation and providing suggestions. The authors would like to thank Nimrod Levy, Tim Bruijnzeels, Alberto Leiva, Ties de Kock, and Peter Peele for document review and suggestions.

Appendix B. Document changelog

This section is to be removed before publishing as an RFC.

B.1. changes from -05 -> -06

- * Non-content-related updates.

B.2. changes from -04 -> -05

- * Ties contributed clarifications.

B.3. changes from -03 -> -04

- * Alberto pointed out the asID validation also needs to be documented.

B.4. changes from -02 -> -03

- * Reference the IANA assigned OID
- * Clarify validation rules

B.5. changes from -01 -> -02

- * Clarify RSC is part of a puzzle, not panacea. Thanks Randy & Russ.

B.6. changes from -00 -> -01

- * Readability improvements
- * Update document category to match the registry allocation policy requirement.

B.7. individual submission phase

- * On-the-wire change: the 'Filename' switched from 'required' to 'optional'. Some SIDROPS Working Group participants proposed a checksum itself is the most minimal information required to address digital objects.

Authors' Addresses

Job Snijders
Fastly
Amsterdam
Netherlands

Email: job@fastly.com

Tom Harrison
Asia Pacific Network Information Centre
6 Cordelia St
South Brisbane QLD 4101
Australia

Email: tomh@apnic.net

Ben Maddison
Workonline Communications
Cape Town
South Africa

Email: benm@workonline.africa

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 8 September 2022

C. Martinez
LACNIC
G. Michaelson
T. Harrison
APNIC
T. Bruijnzeels
NLnet Labs
R. Austein
Dragon Research Labs
7 March 2022

RPKI Signed Object for Trust Anchor Key
draft-ietf-sidrops-signed-tal-09

Abstract

A Trust Anchor Locator (TAL) is used by Relying Parties (RPs) in the Resource Public Key Infrastructure (RPKI) to locate and validate a Trust Anchor (TA) Certification Authority (CA) certificate used in RPKI validation. This document defines an RPKI signed object for a Trust Anchor Key (TAK), that can be used by a TA to signal the location(s) of the accompanying CA certificate for the current key to RPs, as well as the successor key and the location(s) of its CA certificate. This object helps to support planned key rolls without impacting RPKI validation.

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 <https://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 8 September 2022.

Copyright Notice

Copyright (c) 2022 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 (<https://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 Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Requirements Notation	3
2. Overview	3
3. TAK Object Definition	4
3.1. The TAK Object Content Type	4
3.2. The TAK Object eContent	4
3.2.1. TAKey	4
3.2.2. TAK	5
3.3. TAK Object Validation	5
4. TAK Object Generation and Publication	6
5. Relying Party Use	7
6. Maintaining Multiple TA Keys	9
7. Performing TA Key Rolls	10
7.1. Phase 1: Add a TAK for Key 'A'	10
7.2. Phase 2: Add a Key 'B'	10
7.3. Phase 3: Update TAL to point to 'B'	11
7.4. Phase 4: Remove Key 'A'	11
8. Deployment Considerations	11
9. Security Considerations	12
10. IANA Considerations	13
10.1. OID	13
10.2. File Extension	13
10.3. Module Identifier	13
11. Implementation Status	13
11.1. APNIC	14
12. Revision History	14
13. Acknowledgments	15
14. References	15
14.1. Normative References	15
14.2. Informative References	16
Appendix A. ASN.1 Module	16
Authors' Addresses	17

1. Requirements Notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Overview

A TAL [RFC8630] is used by an RP in the RPKI to locate and validate TA CA certificates used in RPKI validation. However, until now there has been no in-band way of notifying RPs of updates to a TAL. In-band notification means that TAs can be more confident of RPs being aware of key roll operations.

This document defines a new RPKI signed object that can be used to document the location(s) of the TA CA certificate for the current TA key, as well as the value of the successor key and the location(s) of its TA CA certificate. This allows RPs to be notified automatically of such changes, and enables TAs to stage a successor key so that planned key rolls can be performed without risking the invalidation of the RPKI tree under the TA. We call this object the Trust Anchor Key (TAK) object.

When RPs are first bootstrapped, they use a TAL to discover the key and location(s) of the CA certificate for a TA. The RP can then retrieve and validate the CA certificate, and subsequently validate the manifest [RFC6486] and CRL published by that TA (section 5 of [RFC6487]). However, before processing any other objects it will first validate the TAK object, if present. If the TAK object lists only the current key, then the RP continues processing as per normal. If the TAK object includes a successor key, the RP starts an acceptance timer, and then continues processing as per normal. If, during the following validation runs up until the expiry of the acceptance timer, the RP has not observed any changes to the keys and certificate URLs listed in the TAK object, then the RP will fetch the successor key, update its local state with that key and its associated certification location(s), and continue processing using that key.

