Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 19, 2019

T. Bruijnzeels
NLnet Labs
C. Martinez
LACNIC
R. Austein
Dragon Research Labs
October 16, 2018

RPKI Signed Object for Trust Anchor Keys
draft-ietf-sidrops-signed-tal-02

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.

Table of Contents

1. Requirements notation 3
2. Overview 3
3. TAK Object definition 4
3.1. The TAK Object Content Type 5
3.2. The TAK Object eContent 5
3.2.1. version 5
3.2.2. current 5
3.2.3. revoked 6
3.3. TAK Object Validation 6
4. Maintaining multiple TA keys 7
4.1. Prepare a new TA key 7
4.2. Publishing for Multiple TA Keys 7
5. TAK Object Generation and Publication 8
6. Performing TA Key Rolls 9
6.1. Opting in to Key Rolls 10
6.1.1. Trust Anchor 10
6.1.2. Relying Parties 12
6.2. Pre-stage a New Key 12
6.2.1. Trust Anchor 12
6.2.2. Relying Parties 14
6.3. Planned Key Revocation 14
6.3.1. Trust Anchor 14
6.3.2. Relying Parties 17
6.4. Unplanned revocation 17
6.4.1. Trust Anchor 17
7. Deployment Considerations 18
8. IANA Considerations 18
8.1. OID 18
8.2. File Extension 19
9. Security Considerations 19
10. Acknowledgements 19
11. References 19
11.1. Normative References 19
11.2. Informative References 21
Authors' Addresses 21

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 loctaion(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 retreive 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 enumarated 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 enumarated 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.

If the key used to start this process is still considered current, then validation continues. But if the key was revoked, then validation is restarted using one of the remaining keys in the 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 two TAK objects. However, typically these files will agree that both keys are current and validation continues.

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.

Note that in theory a TA can revoke all of its keys and make itself obsolete. In practice however, a well operated TA will have measures in place to prevent this. Furthermore they can protect themselves against key loss to adversaries through the use of such as the use of a Hardware Security Module (HSM) to protect keys. Protecting against this mis-operation would incur complexity and guesswork on the RPs. Therefore it is believed that it is best to keep the process straightforward, and offer a solution for the more likely issues of loss of a key, e.g. because an HSM or card set is broken, and planned key rolls.

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. Maintaining multiple TA keys

As described in Section 6 a TA will most likely choose to operate two keys at any one time in order to be prepared for an emergency key roll. When a TA operates multiple keys, each key MUST use its own CA repository publication point as described in [RFC6481]. The CRL and Manifest [RFC6486] for each of these keys will be unique to each key, but the TA MUST ensure that equivalent CA certificates and RPKI signed objects are issued under each key. Note that this is similar to how such certificates and RPKI signed objects are re-issued as part of a lower level CA key roll, described in section 4 of [RFC6489].

4.1. Prepare a new TA key

The Trust Anchors MUST generate a new key pair and generate a new TA Certificate. For the Subject Information Access (see section 4.8.8.1 of [RFC6487]) this MUST use URIs that will be used by the new key to publish objects. These URIs MUST be uniqe for use by this new key only. The Internet Number Resources on this new certificate MUST be equivalent to those found on the current certificate.

The new TA certificate MUST be published under one or more new Certificate URIs for use by this new key only.

As decribed above, the TA MUST issue and publish equivalent CA certificates and RPKI signed objects under this new key.

It is RECOMMENDED that the TA now generates a new TAL [I-D.ietf-sidrops-https-tal] and verifies that the new Trust Anchor certificate can be retrieved from all locations, and that it generates the same results when it is used for top-down validation instead of (any of) the current TA key(s).

Note that the TA MAY choose to make this TAL available to Relying Parties, in particular to those that do not support TAK objects, and for inclusion in the distribution of RP software in order to minimise the overhead in bootstrapping fresh installations.

4.2. Publishing for Multiple TA Keys

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:

- It is strongly RECOMMENDED that TAs do not issue any RPKI Signed Objects, such as ROAs [RFC6482], but limit their operations to maintaining a CRL, Manifest and CA certificates only. If an organisation maintaining a TA has an operational need for such objects then it is strongly RECOMMENDED that they operate a separate non-TA CA as a child of their TA for these operations. If this approach is used the remaining issues regarding temporary inconsistencies between multiple TA key repository publication points is greatly reduced.

