Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 9, 2020

T. Bruijnzeels
NLnet Labs
C. Martinez
LACNIC
R. Austein
Dragon Research Labs
July 8, 2019

RPKI Signed Object for Trust Anchor Keys
draft-ietf-sidrops-signed-tal-03

Abstract

Trust Anchor Locators (TALs) [I-D.ietf-sidrops-https-tal] are used by Relying Parties in the RPKI to locate and validate Trust Anchor certificates used in RPKI validation. This document defines an RPKI signed object for Trust Anchor Keys (TAK), that can be used by Trust Anchors to signal their set of current keys and the location(s) of the accompanying CA certiifcates to Relying Parties, as well as changes to this set in the form of revoked keys and new keys, in order to support both planned and unplanned 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 January 9, 2020.

Copyright Notice

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

Trust Anchor Locators (TALs) [I-D.ietf-sidrops-https-tal] are used by Relying Parties in the RPKI to locate and validate Trust Anchor (TA) certificates used in RPKI validation. However, until now there has been no formal way of notifying Relying Parties (RP) of updates to a TAL. Such updates may be needed in particular in case a Trust Anchor needs to perform a planned, or unplanned, key roll.

This document defines a new RPKI signed object that can be used to document the current set of keys and the location(s) of the accompanying CA certificates, as well as any changes to this set. This allows RPs to be notified automatically of such changes, and enables Trust Anchors to pre-stage a number of operational keys so that planned and unplanned key rolls can be performed without risking the invalidation of the RPKI tree under the TA. We call this object the Trust Anchor Keys (TAK) object.

When Relying Parties (RPs) are first bootstrapped, they use any current TAL to discover a key and location(s) of the TA certificate(s) for a TA. The RP can then retrieve and validating the TA certificate, and subsequently validate the manifest [RFC6486] and CRL [section 5 of @!RFC6487]. However, before processing any other objects it will then first validate the TAK object, if present. All enumerated new keys (and locations) are then added to a new list of current TA keys for this TA. The RP will then recursively fetch and validate the TA certificates, manifest, CRL and TAK objects for each of these keys. As a part of this process the RP will also compile a list of revoked keys enumerated by any of the validly signed TAK objects. As the final step the RP will then filter out any revoked TA keys from its new set. This new set now replaces the previous set.

This process allows Trust Anchors to operate a set of N current keys, where any key can effectively revoke any or all of the other keys to perform either a planned, or an unplanned, key roll. This also allows Trust Anchors to produce long lived TAK objects as forward pointers to RPs, and retire its old key when doing a key roll. While the generic process is quite involved, the amount of work needed to support an envisioned normal key roll is fairly limited. Under normal circumstances a TA will typically have two current keys, so that is can perform an emergency roll over in case one of the keys is lost. This means that the RP will need to validate one additional CA certificate, a CRL, a manifest and two TAK objects.

When a key roll is executed a TA will remove one old key, and introduce one new (back-up) key. The RP will remove the old key from

its set, and it will not be queried again, and it will add the new key and its TA certifcate location(s).

Only in a situation where an RP is very outdated can it be expected that the RP will have to discover several chained TAK object. But, since it will remove the outdated TALs in this process, this presents a one time cost only.

3. TAK Object definition

The TAK object makes use of the template for RPKI digitally signed objects [RFC6488], which defines a Crytopgraphic Message Syntax (CMS) [RFC5652] wrapper for the Signed TALs 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 defined 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 defined in Section 3.2.

- Additional steps to the validation steps specified in [RFC6488] required to validate the TAK, defined in Section 3.3.

3.1. The TAK Object Content Type

This document requests an OID for TAK objects as follows:

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

```
TAK ::= SEQUENCE {
   version   INTEGER DEFAULT 0,
   current   ::= SEQUENCE SIZE (1..MAX) OF CurrentKey,
   revoked   ::= SEQUENCE OF SubjectPublicKeyInfo
}

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

CertificateURI ::= IA5String

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

3.2.1. version

The version number of the TAK object MUST be 0.

3.2.2. current

This field defines the set of current keys (CurrentKey) according to the signer of this Signed TALs object.

3.2.2.1. CurrentKey

This field defines a current TA Key, equivalent to [I-D.ietf-sidrops-https-tal]. This structure contains a sequence of one or more URIs and a SubjectPublicKeyInfo.

3.2.2.1.1. certificateURIs

This field is equivalent to the URI section in section 2.1 of [I-D.ietf-sidrops-https-tal]. 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].

3.2.2.1.2. subjectPublicKeyInfo

This field contains a SubjectPublicKeyInfo [section 4.1.2.7 or @!RFC5280] in DER format [X.690].

3.2.3. revoked

This field contains the list of keys, identified by SubjectPublicKeyInfo, that are no longer to be used according to the signer of this document.

3.3. TAK Object Validation

To determine whether a TAK object is valid, the RP MUST perform the following steps 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 Trust Anchor CA certificate.
- This Trust Anchor 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]). In case more than one TAK object is found, all such objects MUST be considered invalid.
- 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.

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

4. TAK Object Generation and Publication

A TA MAY choose to use TAK objects to communicate its set of current, and revoked keys. If a TA chooses to use TAK objects, then it SHOULD generate and publish TAK objects under each of its current keys. An exception to this rule exists when a TA has lost permanent access to one of its keys or the accompanying repository publication point. In such cases however, the key in question MUST be revoked as described below in Section 7.

A non-normative guideline for naming this object is that the filename chosen for the Signed TAL 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. Note that this is in-line with filename extensions defined in section 7.2 of [RFC6481]

In order to generate the TAK Objects, 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 Signed TAL 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 same set of current keys (see Section 3.2.2) MUST be included on each TAK object for each current key.
- The TAK object MUST include all revoked keys (see Section 3.2.3) that became revoked while the key signing the TAK in question was current.

5. Relying Party Use

Relying Parties MUST keep a record of all current keys for each configured Trust Anchor, as well as the URI(s) where the CA certificate for each of these keys may be retrieved. This record MAY be bootstrapped by the use of a pre-configured (and unsigned) TAL file [I-D.ietf-sidrops-https-tal], but it MUST be updated with authoritative signed information found in valid TAK objects found in subsequent validation runs.

When performing top-down validation RPs MUST first validate and process any TAK objects for each of its known current keys for a TA by performing the following steps:

- A CA certificate is retrieved and validated from the known URIs as described in sections 3 and 4 of [I-D.ietf-sidrops-https-tal].
- The manifest and CRL for this certificate are then validated first as described in [RFC6487] and [RFC6486].
- The TAK file, if present, is validated as described in Section 3.3.

For each valid TAK file thus found all current keys, i.e. SubjectPublicKeyInfo and URIs, are kept. If any previously unknown keys are added to the set of current keys, then they MUST also be processed as described above.

Once the TAK objects for all keys are processed the set of current keys and URIs for the TA is updated as follows: * All new current keys found on any valid TAK object are added to the set of current keys. * The set of URIs for each current key is replaced by the union of all URIs for this key found on all valid TAK objects. * Finally, any current key that matches any revoked key on any valid TAK object is removed from the set of current keys.

Note that if a current key does not occur on any valid TAK object, but it is not revoked either, then it and any previously known URIs for it are kept. Also note that if an RP was bootstrapped using a TAL file [I-D.ietf-sidrops-https-tal], the keys and URIs will now have been replaced by values found on TAK objects.

After this the RP can choose any one of the valid CA certificates for any key that is still in the set of current keys for this TA, in order to continue the top-down validation of object for this TA as described in [RFC6487].

6. Maintaining multiple TA keys

If a TA operates multiple keys, then 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 space would lead to confusion at best, and in case of file name collisions of objects invalidity.

However, the CA certificates for each key, and the contents published by each key MUST be equivalent. In other words it MUST not make a

difference which of the keys is used as a starting point for top-down validation by RP software.

This means that the IP and AS resources contained on all current CA certificates for the current 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 Certification Authorities.

If a TA uses a single remote publication server for its keys using the RPKI publication protocol [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 TA keys repository publication points have been updated. Given that TA key operations are normally performed infrequently we don't expect that this is a problem. I.e. if the revocation or shrinking of an issued CA certificate is staged for days, or weeks anyway, then experiencing a delay of several minutes for the repository publication points to all be updated is fairly insignificant.

- In cases where a CA certificate is replaced by a certificate with an extend set of resources the TA MUST inform the receiving CA only after all 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' value on certificates issued by the child CAs may not match (section 4.8.7 of [RFC6487]). However, this information is not considered critical for validation of these objects and provided as hints to RP software

only. Therefore RP software MUST NOT reject these certificates based on a mismatch of this value.

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 change to having two current keys at all times allowing them to perform both planned and unplanned key rolls.

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