The primary motivation for this work is being able to migrate from a Hardware Security Module (HSM) produced by one vendor to one produced by another, where the first vendor does not support exporting keys for use by the second. There may be other scenarios in which key rollover is useful, though.

3. TAK Object Definition

The TAK object makes use of the template for RPKI digitally signed objects [RFC6488], which defines a Cryptographic Message Syntax (CMS) [RFC5652] wrapper for the content as well as a generic validation procedure for RPKI signed objects. Therefore, to complete the specification of the TAK object (see Section 4 of [RFC6488]), this document defines:

- * The OID (in Section 3.1) that identifies the signed object as being a TAK. (This OID appears within the eContentType in the encapContentInfo object, as well as the content-type signed attribute in the signerInfo object.)
- * The ASN.1 syntax for the TAK eContent, in Section 3.2.
- * The additional steps required to validate a TAK, in Section 3.3.

3.1. The TAK Object Content Type

This document requests an OID for the TAK object as follows:

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

This OID MUST appear both within the eContentType in the encapContentInfo object, as well as the content-type signed attribute in the signerInfo object (see [RFC6488]).

3.2. The TAK Object eContent

The content of a TAK object is ASN.1 encoded using the Distinguished Encoding Rules (DER) [X.690], and is defined per the module in Appendix A.

3.2.1. TAKey

This structure defines a TA key, similarly to [RFC8630]. It contains a sequence of one or more URIs and a SubjectPublicKeyInfo.

3.2.1.1. certificateURIs

This field is equivalent to the URI section defined in section 2.2 of [RFC8630]. It MUST contain at least one CertificateURI element. Each CertificateURI element contains the IA5String representation of either an rsync URI [RFC5781], or an HTTPS URI [RFC7230].

3.2.1.2. subjectPublicKeyInfo

This field contains a SubjectPublicKeyInfo (section 4.1.2.7 of [RFC5280]) in DER format [X.690].

3.2.2. TAK

3.2.2.1. version

The version number of the TAK object MUST be 0.

3.2.2.2. current

This field contains the TA key of the repository in which the TAK object is published.

3.2.2.3. predecessor

This field contains the TA key that was in use for this TA immediately prior to the current TA key, if applicable.

3.2.2.4. successor

This field contains the TA key to be used in place of the current key, after expiry of the relevant acceptance timer.

3.3. TAK Object Validation

To determine whether a TAK object is valid, the RP MUST perform the following checks in addition to those specified in [RFC6488]:

- * The eContentType OID matches the OID described in Section 3.1.
- * The TAK object appears as the product of a TA CA certificate (i.e. the TA CA certificate is itself the issuer of the EE certificate of the TAK object).
- * The TA CA has published only one TAK object in its repository for this key, and this object appears on the manifest as the only entry using the ".tak" extension (see [RFC6481]).
- * The EE certificate of this TAK object describes its Internet Number Resources (INRs) using the "inherit" attribute.
- * The decoded TAK content conforms to the format defined in Section 3.2.

- * The SubjectPublicKeyInfo value of the current TA key in the TAK object matches that of the TA CA certificate used to issue the EE certificate of the TAK object.

If any of these checks does not succeed, the RP MUST ignore the TAK object, and proceed as though it were not listed on the manifest.

The RP is not required to compare its current set of certificateURIs for the current key with those listed in the TAK object. The RP MAY alert the user that these sets of certificateURIs do not match, with a view to the user manually updating the set of certificateURIs in their configuration. The RP MUST NOT automatically update its configuration to use these certificateURIs in the event of inconsistency, though, because migration of users to new certificateURIs should happen by way of the successor key process.

4. TAK Object Generation and Publication

A TA MAY choose to use TAK objects to communicate its current, predecessor, and successor keys. If a TA chooses to use TAK objects, then it SHOULD generate and publish TAK objects under each of its keys.

A non-normative guideline for naming this object is that the filename chosen for the TAK object in the publication repository be a value derived from the public key part of the entity's key pair, using the algorithm described for CRLs in section 2.2 of [RFC6481] for generation of filenames. The filename extension of ".tak" MUST be used to denote the object as a TAK.

In order to generate a TAK object, the TA MUST perform the following actions:

- * The TA MUST generate a key pair for a "one-time-use" EE certificate to use for the TAK.
- * The TA MUST generate a one-time-use EE certificate for the TAK.
- * This EE certificate MUST have an SIA extension access description field with an accessMethod OID value of id-ad-signedObject, where the associated accessLocation references the publication point of the TAK as an object URL.

- * As described in [RFC6487], an [RFC3779] extension is required in the EE certificate used for this object. However, because the resource set is irrelevant to this object type, this certificate MUST describe its Internet Number Resources (INRs) using the "inherit" attribute, rather than explicit description of a resource set.
- * This EE certificate MUST have a "notBefore" time that matches or predates the moment that the TAK will be published.
- * This EE certificate MUST have a "notAfter" time that reflects the intended duration for which this TAK will be published. If the EE certificate for a TAK object is expired, it MUST no longer be published, but it MAY be replaced by a newly generated TAK object with equivalent content and an updated "notAfter" time.
- * The current TA key for the TAK MUST match that of the TA CA certificate under which the TAK was issued.