- In cases where a CA certificate is revoked completely, or replaced by a certifcate 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 certifcate 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 receiveing 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.

5. 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 6.

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:

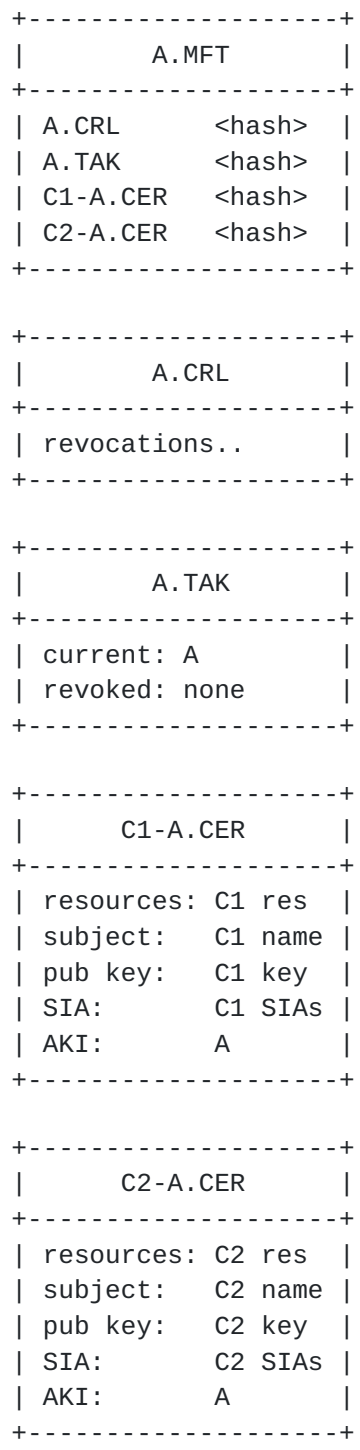
- The TA MUST generate a key pair for a "one-time-use" EE certificate to use for the TAK
- The TA MUST generate a one-time-use EE certificate for the TAK
- This EE certificate MUST have an SIA extension access description field with an accessMethod OID value of id-ad-signedobject, where the associated 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.

6. Performing TA Key Rolls

6.1. Opting in to Key Rolls

6.1.1. Trust Anchor

For simplicitly let's start with a situation where a TA has only one key. The TA wants to start using TAK objects to perform key rolls in future, so it introduces a TAK object under its single key 'A'. The repository structure looks as follows (irrelevant details omitted):



So, the TA publishes a CRL and MFT under its key A, listing a TAK object and in this case two certificates issued to children 'C1' and 'C2' signed using key A. The TAK object lists key 'A' as the only current key, and has no revoked keys.

6.1.2. Relying Parties

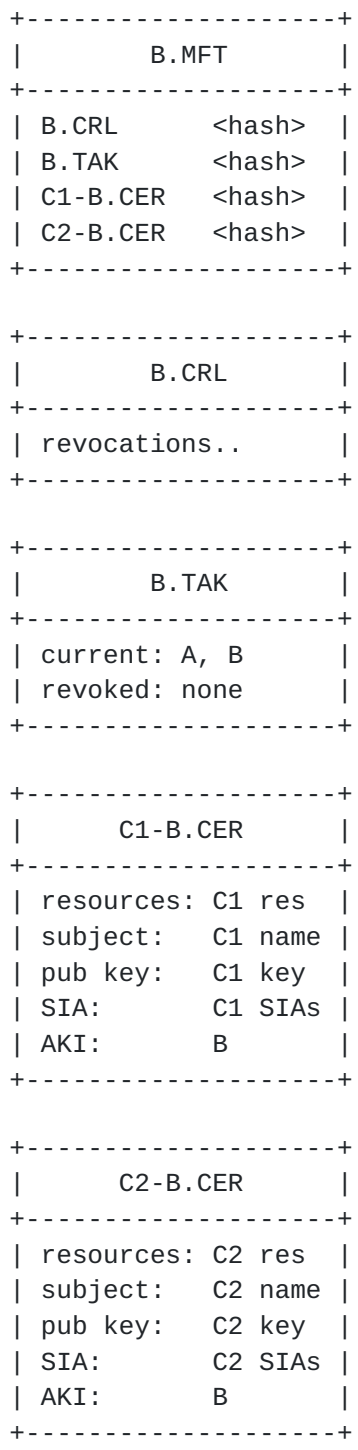
Relying Parties who have a TAL for key 'A' configured will discover the TAK object. If the RP does not support this object, it will reject this object but continue to validate the remaining RPKI tree as usual. If the RP does support TAK objects it will conclude that key 'A' is the one and only current key, and will proceed to validate the remaining RPKI tree as usual.