Before adding any new keys a Trust Anchor may want to build up operational experience in maintaining a TAK object that describes its current key only. We will call refer to this key as key 'A' throughout this section.

The TA will have a TAL file [I-D.ietf-sidrops-https-tal] that contains one or more URIs where the (equivalent) CA certificates for this key 'A' can be retrieved. The TA can now generate a TAK objects that includes key 'A' only in its sequence of 'CurrentKey' values.

The TA SHOULD publish the CA certificate for key 'A' at one or more new locations not used in the TAL file, and use these new URIs in the TAK object. The TA is free to choose any naming strategy for these locations. As a non-normative suggestion, one such approach could be to use the date that this phase was started as part of the file name or a directory where the CA certificate is published.

The TA can now monitor the retrieval of its CA certificates from the URI(s) in the newly published TAK object, relative to the retrieval from the URI(s) listed in its TAL file, to learn the proportion of RPs that can successfully validate and use the TAK object.

7.2. Phase 2: Add a Key 'B'

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

At this point, the TA can also issue a new TAL file [I-D.ietf-sidrops-https-tal] for key 'B', and test locally that the validation outcome for the new key is indeed equivalent to the other current key(s).

When the TA is certain that both keys are equivalent, it MUST issue a new TAK object under each of its current keys, and include both the old key 'A' and this new key 'B' in the set of current keys.

The TA SHOULD now also release a new TAL file for this new key 'B' as the intended new key to be used by RP software. However, as described above, it SHOULD use a different set of URIs in the TAL compared to the TAK file, so that it can learn the proportion of RPs that can successfully validate and use the updated TAK objects.

7.3. Phase 3: Roll to Key 'C'

In this phase a new key, key 'C' is generated as described above in Section 7.2. And one of the previous keys is revoked.

7.3.1. Planned Direction Roll

If the key roll is planned, and the TA has access to all its keys 'A', 'B' and 'C', and the publication servers for each of the keys, then a new TAK object is generated for each of these keys listing keys 'B' and 'C' as current, and key 'A' as revoked.

The TA SHOULD now publish a long-lived TAK file, CRL and Manifest under key 'A', remove all other content, and destroy key 'A'. This way RP software that uses a TAL for key 'A' can still successfully find keys 'B' and 'C', and in future 'D', 'E', etc.

If access to key 'A' was lost, then the process is slightly different. The TAK object for key 'A' cannot be updated and will therefore still refer to keys 'A' and 'B' as the current keys, and include no revocations. However, an updated TAK object listing keys 'B' and 'C' as current, and listing key 'A' as revoked can still be issued and published under keys 'B' and 'C'. As described in Section 5 RPs will then discover that key 'A' is revoked, and continue to use keys 'B' and 'C'.

7.3.2. Unplanned Direction Roll

If key 'B' is compromised, the process is similar to above, except of course that now keys 'A' and 'C' are included in the set of current keys, and key 'B' is in the set of revoked keys. If the TA still has access to key 'B', then it SHOULD publish a long-lived TAK file, CRL and manifest for key 'B' and remove all other content for it. If it cannot perform this action then simply marking key 'B' as revoked will still notify RPs to disregard it.

7.4. Phase X: Roll to Key 'D', 'E', ..

Further key rolls are essentially no different the roll to key 'C' described in Section 7.3, except that there is no need to still include key 'A' in the list of revoked keys when the the roll to key 'D' is performed. RPs will already have learned to that key 'A' is revoked, before they learn about key 'D'.

8. Deployment Considerations

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

That said, the flagging mechanism introduced here can only be relied on once a majority of RPs support it. Defining when that moment arrives is by definition something that cannot be established at the time of writing this document. The use of unique URIs in TAK objects compared to their equivalent TAL files should help operators understand which proportion of RPs support this mechanism.

9. Security Considerations

It should be noted that because any key can revoke the other key(s), a risk introduced: if an adversary can gain access to one of the keys, and publication servers for it, then they can essentially take over a TA. It should also be noted that a TA can revoke all of its keys by accident and make itself obsolete.

However, these risks can be mitigated greatly by the use of Hardware Security Modules (HSM) by TAs, which will guard against theft of a private key, and operational processes to guard against (accidental) mis-use of the keys in an HSM by operators.

Although HSMs can help against key theft, the risk of key loss is still very applicable. In some ways more so, because back-ups are hard by design. Key loss can easily happen for example when an operator card set that is used to authorise use of a key in an HSM can no longer be used, e.g. because cards are broken or lost, or a persons who holds a card is sadly no longer with us, or passwords are forgotten, etc.

In such cases the ability to perform an unplanned roll as described in this document will be very useful, provided that access to the both keys is arranged differently, and the issues affecting one key, do not necessarily affect the other key.

An example where the planned rolls are useful is when a TA is using an HSM from vendor X, and they want to migrate to an HSM from vendor Y.

10. IANA Considerations

10.1. OID

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

Decimal	Description	References
TBD	Trust Anchor Keys	[section 3.1]

10.2. File Extension

IANA is 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 Keys	[this document]

11. Security Considerations

TBD

12. Acknowledgements

The authors wish to thank Martin Hoffmann for a thourough review of this document.

13. References

13.1. Normative References

[I-D.ietf-sidrops-https-tal]
Huston, G., Weiler, S., Michaelson, G., Kent, S., and T. Bruijnzeels, "Resource Public Key Infrastructure (RPKI) Trust Anchor Locator", draft-ietf-sidrops-https-tal-08 (work in progress), April 2019.

[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>.

[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>.

[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.

13.2. 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>.

Authors' Addresses

Tim Bruijnzeels
NLnet Labs

Email: tim@nlnetlabs.nl
URI: https://www.nlnetlabs.nl/

Carlos Martinez
LACNIC

Email: carlos@lacnic.net
URI: https://www.lacnic.net/

Rob Austein
Dragon Research Labs

Email: sra@hactrn.net