5. Relying Party Use

Relying Parties MUST keep a record of the current key for each configured TA, as well as the URI(s) where the CA certificate for this key may be retrieved. This record is typically bootstrapped by the use of a pre-configured (and unsigned) TAL file [RFC8630].

When performing top-down validation, RPs MUST first validate and process the TAK object for its current known key, by performing the following steps:

- * A CA certificate is retrieved and validated from the known URIs as described in sections 3 and 4 of [RFC8630].
- * The manifest and CRL for this certificate are then validated as described in [RFC6487] and [RFC6486].
- * The TAK object, if present, is validated as described in Section 3.3.

If the TAK object includes a successor key, then the RP must verify the successor key by doing the following:

- * performing top-down validation using the successor key, in order to validate the TAK object for the successor TA;
- * ensuring that a valid TAK object exists for the successor TA;

- * ensuring that the successor TAK object's current key matches the initial TAK object's successor key; and
- * ensuring that the successor TAK object's predecessor key matches the initial TAK object's current key.

If any of these steps fails, then the successor key has failed verification.

If the successor key passes verification, and the RP has not seen that successor key on the previous successful validation run for this TA, then the RP:

- * sets an acceptance timer of 30 days for this successor key for this TA;
- * cancels the existing acceptance timer for this TA (if applicable); and
- * continues standard top-down validation as described in [RFC6487] using the current key.

If the successor key passes verification, and the RP has seen that successor key on the previous successful validation run for this TA:

- * if the relevant acceptance timer has not expired, the RP continues standard top-down validation using the current key;
- * otherwise, the RP updates its current known key details for this TA to be those of the successor key, and then begins top-down validation again using the successor key.

If the successor key does not pass verification, or if the TAK object does not include a successor key, the RP cancels the existing acceptance timer for this TA (if applicable).

An RP MUST NOT use a successor key for top-down validation outside of the process described above, except for the purpose of testing that the new key is working correctly. This allows a TA to publish a successor key for a period of time, allowing RPs to test it, while still being able to rely on RPs using the current key for their production RPKI operations.

A successor key may have the same SubjectPublicKeyInfo value as the current key: this will be the case where a TA is updating the certificateURIs for that key.

6. Maintaining Multiple TA Keys

Although an RP that can process TAK objects will only ever use one key for validation (either the current key, or the successor key, once the relevant acceptance timer has expired), an RP that cannot process TAK objects will continue to use the key details per its TAL (or equivalent manual configuration) indefinitely. As a result, even when a TA is using a TAK object in order to migrate clients to a new key, the TA may have to maintain the previous key for a period of time alongside the new key in order to ensure continuity of service for older clients.

For each TA key that a TA is maintaining, the signed material for these keys MUST be published under different directories in the context of the 'id-ad-caRepository' and 'id-ad-rpkiManifest' Subject Information Access descriptions contained on the CA certificates [RFC6487]. Publishing objects under the same directory is potentially confusing for RPs, and could lead to object invalidity in the event of file name collisions.

Also, the CA certificates for each maintained key, and the contents published by each key, MUST be equivalent (except for the TAK object). In other words, for the purposes of RPKI validation, it MUST NOT make a difference which of the keys is used as a starting point.

This means that the IP and AS resources contained on all current CA certificates for the maintained TA keys MUST be the same. Furthermore, for any delegation of IP and AS resources to a child, the TA MUST have an equivalent CA certificate published under each of its keys. Any updates in delegations MUST be reflected under each of its keys. A TA SHOULD NOT publish any other objects besides a CRL, a Manifest, a single TAK object, and any number of CA certificates for delegation to child CAs.

If a TA uses a single remote publication server for its keys, per [RFC8181], then it MUST include all <publish/> and <withdraw/> PDUs for the products of each of its keys in a single query, in order to ensure that they will reflect the same content at all times.

If a TA uses multiple publication servers, then it is by definition inevitable that the content of different keys will be out of sync at times. In such cases, the TA SHOULD ensure that the duration of these moments are limited to the shortest possible time. Furthermore, the following should be observed:

- * In cases where a CA certificate is revoked completely, or replaced by a certificate with a reduced set of resources, these changes will not take effect fully until all the relevant repository publication points have been updated. Given that TA key operations are normally performed infrequently, this is unlikely to be a problem: if the revocation or shrinking of an issued CA certificate is staged for days/weeks, then experiencing a delay of several minutes for the repository publication points to be updated is fairly insignificant.
- * In cases where a CA certificate is replaced by a certificate with an extended set of resources, the TA MUST inform the receiving CA only after all of its repository publication points have been updated. This ensures that the receiving CA will not issue any products that could be invalid if an RP uses a TA key just before the CA certificate was due to be updated.