6.2. Pre-stage a New Key

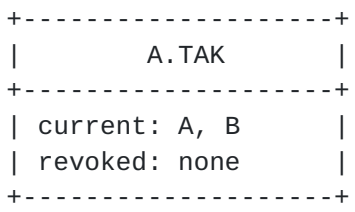
6.2.1. Trust Anchor

Now the TA prestages a new key 'B' and produces equivalent CA certificates for children 'C1' and 'C2', i.e. the resources, subject name, public key and SIA etc are all equivalent, but these certificates are signed under key 'B'. (See Section 4 for a more thorough description of this). The TAK object for key 'B' recognises both keys 'A' and 'B' as current.

The repostory structure and TAK object for key B are then as follows:



When the TA is certain that the content for key 'B' is correct, it can also update the TAK object for key 'A' to include 'B':



One way to do this is by generating a TAL [I-D.ietf-sidrops-https-tal] for key B and verifying that validation using this yields the same results as validation using the TAL for key A would. However, note, that it is preferred that this is done as part of an automated process that is sufficiently well tested, and that the contents of the repositories for keys 'A' and 'B' are updated as a single delta if the publication protocol [RFC8181] is used (see also: Section 5).

6.2.2. Relying Parties

Relying Parties who have a TAL for key 'A' configured will discover the TAK object. If the RP does not support this object, it will reject this object but continue to validate the remaining RPKI tree as usual. If the RP does support TAK objects it will conclude that there are now two keys 'A' and 'B', and no revoked keys that it should be aware of. Since key 'A' is still current, the RP will continue to validate the RPKI tree structure using the repository for key 'A', ignoring the non-TAK objects in the repository for key 'B'.

The result will be the same for Relying Parties who have a TAL for key 'B' configured, because both keys are equivalent at this time.

6.3. Planned Key Revocation

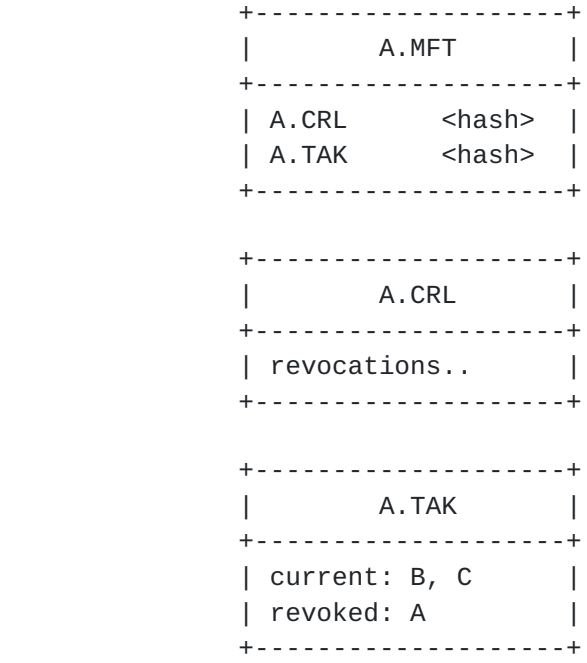
6.3.1. Trust Anchor

The TA has now decided that key 'A' must be revoked. It still has access to this key and the repository, so it can perform a planned key roll. In addition to revoking key 'A', the TA will also generate new key 'C' to ensure that it has at least two current keys at all times for redundancy.

