RPKI Validation Re-Reconsidered

draft-ietf-sidrops-rpki-validation-update

Job Snijders <job@fastly.com>
Ben Maddison <benm@workonline.africa>

Terminology for reasoning about this

Assumed Trust
In the RPKI hierarchical structure, a Trust Anchor is an authority for which trust is assumed and not derived. Assuming trust means that violation of that trust is out-of-scope for the threat model.

Derived Trust
Derived Trust can be automatically and securely computed with subjective logic. In the context of the RPKI, trust is derived according to the rules for validation of RPKI Certificates and Signed Objects.

In other words

It is possible to define multiple deterministic validation algorithms for PKIs, like the RPKI.

Which algorithm is the right (or “correct”) algorithm is in the eye of the beholder!

The current algorithm is problematic

Defined in RFC 3779 section 2.3 and section 3.3; and RFC 6487 section 7.

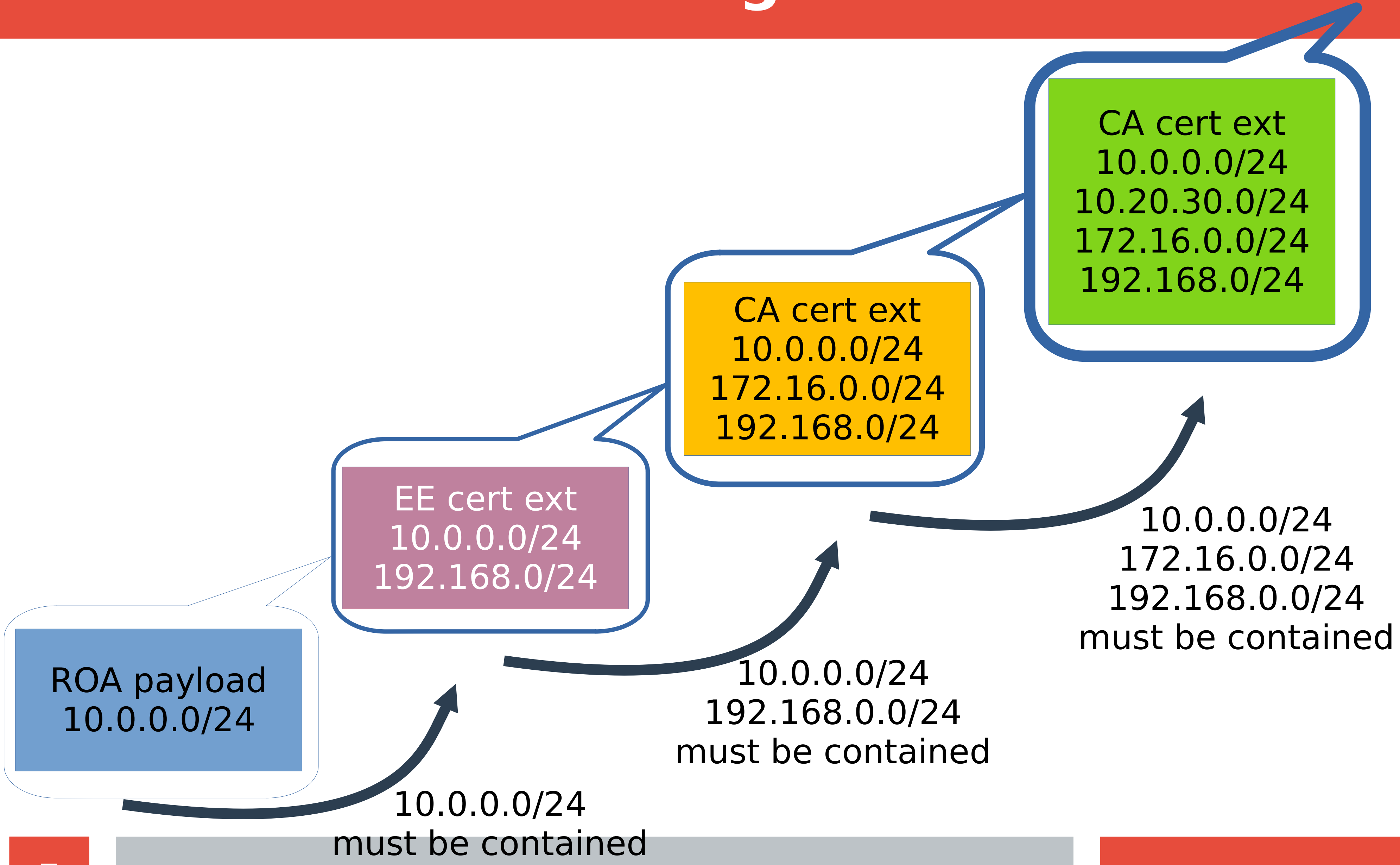
Number Resources *unrelated* to the ROA payload entry at hand also need to be contained (cumulatively).