Finally, note that the publication locations of CA certificates for delegations to child CAs under each key will be different, and therefore the Authority Information Access 'id-ad-caIssuers' values (section 4.8.7 of [RFC6487]) on certificates issued by the child CAs may not be as expected when performing top-down validation, depending on the TA key that is used. However, these values are not critical to top-down validation, so RPs performing such validation MUST NOT reject a certificate simply because this value is not as expected.

7. Performing TA Key Rolls

In this section we will describe how present-day RPKI TAs that use only one key pair, and that do not use TAK objects, can use a TAK object to perform a planned key roll.

7.1. Phase 1: Add a TAK for Key 'A'

Before adding a successor key, a TA may want to confirm that it can maintain a TAK object for its current key only. We will refer to this key as key 'A' throughout this section.

7.2. Phase 2: Add a Key 'B'

The TA can now generate a new key pair for key 'B'. This key MUST now be used to create a new CA certificate for this key, and to issue equivalent CA certificates for delegations to child CAs, as described in Section 6.

At this point, the TA can also construct a new TAL file [RFC8630] for key 'B', and test locally that the validation outcome for the new key is equivalent to that of the other current key(s).

When the TA is certain that both keys are equivalent, and wants to initiate the migration from 'A' to 'B', it issues a new TAK object under key 'A', with key 'A' as the current key for that object, key 'B' as the successor key, and no predecessor key. It also issues a TAK object under key 'B', with key 'B' as the current key for that object, key 'A' as the predecessor key, and no successor key.

Once this has happened, RP clients will start seeing the new key and setting acceptance timers accordingly.

7.3. Phase 3: Update TAL to point to 'B'

At about the time that the TA expects clients to start setting key 'B' as the current key, the TA must release a new TAL file for key 'B'. It SHOULD use a different set of URIs in the TAL compared to the TAK file, so that the TA can learn the proportion of RPs that can successfully validate and use the updated TAK objects.

To support RPs that do not take account of TAK objects, the TA should continue operating key 'A' for a period of time after the expected migration of clients to 'B'. The length of that period of time is a local policy matter for that TA: it might operate the key until no clients are attempting to validate using it, for example.

7.4. Phase 4: Remove Key 'A'

The TA SHOULD now remove all content from the repository used by key 'A', and destroy the private key for key 'A'. RPs attempting to rely on a TAL for key 'A' from this point will not be able to perform RPKI validation for the TA, and will have to update their local state manually, by way of a new TAL file.

8. Deployment Considerations

Including TAK objects while RPs do not support this standard will result in those RPs rejecting these objects. It is not expected that this will result in the invalidation of any other object under a Trust Anchor.

The mechanism introduced here can only be relied on once a majority of RPs support it. Defining when that moment arrives is something that cannot be established at the time of writing this document. The use of unique URIs for keys in TAK objects, different from those used for the corresponding TAL files, should help TAs understand the proportion of RPs that support this mechanism.

Some RPs may purposefully not support this mechanism: for example, they may be implemented or configured such that they are unable to update local current key data. TAs should take this into consideration when planning key rollover. However, these RPs would ideally still notify their operators of planned key rollovers, so that the operator could update the relevant configuration manually.

9. Security Considerations

A TA needs to consider the length of time for which it will maintain previously-current keys and their associated repositories. An RP that is seeded with old TAL data will run for 30 days using the previous key before migrating to the next key, due to the acceptance timer requirements, and this 30-day delay applies to each new key that has been issued since the old TAL data was initially published. It may be better in these instances to have the old publication URLs simply fail to resolve, so that the RP reports an error to its operator and the operator seeds it with up-to-date TAL data immediately.

Once a TA has decided not to maintain a previously-current key and its associated repository, it needs to consider how to protect against an adversary gaining access to that key and its associated publication points in order to send invalid/incorrect data to RPs seeded with the TAL data for that key. One possible mitigation here is to reuse the TA CA certificate URLs from that TAL data for newer keys.

The use of acceptance timers means that an adversary that gains access to a TA's current key is not able to migrate RPs to a new key without delay. Although access to that key does permit arbitrary action within the corresponding TA (assuming that the adversary has control over the relevant publication points), being unable to migrate RPs to a new key means that it is possible for the TA operator to regain control over the key and the TA itself, such that it may not be necessary for all RPs to carry out manual reconfiguration.

If an adversary gains access to the key listed as the successor to a TA's current key (i.e. listed as the successor, but the acceptance timer period has not yet elapsed since it was listed), the TA operator can recover from this by simply removing the successor key from the TAK object.

In general, the risk of key compromise can be mitigated by the use of Hardware Security Modules (HSMs) by TAs, which will guard against theft of a private key, as well as operational processes to guard against (accidental) misuse of the keys in an HSM by operators.