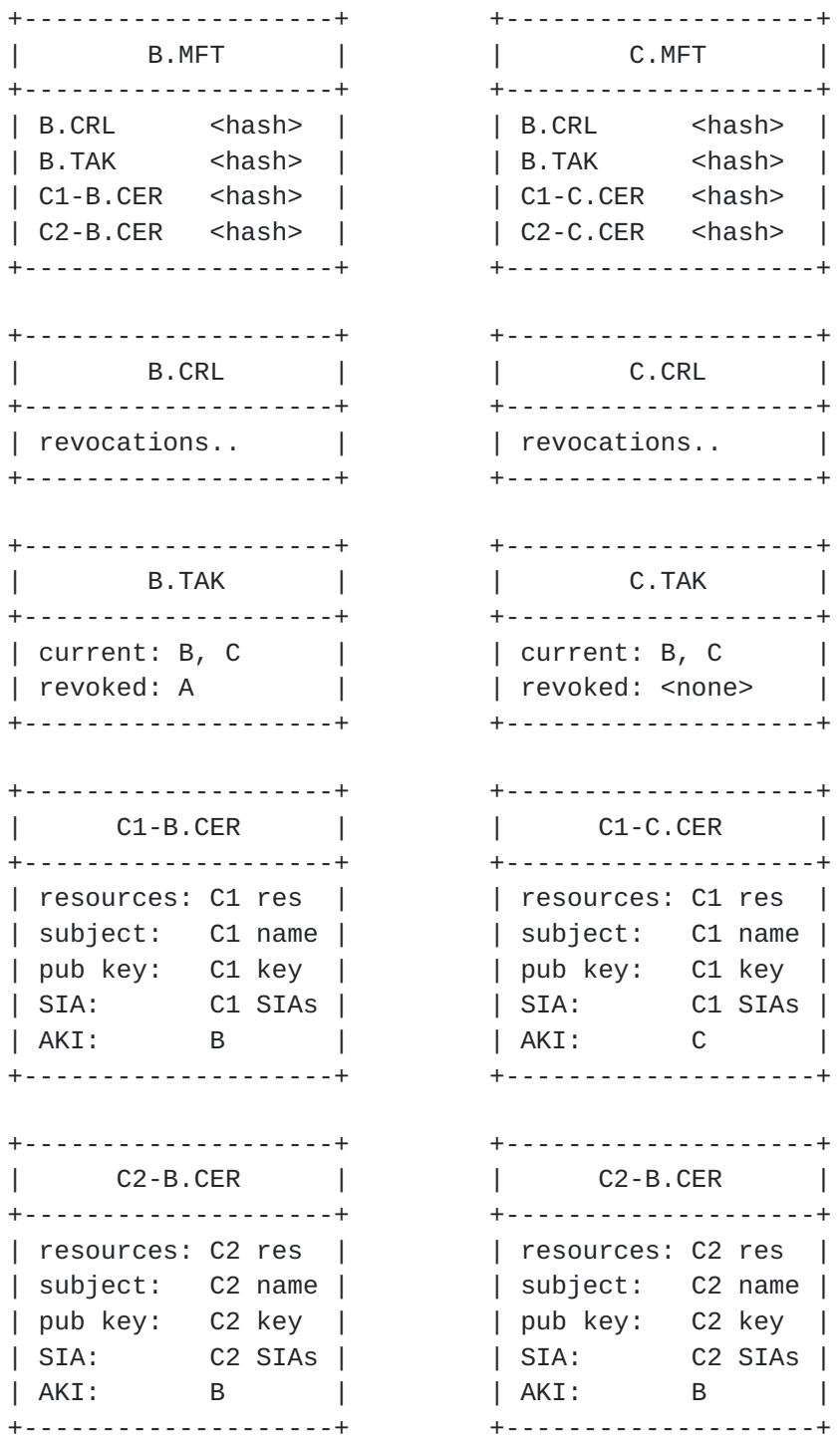
Keys 'B' and 'C' will become current keys on the TAK objects for all keys: 'A', 'B' and 'C'. Key 'A' will become part of the revoked keys on the TAK objects for keys 'A' and 'B'. Note that it is not needed to list key 'A' as revoked on the TAK file for key 'C', because RPs will only learn about key 'C' at the same time as learning about the revocation of key 'A' (see also below).