Blast radius too big facing resource over-claiming.

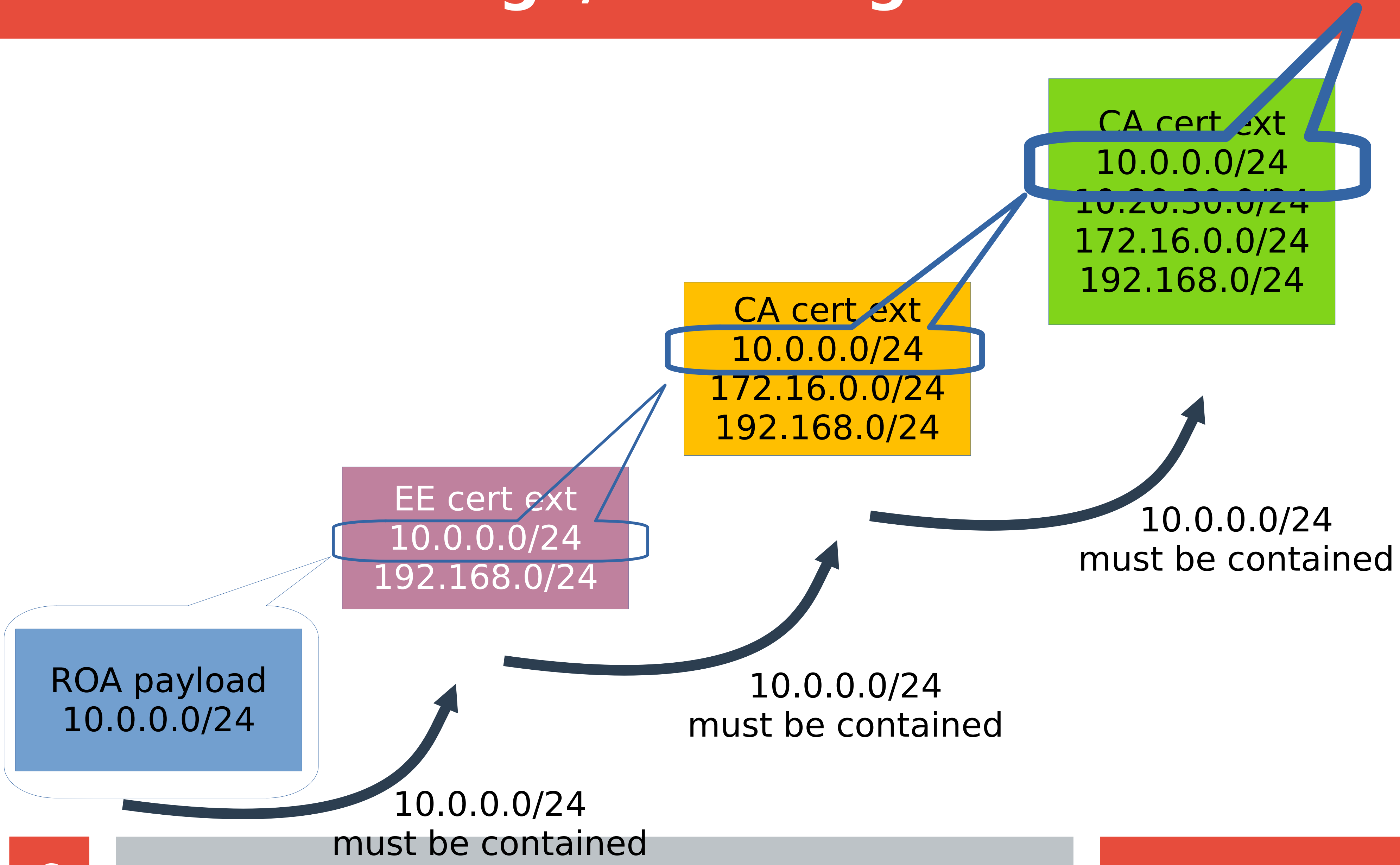
Lot of friction around inter-RIR/LIR transfers.