Alternate models of TAL update exist and can be complementary to this mechanism. For example, TAs can liaise directly with validation software developers to include updated and reissued TAL files in new code releases, and use existing code update mechanisms in the RP community to distribute the changes.

10. IANA Considerations

10.1. OID

IANA is asked to add the following to the "RPKI Signed Objects" registry:

Decimal	Description	References
TBD	Trust Anchor Key	[section 3.1]

10.2. File Extension

IANA is asked to add an item for the Signed TAL file extension to the "RPKI Repository Name Scheme" created by [RFC6481] as follows:

Extension	RPKI Object	References
.tak	Trust Anchor Key	[this document]

10.3. Module Identifier

IANA is asked to register an object identifier for one module identifier in the "SMI Security for S/MIME Module Identifier (1.2.840.113549.1.9.16.0)" registry as follows:

Decimal	Description	References
TBD	Trust Anchor Key	[section 3.1]

* Description: RPKISignedTrustAnchorList-2021

* OID: 1.2.840.113549.1.9.16.0.TBD

* Specification: [this document]

11. Implementation Status

NOTE: Please remove this section and the reference to RFC 7942 prior to publication as an RFC.

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC7942]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to RFC 7942, "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

11.1. APNIC

- * Responsible Organization: Asia-Pacific Network Information Centre
- * Location: <https://github.com/APNIC-net/rpki-signed-tal-demo>
- * Description: A proof-of-concept for relying party TAK usage.
- * Level of Maturity: This is a proof-of-concept implementation.
- * Coverage: This implementation includes all of the features described in version 08 of this specification. The repository includes a link to various test TALs that can be used for testing TAK scenarios, too.
- * Contact Information: Tom Harrison, tomh@apnic.net

12. Revision History

- 03 - Last draft under Tim's authorship.
- 04 - First draft with George's authorship. No substantive revisions.
- 05 - First draft with Tom's authorship. No substantive revisions.
- 06 - Rob Kisteleki's critique.
- 07 - Switch to two-key model.

08 - Keepalive.

09 - Acceptance timers, predecessor keys, no long-lived CRL/MFT.

13. Acknowledgments

The authors wish to thank Martin Hoffmann for a thorough review of this document, and Russ Housley for reviewing the ASN.1 definitions and providing a new module for the TAK object.

14. References

14.1. 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, <<https://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, <<https://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, <<https://www.rfc-editor.org/info/rfc5280>>.
- [RFC5781] Weiler, S., Ward, D., and R. Housley, "The rsync URI Scheme", RFC 5781, DOI 10.17487/RFC5781, February 2010, <<https://www.rfc-editor.org/info/rfc5781>>.
- [RFC6481] Huston, G., Loomans, R., and G. Michaelson, "A Profile for Resource Certificate Repository Structure", RFC 6481, DOI 10.17487/RFC6481, February 2012, <<https://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, <<https://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, <<https://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, <<https://www.rfc-editor.org/info/rfc6488>>.
- [RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <<https://www.rfc-editor.org/info/rfc7230>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8181] Weiler, S., Sonalker, A., and R. Austein, "A Publication Protocol for the Resource Public Key Infrastructure (RPKI)", RFC 8181, DOI 10.17487/RFC8181, July 2017, <<https://www.rfc-editor.org/info/rfc8181>>.
- [RFC8630] Huston, G., Weiler, S., Michaelson, G., Kent, S., and T. Bruijnzeels, "Resource Public Key Infrastructure (RPKI) Trust Anchor Locator", RFC 8630, DOI 10.17487/RFC8630, August 2019, <<https://www.rfc-editor.org/info/rfc8630>>.
- [X.690] ITU-T Recommendation X.690 (2002) | ISO/IEC 8825-1:2002, "Information technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", 2002.

14.2. Informative References

- [RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <<https://www.rfc-editor.org/info/rfc5652>>.
- [RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <<https://www.rfc-editor.org/info/rfc7942>>.

Appendix A. ASN.1 Module

This appendix includes the ASN.1 module for the TAK object.

```
<CODE BEGINS>
RPKISignedTrustAnchorList-2021
  { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
    pkcs9(9) smime(16) mod(0) TBD }

DEFINITIONS EXPLICIT TAGS ::=
BEGIN

IMPORTS

CONTENT-TYPE
  FROM CryptographicMessageSyntax-2009 -- in [RFC5911]
  { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
    pkcs9(9) smime(16) modules(0) id-mod-cms-2004-02(41) }

SubjectPublicKeyInfo
  FROM PKIX1Explicit-2009 -- in [RFC5912]
  { iso(1) identified-organization(3) dod(6) internet(1)
    security(5) mechanisms(5) pkix(7) id-mod(0)
    id-mod-pkix1-explicit-02(51) } ;

ct-signedTAL CONTENT-TYPE ::=
  { TYPE TAK IDENTIFIED BY
    id-ct-signedTAL }

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

CertificateURI ::= IA5String

TAKey ::= SEQUENCE {
  certificateURIs SEQUENCE SIZE (1..MAX) OF CertificateURI,
  subjectPublicKeyInfo SubjectPublicKeyInfo
}

TAK ::= SEQUENCE {
  version INTEGER DEFAULT 0,
  current TAKey,
  predecessor TAKey OPTIONAL,
  successor TAKey OPTIONAL
}

END
<CODE ENDS>
```

