RPKI Signed Checklists

Job Snijders
~~NTT~~ Fastly
job@fastly.com

What are we trying to solve?

Same thing as *draft-ietf-sidrops-rpki-rta* set out to solve, but with a slightly different (simpler) CMS object profile.

The Internet network operator community would benefit from an IETF standardized signed object profile for an industry-wide understood RPKI-based attestation mechanism.

The author expects to be able to automate the following inter-op examples:

- Associating PeeringDB.com user accounts to Internet Number Resources
- Improved operations for BYOIP workflows (such as Amazon EC2's BYOIP)
- Out-of-band distribution of Ghost Buster Record-style information

Differences between RTA and RSC

RTA suggests multiple signers subordinate to multiple different Trust Anchors can together composite a robust CMS signed object. A complication I foresee is handling AKI/SKI collisions under different Trust Anchors.

For example (in Fastly) each Trust Anchor exists in separate *X509_STORE's* in physically different memory address space. There are no industry-recognized error-codes for "A common Trust Anchor is lacking", a "multiple TA" RTA just won't pass validation with no robust recovery strategy.

RTA suggests intermediate CA & CRLs can be included as a convenience to the Relying Party. However I fear this discourages Relying Parties to attempt to confirm whether they have the latest CRL or not. My fear also is that some RPs might end up accidentally accepting self-signed root certificates which don't chain up to any locally configured TAs.

The RTA model appears to encourage "sign with all your resources", whereas the RSC model invites operators to sign with "only the relevant resource". Anticipation of operator perception is of course highly subjective!

All in all - conceptually RTA and RSC are very similar, but some implementation details differ.

Implementation status

Signer #1: An example RSC object *example/checklist.sig* is available at

https://github.com/job/draft-rpki-checklists/tree/main/example

The example chains up to the current RIPE NCC Trust Anchor

Signer #2: APNIC provided a signer https://github.com/APNIC-net/rpki-rsc-demo

Validator #1: A work-in-progress development branch exists in the *rpki-client* project

Validator #2: APNIC provided a validator https://github.com/APNIC-net/rpki-rsc-demo

Next step: IANA Early Allocation (to replace temp PEN OID)

```
Date: Tue, 09 Mar 2021 02:38:47 +0100
From: Job Snijders <job@fastly.com>
To: sidrops-chairs@ietf.org
Subject: request for IANA Early Allocation Of Codepoints procedure for
        draft-ietf-sidrops-rpki-rsc

Dear SIDROPS Chairs,

I'd like to submit a request for early allocation for codepoints for
'RPKI Signed Checklists' (draft-ietf-sidrops-rpki-rsc-01), following the
procedure outlined in RFC 7120.

Up until now implementers having been using an OID from the Private
Enterprise Number (PEN) space to test RPKI Signed Checklist
functionality. To prevent use of this PEN number 'in the wild' (as
consequence of of wider deployment), an early allocation of an OID might
be beneficial. There is no scarcity of space from which RPKI code points
can be allocated.

To foster interopability between current and future implementations, I
believe it would be helpful to now request IANA to add a temporary early
allocation to the "Resource Public Key Infrastructure (RPKI)" registry
in sub-registry "RPKI Signed Objects" at
http://www.iana.org/assignments/rpki/rpki.xhtml#signed-objects
aligned with the IANA Considerations in Section 9 of
draft-ietf-sidrops-rpki-rsc-01

The reference for the allocation should be
https://datatracker.ietf.org/doc/draft-ietf-sidrops-rpki-rsc

RFC 7120 indicates as next steps for working group chairs:
```

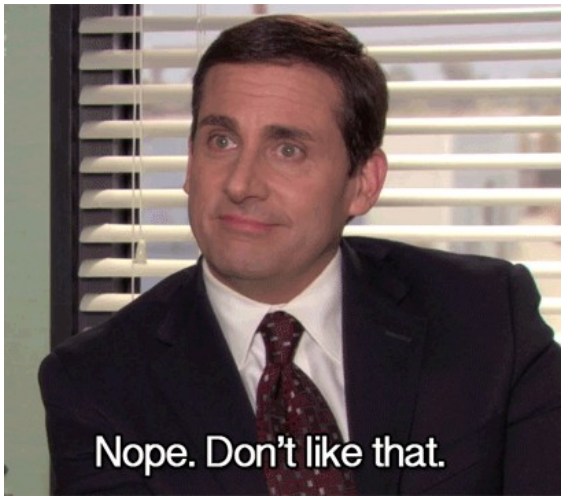
Question: I don’t like any of this

The OID is allocated from an ‘infinite’ code point space, there is no scarcity

The RSC .sig files do not add any load to the RPKI publication system, RSC are distributed “out-of-band”.

RSC CRL checks can piggyback on existing fetch operations performed by Relying Parties anyway.

Certificate Authorities are not mandated to produce RSCs



Identity vs Authorization

A Trust Anchor operator will always be able to produce valid RTA/RSCs (just like ROAs can appear elsewhere/higher up in the TA's tree).

Reminder: the Trust Anchor represents *assumed* trust, not *derived* trust.

RSC/RTA help establish that *someone* signed-off on a specific attestation, not *who* exactly signed off on it. The Relying Party trusts the Trust Anchor, no more is implied.

For the purpose of BYOIP, or PeeringDB account <> INR association, the above is considered sufficient to conduct business.

How to implement RSC support? Linking existing RPKI ideas

1 part GBR code (the code path towards the outer envelope)

1 part CRL checking (should come for free in CRL-aware implementations)

1 part RFC 3779 code (to parse the ResourceBlock in the RSC eContent)

1 part ROA code (the concept of the eContent matching specific EE extensions)

1 part Manifest eContent handling code (the file <> sha256 code)

.... 5 spoons of sugar, a lemon, dash of gin

All in all, because the RSC specification closely follows the RFC 6488 template, the cost of *implementing* and *maintaining* RSC validator code is reduced compared to RTA.

Questions?