Stream: Internet Engineering Task Force (IETF)
RFC: 9691
Category: Standards Track
Published: November 2024
ISSN: 2070-1721
Authors: C. Martinez, *LACNIC*; G. Michaelson, *APNIC*; T. Harrison, *APNIC*; T. Bruijnzeels, *RIPE NCC*
R. Austein, *Dragon Research Labs*

RFC 9691
Resource Public Key Infrastructure (RPKI) Signed Objects for Trust Anchor Keys (TAKs)

Abstract

A Trust Anchor Locator (TAL) is used by Relying Parties (RPs) in the Resource Public Key Infrastructure (RPKI) to locate and validate Trust Anchor (TA) Certification Authority (CA) certificates 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 public key to RPs, as well as the successor public key and the location(s) of its CA certificate. This object helps to support planned key rollovers without impacting RPKI validation.

Status of This Memo

This is an Internet Standards Track document.

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

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9691.

Copyright Notice

Table of Contents

1. Overview 3
   1.1. Requirements Notation 4
2. TAK Object Definition 4
   2.1. The TAK Object Content Type 4
   2.2. The TAK Object eContent 5
      2.2.1. TAKey 5
      2.2.2. TAK 5
   2.3. TAK Object Validation 5
3. TAK Object Generation and Publication 6
4. Relying Party Use 7
   4.1. Manual Update of TA Public Key Details 8
5. Maintaining Multiple TA Key Pairs 8
6. Performing TA Key Rolls 9
   6.1. Phase 1: Add a TAK for Key Pair 'A' 9
   6.2. Phase 2: Add a Key Pair 'B' 9
   6.3. Phase 3: Update TAL to point to 'B' 10
   6.4. Phase 4: Remove Key Pair 'A' 10
7. Using TAK Objects to Distribute TAL Data 10
8. Deployment Considerations 11
   8.1. Relying Party Support 11
   8.2. Alternate Transition Models 11
9. Operational Considerations 12
   9.1. Acceptance Timers 12

10. Security Considerations 12
10.1. Previous Keys 12
10.2. TA Compromise 13
10.3. Alternate Transition Models 13
11. IANA Considerations 13
11.1. Content Type 13
11.2. Signed Object 13
11.3. File Extension 14
11.4. Module Identifier 14
11.5. Registration of Media Type application/rpki-signed-tal 14
12. References 15
12.1. Normative References 15
12.2. Informative References 16
Appendix A. ASN.1 Module 17
Acknowledgments 18
Authors' Addresses 18

1. Overview

A Trust Anchor Locator (TAL) [RFC8630] is used by Relying Parties (RPs) in the Resource Public Key Infrastructure (RPKI) to locate and validate Trust Anchor (TA) Certification Authority (CA) certificates used in RPKI validation. However, until now, there has been no in-band way of notifying RPs of updates to a TAL. An in-band notification means that TA operators can be more confident of RPs being aware of key rollover 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 public key, as well as the value of the successor public 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 public key so that planned key rollovers 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 public 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 [RFC9286] and Certificate Revocation List (CRL) published by that TA

(Section 5 of [RFC6487]). However, before processing any other objects, it will first validate the TAK object if it is present. If the TAK object lists only the current public key, then the RP continues processing as it would in the absence of a TAK object. If the TAK object includes a successor public key, the RP starts a 30-day acceptance timer for that key and then continues standard top-down validation with the current public key. If, during the following validation runs up until the expiry of the acceptance timer, the RP has not observed any changes to the public keys and certificate URLs listed in the TAK object, then the RP will fetch the successor public key, update its local state with that public key and its associated certification location(s), and continue processing using that public 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 private keys for use by the second. There may be other scenarios in which key rollover is useful, though.

1.1. 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. 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 following:

- the OID (Section 2.1) 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 (Section 2.2)
- the additional steps required to validate a TAK (Section 2.3)

2.1. The TAK Object Content Type

This document specifies 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 in both the eContentType in the encapContentInfo object and the content-type signed attribute in the signerInfo object (see [RFC6488]).

2.2. 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.

2.2.1. TAKey

This structure defines a TA public key similar to that from [RFC8630]. It contains a sequence of zero or more comments, one or more certificate URIs, and a SubjectPublicKeyInfo.

comments: This field is semantically equivalent to the comment section defined in Section 2.2 of [RFC8630]. Each comment is human-readable informational UTF-8 text [RFC3629], conforming to the restrictions defined in Section 2 of [RFC5198]. The leading "#" character that is used to denote a comment in [RFC8630] is omitted here.

certificateURIs: This field is semantically 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 [RFC9110].

subjectPublicKeyInfo: This field contains a SubjectPublicKeyInfo (Section 4.1.2.7 of [RFC5280]) in the DER format [X.690].

2.2.2. TAK

version: The version number of the TAK object **MUST** be 0.