Authors' Addresses

Carlos Martinez
LACNIC
Email: carlos@lacnic.net
URI: <https://www.lacnic.net/>

George G. Michaelson
Asia Pacific Network Information Centre
6 Cordelia St
South Brisbane
QLD 4101
Australia
Email: ggm@apnic.net

Tom Harrison
Asia Pacific Network Information Centre
6 Cordelia St
South Brisbane
QLD 4101
Australia
Email: tomh@apnic.net

Tim Bruijnzeels
NLnet Labs
Email: tim@nlnetlabs.nl
URI: <https://www.nlnetlabs.nl/>

Rob Austein
Dragon Research Labs
Email: sra@hactrn.net

SIDROPS
Internet-Draft
Intended status: Standards Track
Expires: January 9, 2022

C. Shen
W. Yu
CAICT
Y. Liu
China Mobile
H. Wang
S. Chen
Huawei Technologies
July 08, 2021

Verification of Routes Using Region Authorization
draft-shen-sidrops-region-verification-00

Abstract

BGP routing security is becoming a major issue that affects the normal running of Internet services. Currently, there are many solutions, including ROA authentication and ASPA authentication, to prevent route source hijacking, path hijacking, and route leaking. However, on an actual network, large ISPs with multiple ASes can use carefully constructed routes to bypass ROA and ASPA authentication to attack the target network.

This document defines an region-based authentication method for large ISPs with many ASes to prevent traffic hijacking within ISPs.

Requirements Language

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 RFC 2119 [RFC2119].

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 <https://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 January 9, 2022.

Copyright Notice

Copyright (c) 2021 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 (<https://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. Problem Statement	3
3.1. Route hijacking risk within a single ISP	3
3.2. Route hijacking risk between multiple ISPs	4
4. Region based verification	6
4.1. Single region verification	6
4.2. Multiple region verification	6
4.3. Obtaining Region Information	7
4.4. Comparing with routing policy	7
5. Security Considerations	8
6. Acknowledgements	8
7. References	8
7.1. Normative References	8
7.2. Informative References	8
Authors' Addresses	8

1. Introduction

The design of the Border Gateway Protocol (BGP) lacks a mechanism to validate BGP attributes, which is prone to BGP hijacking and BGP route leaks [RFC7908].

[RFC6811] defines a method for verifying the origin of BGP prefixes, which can resolve the most common source AS hijacking. [I-D.ietf-sidrps-aspa-verification] defines an AS-pairs based authentication method to resolve AS-Path hijacking and route leaking.

However, even if these two technologies are deployed on large ISP networks with many ASs, there is still a risk of being attacked by carefully constructed path hijacking.

2. Terminology

OV: Origin Validation

RPKI: Resource Public Key Infrastructure

RP: Relying Party

3. Problem Statement

Currently, some large ISPs have many public ASes to facilitate management. In these ISPs, only a few ASes are used to connect to external ISPs. However, the sub-ASes of these ISPs also exchange routes to provide services for different customers. Therefore, the route access between these sub-ASes may be attacked by carefully constructed as-path.

3.1. Route hijacking risk within a single ISP

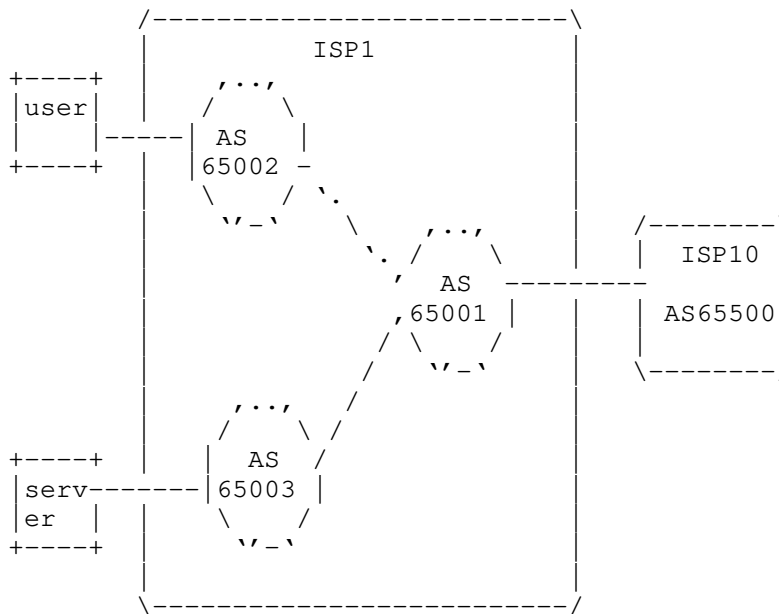


Figure 1 Route hijacking risk within a single ISP

As shown in the Figure 1. ISP1 has AS65001, AS65002, and AS65003 and connects to an external ISP, such as AS65500. There is a server

connect to the AS65003, and a user connecte to the AS65002. AS65003 advertises the server's route to AS65002, and AS65002 uses the route to provide services for users.