The TA will publish a long-lived TAK file and MFT and CRL only for key 'A' and publish these objects as waypointers for RPs that have a TAL pointing at key 'A' before destroying key 'A'.

The resulting structure for key 'A' will be as follows:



The resulting structures for keys 'B' and 'C' will be as follows:



In addition to this the TA SHOULD reach out to RP vendors so that they can update the TAL included in the RP software distribution to use key 'B'.

6.3.2. Relying Parties

Relying Parties who have a TAL for key 'A' configured will discover the TAK object. If the RP does not support this object, it will reject this object but continue to validate the remaining RPKI tree as usual. In this case that means that validation will stop, because there are no more objects under key 'A'. Therefore it is important that RPs that do not support TAK files are updated to use the TAL for key 'B' through some other process.

If the RP uses a TAL for key 'A' and it supports TAK objects, it will discover that the TAL for key 'A' has keys 'B' and 'C' as current, and revokes itself. It will then proceed to process keys 'B' and 'C' and find TALs which list the same current keys. So, it will now replace its notion of the current key set for this TA based on its TAL (key 'A') with what it learned. To keep things simple the RP will now conclude that it should re-start validation using a remaining current key, in this case key either 'B' or 'C' may be used.

If the RP already had a TAL for key 'B' and it supports TAK objects, or it simply started with key 'B' because it added it to its set of current keys when this key was pre-staged (see Section 6.2), it will learn that key 'A' is revoked and therefore will not attempt to verify the TAK file for key 'A'. It will also learn about key 'C' and inspect this key's TAL, and discover that only keys 'B' and 'C' are considered current. Since it started the validation process with a key that is still current, it can proceed to validate the RPKI tree using the repository under key 'B'.

6.4. Unplanned revocation

6.4.1. Trust Anchor

Now keys 'B' and 'C' are current. The TA may have intended to revoke key 'B', essentially rolling over to key 'C' and a new key 'D', but let us suppose that the TA lost access to key 'C'. In this case the TA will simply revoke key 'C' instead, and still introduce a new key 'D'.

The major difference with the process described above for planned rolls, is that now the TA will not be able to update the TAK object, MFT or CRL for key 'C'. However, because all TAL objects for current keys are evaluated before tree validation is performed, it is safe to leave these objects in a repository. Keys 'B' and 'D' will simply mark key 'C' as being revoked.

If an RP still has a TAL pointing at key 'C' it will discover that key 'D' is added, and that key 'B' has been revoked through the TAK object published for keys 'B' and 'D'. At least, as long as the the MFT and TAK EE certificates have not expired, and the CRL and MFT are not stale.

If the TA is absolutely sure that the TAL for key 'C' never shipped with any RP distribution, then it would also be safe to delete the repository key 'C' altogether. RPs will learn that 'C' is revoked, and therefore will not even attempt to download the TAK object. However, it is hard to be certain of this and there this is NOT RECOMMENDED.

7. 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. Until such time, TAs SHOULD continue to generate unsigned TAL files [I-D.ietf-sidrops-https-tal], and indicate which should be considered their current TAL, and communicate them to RPs through other means.

However, once a majority of RPs support this mechanism it would be RECOMMENDED that Trust Anchor operators perform key rolls regularly. The most assured way to know that such key rolls will work is by making them a part of normal operations. Determining when this moment arrives is by definition out of scope for this document, as it should be based on operational experience.

8. IANA Considerations

8.1. OID

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

Decimal	Description	References
TBD	Trust Anchor Keys	[section 3.1]

8.2. 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]

9. Security Considerations

TBD

10. Acknowledgements

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

11. References

11.1. 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-05 (work in progress), October 2018.

[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>.

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

[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>.

[RFC6489] Huston, G., Michaelson, G., and S. Kent, "Certification Authority (CA) Key Rollover in the Resource Public Key Infrastructure (RPKI)", BCP 174, RFC 6489, DOI 10.17487/RFC6489, February 2012, <https://www.rfc-editor.org/info/rfc6489>.

[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.

11.2. Informative References

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <https://www.rfc-editor.org/info/rfc5652>.

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