Workgroup: Network Working Group
Internet-Draft:
draft-ietf-sidrops-signed-tal-13
Published: 12 March 2023
Intended Status: Standards Track
Expires: 13 September 2023
Authors: C. Martinez G. Michaelson T. Harrison
LACNIC APNIC APNIC
T. Bruijnzeels R. Austein
NLnet Labs Dragon Research Labs

RPKI Signed Object for Trust Anchor Key

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 13 September 2023.

Table of Contents

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-signedTAL OBJECT IDENTIFIER ::=
   { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9)
     smime(16) ct(1) 50 }
```

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 zero or more comments, one or more certificate URIs, and a SubjectPublicKeyInfo.

3.2.1.1. comments

This field is equivalent to the comment section defined in section 2.2 of [RFC8630]. Each comment is human-readable informational UTF-8 text, conforming to the restrictions defined in Section 2 of [RFC5198]. The leading "#" character is omitted.

3.2.1.2. 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].

3.2.1.3. 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.

5.1. Manual update of TA key details

A Relying Party may opt not to support the automatic transition of TA key data, as defined in the previous section. An alternative approach is for the Relying Party to alert the user when a new successor key is seen, and also when the relevant acceptance timer has expired. The user can then manually transition to the new TA key data. This process ensures that the benefits of the acceptance timer

period are still realised, as compared with TA key update based on a TAL distributed out-of-band by a TA.

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. Using TAK objects to distribute TAL data

Relying Parties must be configured with RPKI Trust Anchor data in order to function correctly. This Trust Anchor data is typically distributed in the Trust Anchor Locator (TAL) format defined in RFC 8630. A TAK object can also serve as a format for distribution of this data, though, because the TAKey data stored in the TAK object contains the same data that would appear in a TAL for the associated Trust Anchor.

Relying Parties may support conversion of TAK objects into TAL files. Relying Parties that support conversion MUST validate the TAK object using the process from section 3.3. One exception to the standard validation process in this context is that a Relying Party

MAY treat a TAK object as valid, even though it is associated with a Trust Anchor that the Relying Party is not currently configured to trust. If the Relying Party is relying on this exception when converting a given TAK object, the Relying Party MUST communicate that fact to the user.

When converting a TAK object, a Relying Party MUST default to producing a TAL file based on the 'current' TAKey in the TAK object, though it MAY optionally support producing TAL files based on the 'predecessor' and 'successor' TAKeys.

When converting a TAK object, a Relying Party MUST include in the TAL file any comments from the corresponding TAKey.

If TAK object validation fails, then the Relying Party MUST NOT produce a TAL file based on the TAK object.

Users should be aware that TAK objects distributed out-of-band have similar security properties to TAL files (i.e. there is no authentication). In particular, TAK objects that are not signed by TAs with which the Relying Party is currently configured should only be used if the source that distributes them is one the user trusts to distribute TAL files.

If a Relying Party is not transitioning to new Trust Anchor data using the automatic process described in section 5 or the partially-manual process described in section 5.1, then the user will have to rely on an out-of-band mechanism for validating and updating the Trust Anchor data for the Relying Party. Users in this situation should take similar care when updating a trust anchor using a TAK object file as when using a TAL file to update TA data.

9. Deployment Considerations

9.1. Relying Party Support

Publishing 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.

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.

9.2. Alternate Transition Models

Alternate models of TAL update exist and are complementary to this mechanism. For example, TAs can liaise directly with RP 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.

Additionally, these non-TA channels for distributing TAL data may themselves rely on monitoring for TAK objects and then updating the TAL data in their distributions or packages accordingly. In this way, TAK objects may be useful even for RPs that don't implement in-band support for the protocol.

Non-TA channels for distributing TAL data should ensure so far as is possible that their update mechanisms take account of any changes that a user has made to their local TA key configuration. For example, if a new key is published for a TA, but the non-TA channel's mechanism is able to detect that a user had removed the TA's previous key from their local TA key configuration such that the user no longer relies on it, then the mechanism should not by default add the new key to the user's TA key configuration.

10. Operational Considerations

10.1. Acceptance Timers

Acceptance timers are used in TAK objects in order to permit RPs to test that the new key is working correctly. This in turn means that the TA will be able to gain confidence in the correct functioning of the new key before RPs are relying on that in their production RPKI operations. If a successor key is not working correctly, a TA may remove that key from the current TAK object.

