RFC 6488 : Signed Object Template for the Resource Public Key Infrastructure

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 4 février 2012

Date de publication du RFC : Février 2012

https://www.bortzmeyer.org/6488.html

Le système de sécurisation de BGP <https://www.bortzmeyer.org/securite-routage-bgp-rpki-roa.html> et du routage sur l'Internet, qui tourne autour de la RPKI, utilise des objets numériques signés cryptographiquement pour représenter les autorisations d'annoncer telle ou telle route. Ces objets sont définis avec la syntaxe CMS et le profil que normalise ce RFC.

Le contenu typique de ces objets est une autorisation du genre « L'AS 64641 est autorisé à annoncer le préfixe 192.0.2.0/24 ». Ils sont ensuite signés avec les clés contenues dans les certificats qui identifient le titulaire de ces ressources (ici, seul le titulaire de 192.0.2.0/24 peut émettre l'autorisation ci-dessus). Ces certificats suivent le format du RFC 6487 [1].

Les objets d'autorisation utilisent la syntaxe de CMS (RFC 5652), celle-ci permettant de réutiliser les nombreux outils CMS existants (dont plusieurs en logiciel libre, voir mon article sur les outils de la RPKI <https://www.bortzmeyer.org/rpki-tests.html>). Sur cette syntaxe CMS est bâti le gabarit présenté par ce RFC 6488, qui s'applique à tous les objets de la RPKI. Enfin, sur ce gabarit, est bâti la syntaxe précise de chaque type d'objet. Par exemple, les **ROA** ("*Route Origination Authorization*"), une classe particulière d'objets, sont normalisés dans le RFC 6482.

La syntaxe est décrit dans la section 2 du RFC. CMS utilisant ASN.1, le profil de notre RFC 6488 est décrit en ASN.1. On part de la définition SignedData du RFC 5652 :

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc6487.txt

```
SignedData ::= SEQUENCE {
  version CMSVersion,
  digestAlgorithms DigestAlgorithmIdentifiers,
  encapContentInfo EncapsulatedContentInfo,
  certificates [0] IMPLICIT CertificateSet OPTIONAL,
  crls [1] IMPLICIT RevocationInfoChoices OPTIONAL,
  signerInfos SignerInfos }
```

et on y ajoute quelques restrictions ou précisions. Par exemple, le `digestAlgorithm` doit être un de ceux spécifiés dans le RFC 6485. Même méthode pour les métadonnées de signature :

```
SignerInfo ::= SEQUENCE {
  version CMSVersion,
  sid SignerIdentifier,
  digestAlgorithm DigestAlgorithmIdentifier,
  signedAttrs [0] IMPLICIT SignedAttributes OPTIONAL,
  signatureAlgorithm SignatureAlgorithmIdentifier,
  signature SignatureValue,
  unsignedAttrs [1] IMPLICIT UnsignedAttributes OPTIONAL }
```

Les objets de la RPKI doivent être encodés en DER. La liste des règles sémantiques à respecter figure dans la section 3.

Je le rappelle, ce profil, bien que spécialisé par rapport à CMS, est encore très générique. Chaque classe d'objets signés va donc nécessiter son propre RFC, remplissant les blancs indiqués par la section 4 :

— Indiquer un OID pour la classe en question (1.2.840.113549.1.9.16.1.24 pour les ROA ou 1.2.840.113549.1.9.16.1.26 pour les manifestes du RFC 6486),
— Définir la syntaxe du champ `encapContentInfo` (qui contient les données effectivement signées),
— Indiquer les règles spécifiques de validation (pour un ROA : que le certificat signeur ait été émis pour le préfixe indiqué dans le ROA), s'ajoutant aux règles génériques de CMS (validité de la signature).

Ces classes d'objets sont ensuite enregistrés à l'IANA <https://www.iana.org/assignments/rpki/rpki.xml#signed-objects> (section 6).

Notez bien que la sécurité que fournissent ces objets signés est d'authentification et d'intégrité. Les données ne sont pas chiffrées et donc pas confidentielles (section 5). C'est logique, les tables BGP et les informations dans les IRR sont publiques.