After the AS65500 obtains the route for the server, it can spoof the route and change the source AS to AS65003. In this way, the spoofed route is advertised to AS65001 with AS-Path AS65500 AS65003. AS65001 selects routes between the routes advertised by AS65003 and AS65500. Therefore, AS65001 may preferentially select the forged routes of AS65500. As a result, subsequent traffic from users to the server is hijacked to AS65500.

IIn actual deployment, to facilitate traffic adjustment, the mask of the address in the ROA database registered by ISP1 may be in a certain range. In this case, the AS65500 can more easily hijack traffic by using more specific prefixes and spoofing the source AS.

The scenario described here can be prevented by ASPA because the AS pair (AS65500,AS65003) does not exist..

3.2. Route hijacking risk between multiple ISPs

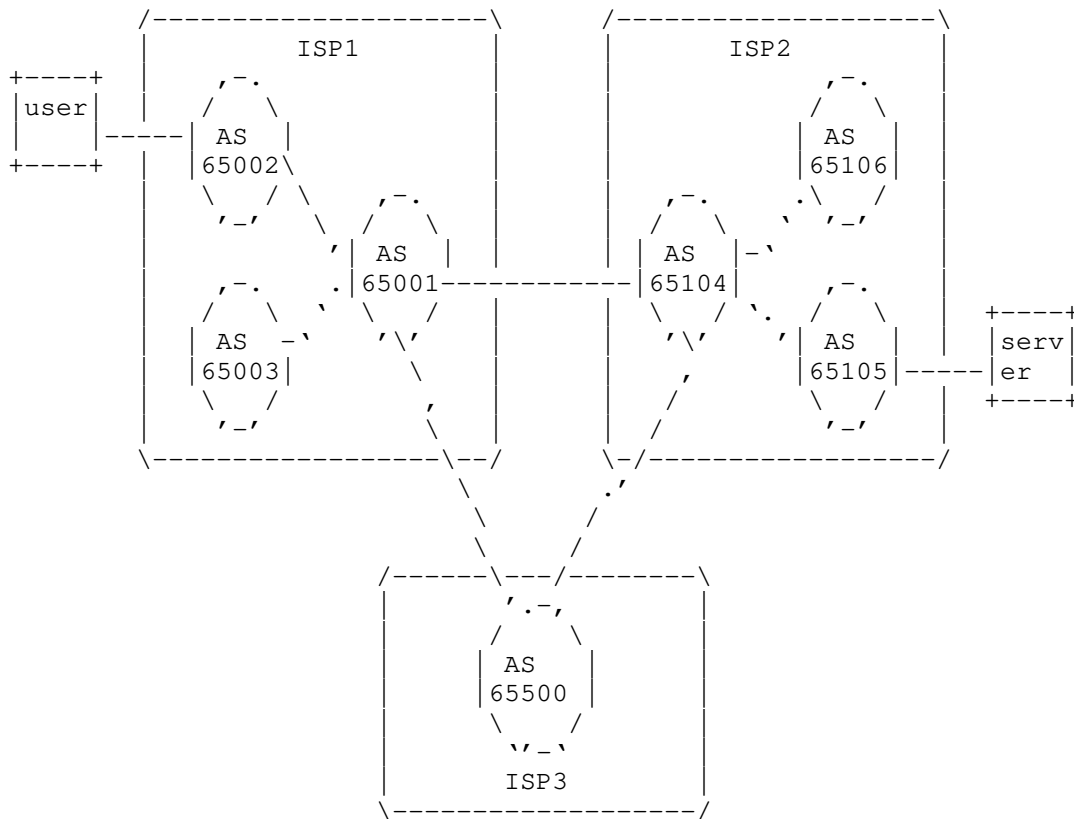


Figure 2 Route hijacking risk between multiple ISPs

As shown in Figure 2. ISP1 has AS65001, AS65002, and AS65003 and connects to external ISPs, such as AS65500 and ISP2's AS65104. ISP2 has AS65104, AS65105, and AS65106, and connects to external ISPs such as AS65500 and ISP1's AS65001. There is a server connect to AS65105, and a user connect to AS65002. AS65105 advertises the server's route to AS65002 through the BGP peer. AS65002 then provides services for users.

The AS65500 can also obtain the route for the server from AS65104. The AS65500 can spoof the route of the server and change the source AS to AS65105. In this way, the AS65500 constructs a more specific prefix, which AS-Path is AS65500 AS65104 AS65105, and advertises the route to AS65001. The traffic from the user to the server will be hijacked to AS65500.

