RFC 9925 : Unsigned X.509 Certificates

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 27 février 2026

Date de publication du RFC : Février 2026

https://www.bortzmeyer.org/9925.html

Un certificat est signé par une AC, normalement, ou à la rigueur auto-signé, ce qui ne sert pas à grand'chose mais est obligatoire dans X.509. Mais voici que ce RFC rend la signature facultative.

À ma connaissance, le seul intérêt d'auto-signer son certificat est de prouver qu'on a bien la clé privée correspondante (car un plaisantin peut toujours fabriquer un certificat avec la clé publique de quelqu'un d'autre) et, en prime, de vérifier l'intégrité. C'est pas crucial. Mais la norme X.509 (cf. RFC 5280 [1]) ne permet pas d'avoir un certificat sans signature. Notre RFC 9925 contourne cette obligation en enregistrant un nouvel algorithme, `id-alg-unsigned`, qui indique qu'il n'y a pas de signature du tout.

C'est parce que, parfois, on n'a pas besoin des signatures. Un exemple typique se trouve dans votre magasin d'AC : vous ne vérifiez pas les signatures (vous ne pouvez pas, puisque les AC sont à la racine de la validation, RFC 5280, section 6), la seule présence du certificat dans le magasin inspire la confiance. Les certificats des AC sont donc auto-signés, ce qui ne sert pas à grand'chose, à part à satisfaire les exigences syntaxiques du format X.509. Ici, le certificat d'une AC du magasin de ma machine Ubuntu ; "*Issuer*" est égal à "*Subject*", le certificat est auto-signé :

```
% openssl x509 -text -in /etc/ssl/certs/Autoridad_de_Certificacion_Firmaprofesional_CIF_A62634068.pem
Certificate:
        Issuer: C = ES, CN = Autoridad de Certificacion Firmaprofesional CIF A62634068
        Subject: C = ES, CN = Autoridad de Certificacion Firmaprofesional CIF A62634068
 ...
            X509v3 Basic Constraints: critical
                 CA:TRUE, pathlen:1
             X509v3 Key Usage: critical
                 Certificate Sign, CRL Sign
        ...
```

1. Pour voir le RFC de numéro NNN, `https://www.ietf.org/rfc/rfcNNN.txt`, par exemple `https://www.ietf.org/rfc/rfc5280.txt`

(Une autre solution serait le RFC 5914 mais il ne semble pas très courant.)

Un autre cas où on se moque de la signature du certificat est celui où le certificat est validé d'une autre façon, par exemple un serveur EPP sur TLS qui a une liste de tous les clients possibles, et des certificats que le registre a distribués à ses BE.

Donc, il y a des cas où la signature ne sert guère, voire pas du tout. Mais est-ce gênant ? Ce n'est pas dramatique mais, dans certains cas, c'est sous-optimal, par exemple :

— Les signatures post-quantiques sont souvent de grande taille, donc cela rend les certificats plus gros pour rien.
— La clé publique dans le certificat peut utiliser un algorithme qui ne permet pas la signature (comme ML-KEM, cf. le futur RFC `draft-ietf-lamps-kyber-certificates`).

La section 3 du RFC fournit les détails techniques. Dans le certificat, on met le champ `signatureAlgorithm` à `id-alg-unsigned` et le champ `signatureValue` à une chaine vide. Le champ `issuer` pourrait être vide puisque personne n'a signé le certificat, mais comme il était obligatoire, pour éviter de choquer les applications actuelles, il vaut mieux mettre un `issuer` égal au `subject` (comme pour un auto-signé). On a le droit de mettre plutôt un nom `/id-rdna-unsigned=` (en fait `1.3.6.1.5.5.7.25.1` mais je montre la représentation textuelle du RFC 4514). RDNA signifie "*Relative Distinguished Name Attribute*" `<https://www.iana.org/assignments/smi-numbers/smi-numbers.xml#smi-numbers-1.3.6.1.5.5.7>` et un nouveau registre, « "*SMI Security for PKIX Relative Distinguished Name Attribute*" `<https://www.iana.org/assignments/smi-numbers/smi-numbers.xml#smi-numbers-1.3.6.1.5.5.7.25>` », est créé pour accueillir `id-rdna-unsigned` et d'autres futurs noms (politique de Spécification nécessaire, cf. RFC 8126).

Quand une application rencontre un de ces certificats non signés, si elle valide les signatures, elle doit évidemment rejeter le certificat. Si elle est plus laxiste (ou bien a d'autres mécanismes de vérification), elle peut l'accepter (section 4). Les applications et bibliothèques existantes font déjà cela (elles rejettent les signatures d'algorithmes inconnus) donc l'introduction des certificats non signés ne devrait rien casser.

Quelques conseils de sécurité figurent en section 5, par exemple de ne pas réutiliser les clés dans des contextes différents (X.509, jusqu'à présent, ignorait ce conseil pour les certificats auto-signés puisque la clé publiée servait aussi à signer. Plus de signature, plus de problème.)

Désolé, mais je n'ai pas trouvé comment générer aujourd'hui de tels certificats, que ce soit avec OpenSSL ou d'autres logiciels. Si vous trouvez, ça m'intéresse, je pourrais mettre des essais pratiques ici.