The 6487 outcome is <u>disproportional</u> in context of the RPKI

How the RFC 6487 algorithm works

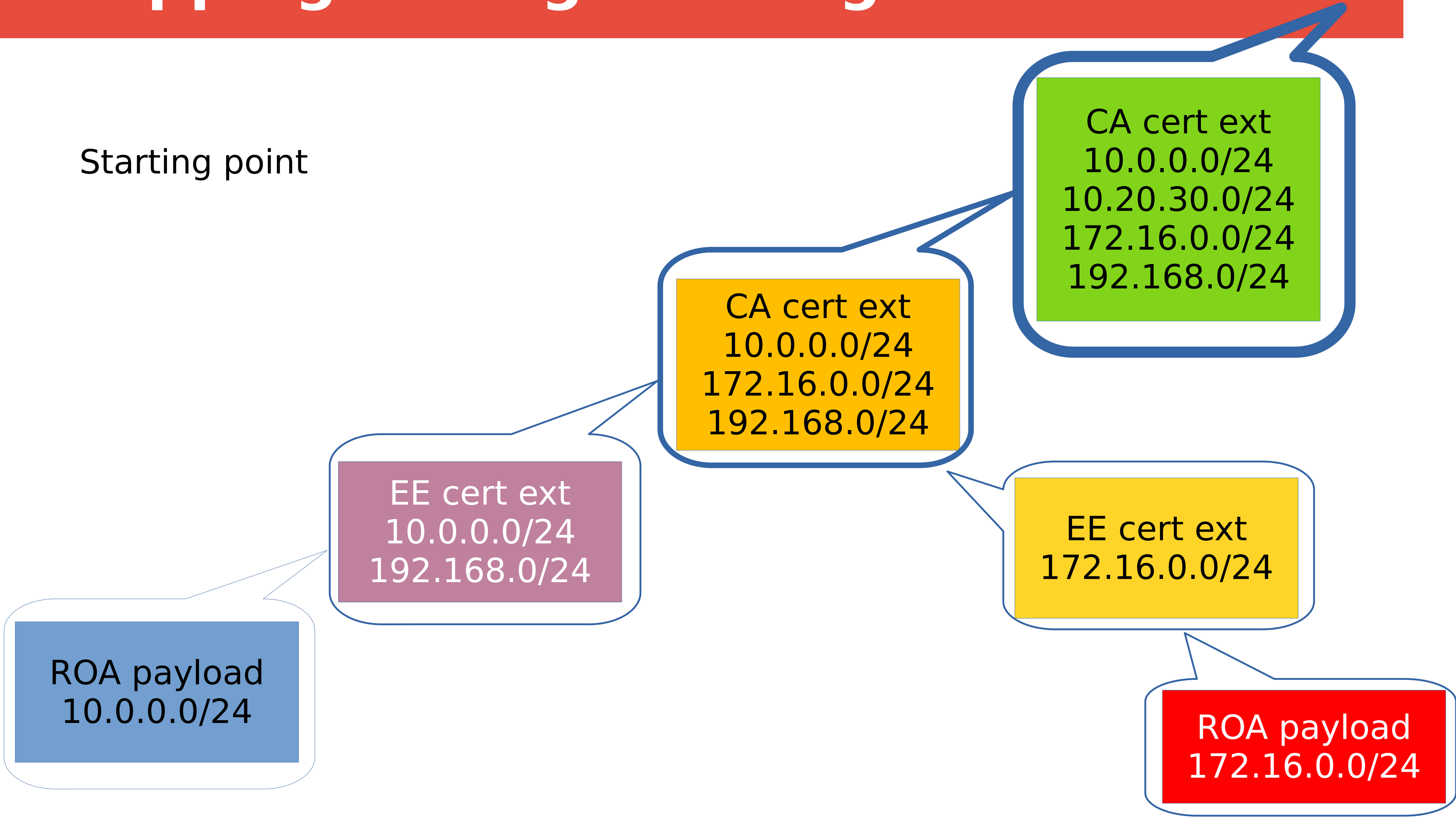


How the George/Geoff algorithm works



Stepping through the algorithms

Starting point

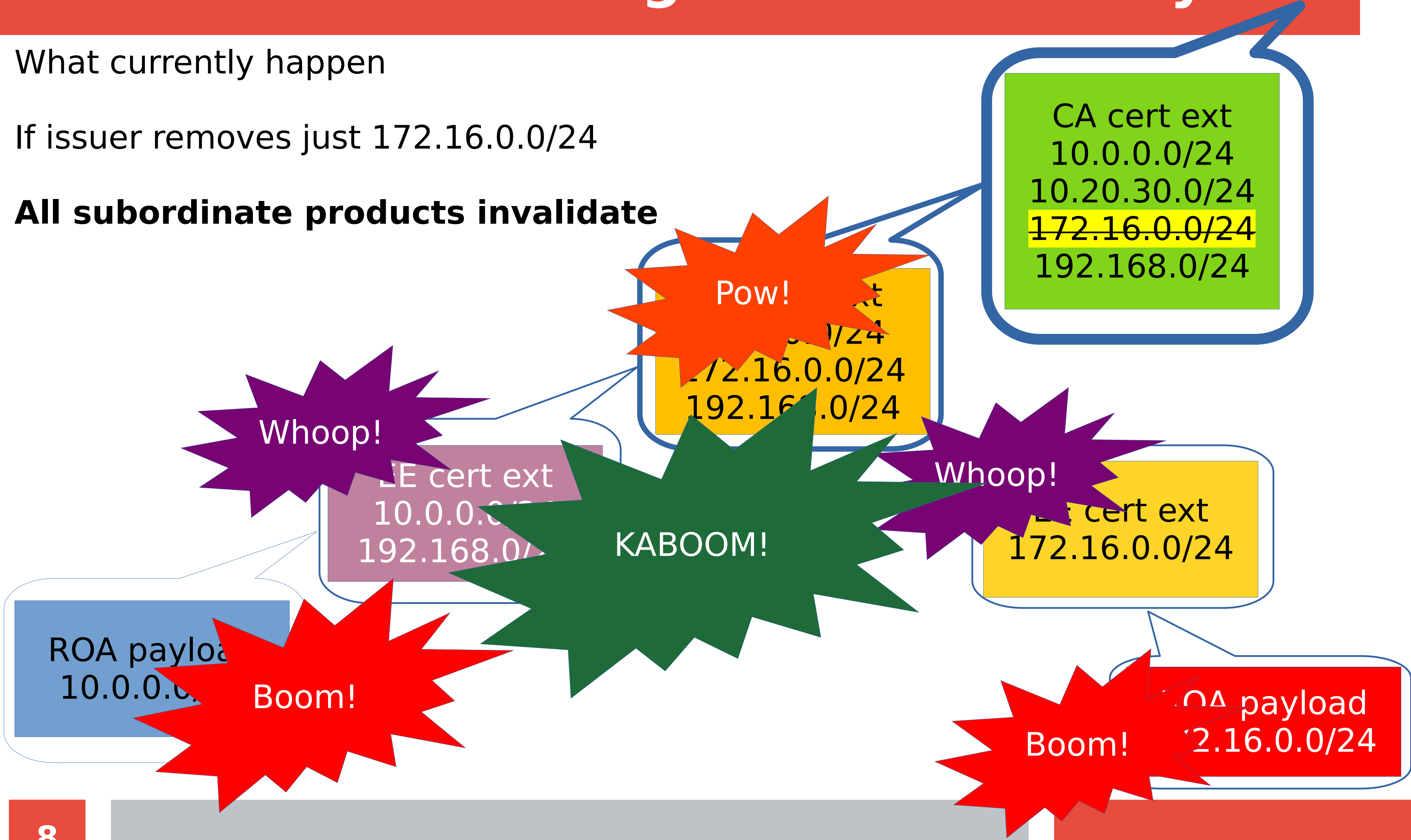


How the current algorithm is thorny

What currently happen

If issuer removes just 172.16.0.0/24

All subordinate products invalidate

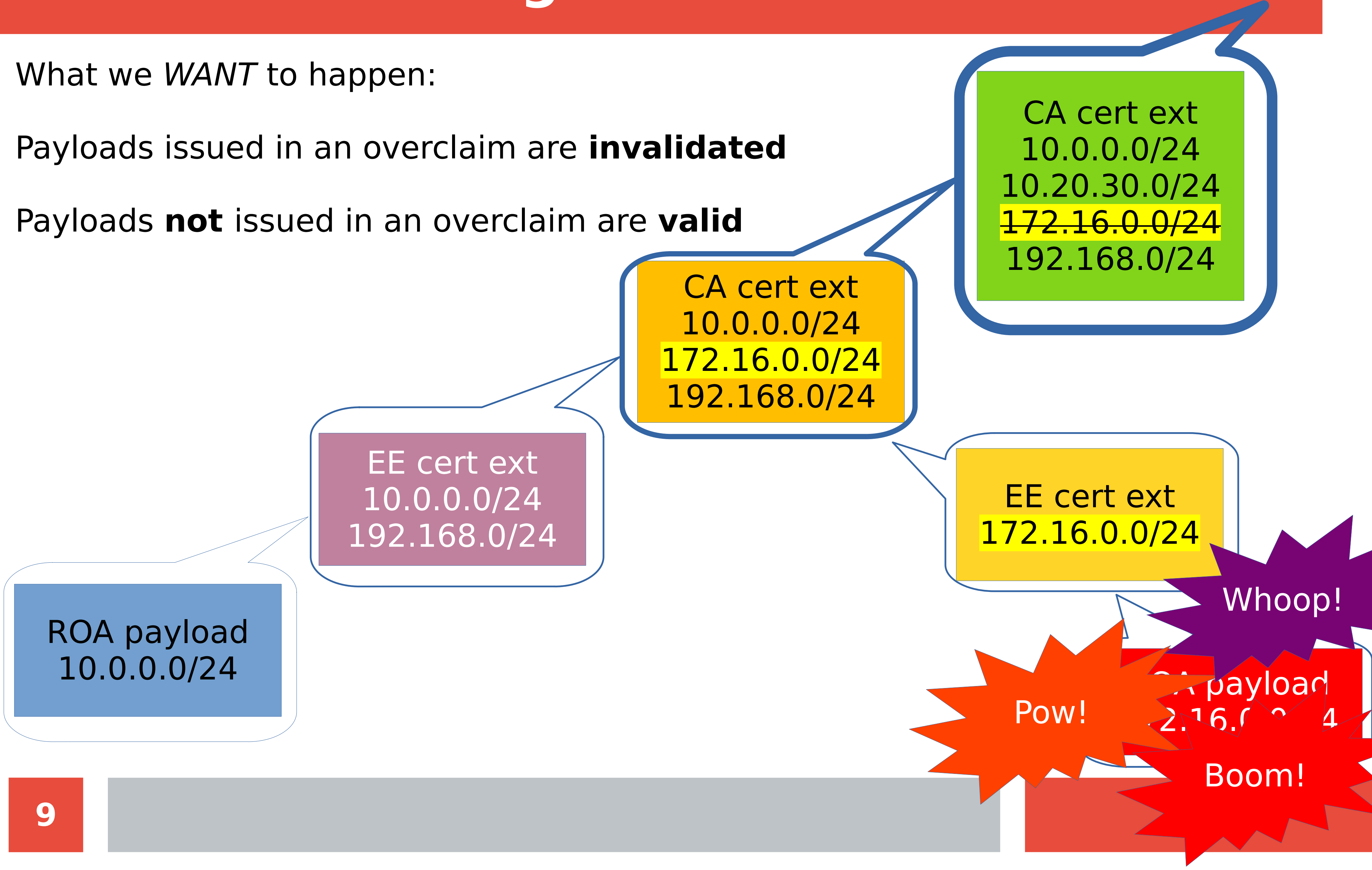


How the new algorithm is desirable

What we *WANT* to happen:

Payloads issued in an overclaim are **invalidated**

Payloads **not** issued in an overclaim are **valid**



The new algorithm is what we want