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

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

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

2.3. 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 2.1.
- The TAK object appears as the product of a TA CA certificate (i.e., the TA CA certificate itself is the issuer of the End-Entity (EE) certificate of the TAK object).
- The TA CA has published only one TAK object in its repository for this public 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 2.2.
- The SubjectPublicKeyInfo value of the current TA public 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 do 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 public 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. However, the RP **MUST NOT** automatically update its configuration to use these certificateURIs in the event of inconsistency because the migration of users to new certificateURIs should happen by way of the successor public key process.

3. TAK Object Generation and Publication

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 one-time-use EE certificate for the TAK.
- This EE certificate **MUST** have a unique key pair.
- This EE certificate **MUST** have a Subject Information Access (SIA) [RFC6487] 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], the EE certificate used for this object must include an [RFC3779] extension. However, because the resource set is irrelevant to this object type, this certificate **MUST** describe its INRs using the "inherit" attribute rather than explicitly describing 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 public key for the TAK **MUST** match that of the TA CA certificate under which the TAK was issued.

In distribution contexts that support media types, the "application/rpki-signed-tal" media type can be used for TAK objects.

4. Relying Party Use

RPs **MUST** keep a record of the current public key for each configured TA, as well as the URI(s) where the CA certificate for this public 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 public 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 [RFC9286].
- If the TAK object is present, it is validated as described in Section 2.3.

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

- perform top-down validation using the successor public key in order to validate the TAK object for the successor TA,
- ensure that a valid TAK object exists for the successor TA,
- ensure that the successor TAK object's current public key matches the initial TAK object's successor public key, and
- ensure that the successor TAK object's predecessor public key matches the initial TAK object's current public key.

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

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

- sets an acceptance timer of 30 days for this successor public 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 public key.

If the successor public key passes verification and the RP has seen that successor public 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 public key;
- otherwise, the RP updates its current known public key details for this TA to be those of the successor public key, and then begins top-down validation again using the successor public key.

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

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

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

4.1. Manual Update of TA Public Key Details

A Relying Party may opt to not support the automatic transition of TA public key data, as defined in Section 4. An alternative approach is for the Relying Party to alert the user when a new successor public key is seen and when the relevant acceptance timer has expired. The user can then manually transition to the new TA public key data. This process ensures that the benefits of the acceptance timer period are still realised, as compared with TA public key update based on a TAL distributed out of band by a TA.

5. Maintaining Multiple TA Key Pairs

Although an RP that can process TAK objects will only ever use one public key for validation (either the current public key or the successor public key once the relevant acceptance timer has expired), an RP that cannot process TAK objects will continue to use the public 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 public key, the TA may have to maintain the previous key pair for a period of time alongside the new key pair in order to ensure continuity of service for older clients.

For each TA key pair that a TA maintains, the signed material for these key pairs **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 filename collisions.

Also, the CA certificates for each maintained key pair and the content published for each key pair **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 public keys is used as a starting point.

This means that the IP and Autonomous System (AS) resources contained on all current CA certificates for the maintained TA key pairs **MUST** be the same. Furthermore, for any delegation of IP and AS resources to a child CA, the TA **MUST** have an equivalent CA certificate published under each of its key pairs. Any updates in delegations **MUST** be reflected under each of its key pairs. 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 key pairs per [RFC8181], then it **MUST** include all <publish/> and <withdraw/> Protocol Data Units (PDUs) for the products of each of its key pairs in a single query in order to reduce the risk of RPs seeing inconsistent data in the TA's RPKI repositories.

If a TA uses multiple publication servers, then the content for different key pairs will be out of sync at times. The TA **SHOULD** ensure that the duration of these moments is limited to the shortest possible time. Furthermore, the following should be observed:

- In cases where a CA certificate is revoked 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 private 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 relatively 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 public 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 pair will be different; 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 public 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.

6. Performing TA Key Rolls

This section describes how present-day RPKI TAs that use only one key pair and do not use TAK objects can use a TAK object to perform a planned key rollover.

6.1. Phase 1: Add a TAK for Key Pair 'A'

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

6.2. Phase 2: Add a Key Pair 'B'

The TA can now generate a new key pair called 'B'. The private key of this key pair **MUST** now be used to create a new CA certificate for the associated public key and issue equivalent CA certificates for delegations to child CAs as described in Section 5.

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

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

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

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

At about the time that the TA expects clients to start setting the public key from key pair 'B' as the current public key, the TA must release a new TAL file for that public key. 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 pair 'A' for a period of time after the expected migration of clients to the public key from 'B'. The length of that period of time is a local policy matter for that TA; for example, it might operate the key pair until no clients are attempting to validate using the associated public key.

6.4. Phase 4: Remove Key Pair 'A'

The TA **SHOULD** now remove all content from the repository used by key pair 'A' and destroy the private key for that key pair. RPs attempting to rely on a TAL for the public key from key pair '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.