In this scenario it also can't be prevented by ASPA.

4. Region based verification

To solve this problem, we expect to use a region-based verification method. This method is applicable to large ISPs with multiple ASes. In addition to OV verification, region-based verification is performed to prevent the attack scenarios mentioned in section 3.

4.1. Single region verification

As shown in Figure 1, ISP1 can be set to area 1, including AS65001, AS65002, and AS65003.

When a device learns a route, it will verify whether the route is a local region route based on basic OV verification.

The verification process is as follows:

- 1) Perform OV verification on the route. If the OV verification result is valid, then perform area verification.
- 2) Check whether the route's origin AS is belong to local region.
- 3) If not, it indicates that the route is not a local region route. No additional verification is required in single region scenarios..
- 4) If the route's origin AS is belong to local region, check whether the peer that learns the route is belong to local region.
- 5) If the peer that learns a route is not belong to local region, the route verification result is invalid.

If the route verification result is invalid, the route can be consider as an invalid route and is not involved in route selection. This prevents routes belong to local region from being learned by external ASs and prevents possible route hijacking.

4.2. Multiple region verification

For the case of Figure 2, we can set region confederations. ISP1 is set to region 1, including AS65001, AS65002, and AS65003. ISP2 is set to region 2, including AS65104, AS65105, and AS6. In addition, the region of ISP1 and ISP2 form a regional confederation, which is set to regional confederation 1.

The verification process is as follows:

- 1) First, perform the step of region verification. After single region verification step 2, if the route's origin AS is not belong to

local region, then check whether the route belongs to the local confederation.

2) If the route belongs to the local confederation, check whether the peer that learned the route is belong to the local confederation.

3) If the peer is not belong to the local confederation, the route verification result is invalid.

4) Optionally, we may further check whether the peer is the region to which the route belongs. If the region to which the route belongs does not match the region to which the learned peer belongs, we may further consider that route with lowest preference. Of course, we don't usually need to do that.

If the route verification result is invalid, the route can be consider as an invalid route and is not involved in route selection. This prevents routes belong to local region from being learned by external ASs and prevents possible route hijacking.

4.3. Obtaining Region Information

The region information and region confederation information can be obtained in either of the following ways:

1) Obtained through the RP. You can register region data with the RPKI and download the region infomation through the RP.

2) Static configuration. When RP is not ready, we may also use static configuration to implement. You can specify an region, its ASes, and the confederation information to which the region belongs.

Generally, the RPKI mode is recommended..

4.4. Comparing with routing policy

The verification here can be implemented through routing policies.

For example, for region verification, you can configure policies and AS regular expressions. For peers connected to ISP's external ASes , you can configure policies to deny all routes whose origin AS is the local ISP's ASes.

However, in this mode, complex policies need to be configured based on the AS planning of the ISP. In addition, these policies need to be integrated with existing routing policies, which is complex to use.

The RPKI mechanism can be used to verify the area information obtained from the RP, which simplifies the deployment.

5. Security Considerations

NA

6. Acknowledgements

NA

7. References

7.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.

7.2. Informative References

- [I-D.ietf-sidrops-aspa-verification] Azimov, A., Bogomazov, E., Bush, R., Patel, K., and J. Snijders, "Verification of AS_PATH Using the Resource Certificate Public Key Infrastructure and Autonomous System Provider Authorization", draft-ietf-sidrops-aspa-verification-07 (work in progress), February 2021.
- [RFC6811] Mohapatra, P., Scudder, J., Ward, D., Bush, R., and R. Austein, "BGP Prefix Origin Validation", RFC 6811, DOI 10.17487/RFC6811, January 2013, <<https://www.rfc-editor.org/info/rfc6811>>.
- [RFC7908] Sriram, K., Montgomery, D., McPherson, D., Osterweil, E., and B. Dickson, "Problem Definition and Classification of BGP Route Leaks", RFC 7908, DOI 10.17487/RFC7908, June 2016, <<https://www.rfc-editor.org/info/rfc7908>>.

Authors' Addresses

Chen Shen
CAICT
No.52, Hua Yuan Bei Road
Beijing 100191
China

Email: shenchen@caict.ac.cn

Wenyan Yu
CAICT
No.52, Hua Yuan Bei Road
Beijing 100191
China

Email: yuwenyan@caict.ac.cn

Yisong Liu
China Mobile
32 Xuanwumenxi Ave.
Beijing 100032
China

Email: liuyisong@chinamobile.com

Haibo Wang
Huawei Technologies
Huawei Campus, No. 156 Beiqing Road
Beijing 100095
China

Email: rainsword.wang@huawei.com

Shuanglong Chen
Huawei Technologies
Huawei Campus, No. 156 Beiqing Road
Beijing 100095
China

Email: chenshuanglong@huawei.com