Proposed in RFC 8360 “Validation Reconsidered”

Number Resources *unrelated* to the VRP entry at hand, do not need to be contained.

Blast radius is precise and limited.

The new algorithm secures payloads how we want.

Resolves friction for inter-RIR/inter-LIR transfers.

Proportional outcomes in context of the RPKI

RFC 8360 is undeployable

“Validation Reconsidered” was imagined to work via new policy identifiers where CAs and RPs do a complicated dance.

RFC8360 is under-specified, things are missing, but adding text won’t solve the core issues.

The 8360 idea & algorithm are good, the execution plan is not feasible.

Path forward

Deprecate RFC 8360 & its code points

Do surgery to RFC 3779 & RFC 6487 to insert the robust validation algorithm.

All implementation effort is with RP projects, not CAs signers. RP projects seem onboard.

People that want this, it is time to speak up.

Next steps

- **Please *review* draft-ietf-sidrops-rpki-validation-update, now adopted as an IETF SIDROPS working group document.**
- ***Coding*: disable the libcrypto RFC3779 containment checking via flag or verify callback.**
- ***Coding*: implement the “new” validation algorithm in rpki-client, and other validators.**

Q&A

Photo credit: https://flic.kr/p/6nCmik

Further reading on recent RPKI work

- **RP handling of RPKI CRL Number extensions**
  draft-spaghetti-sidrops-rpki-crl-numbers
- **Detecting RRDP Session Desynchronization**
  draft-ietf-sidrops-rrdp-desynchronization
- **Route Origin Authorization profile revision (*approved as RFC*)**
  draft-ietf-sidrops-rfc6482bis
- **Optimizing RRDP → RSYNC switchovers (*approved as RFC*)**
  draft-ietf-sidrops-cms-signing-time
- **Closing a potential DDoS issue in RRDP**
  draft-spaghetti-sidrops-rrdp-same-origin
- **Dealing with maxed out Manifest Numbers**
  draft-ietf-sidrops-manifest-numbers
- **Constraining RPKI Trust Anchors**
  draft-snijders-constraining-rpki-trust-anchors