7. 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 TAL format defined in [RFC8630]. 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 2.3. 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 any comments from the corresponding TAKey in the TAL file.

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 4 or the partially manual process described in Section 4.1, 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.

8. Deployment Considerations

8.1. 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 public key data. TA operators 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.

8.2. Alternate Transition Models

Alternate models of TAL updates 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 update 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 public key configuration. For example, if a new public 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 public key from their local TA public key configuration such that the user no longer relies on it, then the mechanism should not add the new public key to the user's TA public key configuration by default.

9. Operational Considerations

9.1. Acceptance Timers

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

A TA that removes a successor public key from a TAK object **SHOULD NOT** add the same successor public 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 public 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 public key was not listed in it. If the unchanged successor public key is added back into the TA, such an RP will transition to using the new TA public 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 operator doesn't want to reissue the SubjectPublicKeyInfo content for the successor public key that was withdrawn is to update the URL set for the successor public key since RPs must take that URL set into account for the purposes of initiating and cancelling acceptance timers.

10. Security Considerations

10.1. Previous Keys

A TA needs to consider the length of time for which it will maintain previously current key pairs and their associated repositories. An RP that is seeded with old TAL data will run for 30 days using the previous public key before migrating to the next public key due to the acceptance timer requirements, and this 30-day delay applies to each new key pair that has been issued since the old TAL data was initially published. In these instances, it may be better for the TA to send error responses when receiving requests for the old publication URLs 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 pair and its associated repository, the TA **SHOULD** destroy the associated private 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 private key and its associated publication points in order to send invalid or incorrect data to RPs seeded with the TAL data for the corresponding public key.

10.2. TA Compromise

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

While it is possible for a malicious actor to use TAK objects to cause RPs to transition from the current TA public key to a successor TA public key, such action is predicated on the malicious actor having compromised the current TA private key in the first place; thus, TAK objects do not alter the security considerations relevant to this scenario.

10.3. Alternate Transition Models

Section 8.2 describes other ways in which a TA may transition from one key pair to another. Transition by way of an in-band process reliant on TAK objects is not mandatory for TAs or RPs, though the fact that the TAK objects are verifiable by way of the currently trusted TA public key is a benefit compared to existing out-of-band mechanisms for TA public key distribution.

There will be a period of time where both the current public key and the successor public key are available for use, and RPs that are initialised at different points of the transition process or from different out-of-band sources may be using either the current public key or the successor public key. TAs are required to ensure, so far as is possible, that there is no difference which public key is used for the purposes of RPKI validation.

11. IANA Considerations

11.1. Content Type

IANA has registered an OID 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 2.1

Table 1

Description: id-ct-signedTAL

OID: 1.2.840.113549.1.9.16.1.50

Specification: Section 2.1

11.2. Signed Object

IANA has added the following to the "RPKI Signed Objects" registry:

Name	OID	Reference
Trust Anchor Key	1.2.840.113549.1.9.16.1.50	Section 2.1

Table 2

IANA has also added 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.

11.3. File Extension

IANA has added the following item for the Signed TAL file extension to the "RPKI Repository Name Schemes" registry created by [RFC6481]:

Filename Extension	RPKI Object	Reference
.tak	Trust Anchor Key	RFC 9691

Table 3

11.4. Module Identifier

IANA has registered an OID 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	RFC 9691

Table 4

Description: RPKISignedTrustAnchorList-2021

OID: 1.2.840.113549.1.9.16.0.74

Specification: RFC 9691

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

IANA has registered 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 9691 for further information.

Interoperability considerations: N/A

Published specification: RFC 9691

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 9691.
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

12. References

12.1. 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>.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November 2003, <https://www.rfc-editor.org/info/rfc3629>.

[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>.

[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>.

[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>.

[RFC9110] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", STD 97, RFC 9110, DOI 10.17487/RFC9110, June 2022, <https://www.rfc-editor.org/info/rfc9110>.

[RFC9286] Austein, R., Huston, G., Kent, S., and M. Lepinski, "Manifests for the Resource Public Key Infrastructure (RPKI)", RFC 9286, DOI 10.17487/RFC9286, June 2022, <https://www.rfc-editor.org/info/rfc9286>.

[X.690] ITU-T, "Information technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", ITU-T Recommendation X.690, ISO/IEC 8825-1:2002, 2002, <https://www.itu.int/rec/T-REC-X.690-200207-S/en>.

12.2. 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>.

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>
```

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.

Authors' Addresses

Carlos Martinez
LACNIC
Rambla Mexico 6125
11400 Montevideo
Uruguay
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
RIPE NCC
Stationsplein 11
Amsterdam
The Netherlands
Email: tim@ripe.net
URI: https://www.ripe.net/

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