A TA that removes a successor key from a TAK object SHOULD NOT add the same successor key back into the TAK object for that TA. This is because there may be an RP that has fetched the TAK object while the successor key was listed in it, and has started an acceptance timer accordingly, but has not fetched the TAK object during the period when the successor key was not listed in it. If the unchanged successor key is added back in to the TA, such an RP will transition to using new the TA key more quickly than other RPs, which may in turn make debugging and similar more complicated. A simple way of addressing this problem in a situation where the TA doesn't want to reissue the SubjectPublicKeyInfo content for the successor key that was withdrawn is to update the URL set for the successor key, since RPs must take that URL set into account for the purposes of initiating and cancelling acceptance timers.

11. Security Considerations

11.1. Previous Keys

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, the TA SHOULD destroy that key. The TA SHOULD also reuse the TA CA certificate URLs from the previous TAL data for the next TAL that it generates. These measures will help to mitigate the risk of an adversary gaining access to the key and its associated publication points in order to send invalid/incorrect data to RPs seeded with the TAL data for that key.

11.2. TA Compromise

TAK objects do not offer protection against compromise of the current TA key or the successor TA key. TA key compromise in general is out of scope for this document.

12. IANA Considerations

12.1. Content Type

IANA is asked to register an object identifier for one content type in the "SMI Security for S/MIME CMS Content Type (1.2.840.113549.1.9.16.1)" registry as follows:

Decimal	Description	References
50	id-ct-signedTAL	[section 3.1]

*Description: id-ct-signedTAL

*OID: 1.2.840.113549.1.9.16.1.50

*Specification: [section 3.1]

12.2. Signed Object

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

Name	OID	Reference
Trust Anchor Key	1.2.840.113549.1.9.16.1.50	[section 3.1]

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

> Objects of the types listed in this registry, as well as RPKI resource certificates and CRLs, are expected to be validated using the RPKI.

12.3. 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:

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

12.4. 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
74	RPKISignedTrustAnchorList-2021	[this document]

*Description: RPKISignedTrustAnchorList-2021

*OID: 1.2.840.113549.1.9.16.0.74

*Specification: [this document]

12.5. Registration of Media Type application/rpki-signed-tal

IANA is asked to register the media type "application/rpki-signed-tal" in the "Media Types" registry as follows:

Type name: application

Subtype name: rpki-signed-tal

Required parameters:
N/A

Optional parameters: N/A

Encoding considerations: binary

Security considerations: Carries an RPKI Signed TAL. This media type contains no active content. See the Security Considerations section of RFC XXXX for further information.

Interoperability considerations: N/A

Published specification: RFC XXXX

Applications that use this media type: RPKI operators

Fragment identifier considerations: N/A

Additional information:

Content: This media type is for a signed object, as defined in RFC 6488, which contains trust anchor key material as defined in RFC XXXX.

Magic number(s): N/A

File extension(s): .tak

Macintosh file type code(s): N/A

Person & email address to contact for further information: iesg@ietf.org

Intended usage: COMMON

Restrictions on usage: N/A

Author: sidrops WG

Change controller: IESG

13. 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".

13.1. 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 13 of this specification, except for writing TAL files based on TAK data. 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

13.2. rpki-client

*Responsible Organization: Job Snijders

*Location: https://marc.info/?l=openbsd-tech&m=166635746808783&w=2

*Description: A relying party implementation which can validate TAKs.

*Level of Maturity: Mature. Trust Anchor operators are encouraged to use rpki-client as part of smoke testing to help ensure high levels of standards compliance when introducing changes, and use rpki-client in a continuous monitoring fashion to help maintain high levels of operational excellence.

*Coverage: This implementation includes all features except TAK acceptance timers.

*Contact information: Job Snijders, job@fastly.com

14. 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.

10 - Using TAK objects for distribution of TAL data.

11 - Manual update guidance, additional security considerations, identifier updates.

12 - TAK object comments.

13 - Removal of compromise text, extra RP support text, key destruction text, media type registration, signed object registry note.

15. Acknowledgments

The authors wish to thank Martin Hoffmann for a thorough review of the document, Russ Housley for multiple reviews of the ASN.1 definitions and for providing a new module for the TAK object, Job Snijders for the extensive suggestions around TAK object structure/distribution and rpki-client implementation work, and Ties de Kock for text/suggestions around TAK/TAL distribution and general security considerations.

16. References

16.1. 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>.

[RFC5198] Klensin, J. and M. Padlipsky, "Unicode Format for Network Interchange", RFC 5198, DOI 10.17487/RFC5198, March 2008, <https://www.rfc-editor.org/info/rfc5198>.

[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.

16.2. 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) 74 }

DEFINITIONS EXPLICIT TAGS ::=
BEGIN

IMPORTS

CONTENT-TYPE
    FROM CryptographicMessageSyntax-2009 -- in [RFC5911]
    { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
      pkcs-9(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) 50 }

CertificateURI ::= IA5String

TAKey ::= SEQUENCE {
    comments  SEQUENCE SIZE (0..MAX) OF UTF8String,
    certificateURIs  SEQUENCE SIZE (1..MAX) OF CertificateURI,
    subjectPublicKeyInfo  SubjectPublicKeyInfo
}

TAK ::= SEQUENCE {
    version     INTEGER DEFAULT 0,
    current     TAKey,
    predecessor [0] TAKey OPTIONAL,
    successor   [1] 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