# RFC 7517 : JSON Web Key (JWK)

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 20 mai 2015

Date de publication du RFC : Mai 2015

https://www.bortzmeyer.org/7517.html

---

Le sigle JOSE ("*JavaScript Object Signing and Encryption*") désigne l'utilisation de moyens cryptographiques pour sécuriser des textes JSON. JOSE permet notamment de signer (RFC 7515[1]) et chiffrer (RFC 7516) des textes JSON. Pour cela, il faut des **clés** cryptographiques et ce RFC 7517 normalise la représentation de ces clés en JSON.

Une JWK ("*JSON Web Key*") est donc une clé représentée par un objet JSON. Que les amateurs de X.509 se rassurent, il n'est pas prévu de remplacer ce format par JSON :-) Le but essentiel est de pouvoir manipuler des clés dans le contexte de JSON (transmettre la clé en même temps qu'une signature, chiffrer une clé privée pour la transporter de manière sûre, etc). Outre la JWK, notre RFC définit aussi le "*JWK set*", un ensemble de JWK.

Commençons par un exemple simple, une clé sur une courbe elliptique :

```
{"kty":"EC",
 "crv":"P-256",
 "x":"f83OJ3D2xF1Bg8vub9tLe1gHMzV76e8Tus9uPHvRVEU",
 "y":"x_FEzRu9m36HLN_tue659LNpXW6pCyStikYjKIWI5a0",
 "kid":"Public key used in JWS A.3 example"
}
```

Ici, le membre `kty` identifie la clé comme utilisant les courbes elliptiques, `crv` indique l'usage de la courbe NIST P-256, `x` et `y` sont les coordonnées de la clé sur cette courbe. `kid` est un identificateur de clé, dont le format est libre.

La section 4 décrit en détail tous les membres possibles de l'objet clé. Parmi les principaux :

1. Pour voir le RFC de numéro NNN, `https://www.ietf.org/rfc/rfcNNN.txt`, par exemple `https://www.ietf.org/rfc/rfc7515.txt`

— `kty` vaut `EC` (courbes elliptiques) ou `RSA` (l'algorithme du même nom). D'autres valeurs sont possibles (cf. RFC 7518, et le registre IANA <https://www.iana.org/assignments/jose/jose.xhtml#web-key-types>).
— `use` (inutilisé dans l'exemple plus haut) indique l'utilisation prévue de la clé (`sig` pour signer, `enc` pour chiffrer...). Là encore, le RFC 7518 prévoit un registre des valeurs possibles <https://www.iana.org/assignments/jose/jose.xhtml#web-key-use>.
— `alg` désigne l'algorithme à utiliser avec cette clé, parmi ceux enregistrés à l'IANA <https://www.iana.org/assignments/jose/jose.xhtml#web-signature-encryption-algorithms>.
— `kid` (*"Key IDentification"*) sert à identifier la clé. Par exemple, lorsqu'on doit désigner une clé particulière au sein d'un ensemble de clés disponibles. Notez que son format n'est pas spécifié par notre RFC.
— `x5c` est un certificat X.509 (RFC 5280), ou une chaîne de certificats. Les certificats sont une valeur en DER encodée en Base64.
— `x5u` est un URL pointant vers un certificat X.509.

JWK peut servir pour la cryptographie asymétrique ou pour la symétrique. Voici un exemple de clé symétrique (le type `oct` signifie « suite d'octets », la clé elle-même est dans le membre `k`) :

```
{
   "kty":"oct",
   "k":"AyM1SysPpbyDfgZld3umj1qzKObwVMkoqQ-EstJQLr_T-1qS0gZH75aKtMN3Yj0iPS4hcgUuTwjAzZr1Z9CAow",
   "kid":"HMAC key used in JWS A.1 example"
}
```

Si on veut manipuler plusieurs clés ensemble (section 5), on peut utiliser un objet JSON comportant un membre `keys` dont la valeur est un tableau de JWK. Ici, une clé sur une courbe elliptique et une clé RSA :

```
{"keys":
  [
    {"kty":"EC",
     "crv":"P-256",
     "x":"MKBCTNIcKUSDii11ySs3526iDZ8AiTo7Tu6KPAqv7D4",
     "y":"4Etl6SRW2YiLUrN5vfvVHuhp7x8PxltmWWlbbM4IFyM",
     "use":"enc",
     "kid":"1"},

    {"kty":"RSA",
     "n": "0vx7agoebGcQSuuPiLJXZptN9nndrQmbXEps2aiAFbWhM78LhWx4cbbfAAtVT86zwu1RK7aPFFxuhDR1L6tSoc_BJECP
     "e":"AQAB",
     "alg":"RS256",
     "kid":"2011-04-29"}
  ]
}
```

Le format JWK permet de représenter des clés privées, que ce soit la partie privée d'une clé asymétrique, ou bien une clé symétrique. Si on échange ces clés, il est évidemment recommandé de les chiffrer, ce qui peut se faire avec JOSE (RFC 7516). Une JWK chiffrée est alors un JWE (contenu chiffré en JOSE) comme les autres.

Comme tous les RFC JOSE, cette norme fait une forte consommation de registres IANA. La section 8 leur est consacrée. La politique d'enregistrement dans ces registres est « description nécessaire » (cf. RFC 5226), ce qui veut dire qu'un texte stable décrivant la valeur enregistrée est nécessaire. Par exemple, si on veut enregistrer un nouvel algorithme cryptographique pour les clés, on doit indiquer un texte décrivant précisément cet algorithme. Les registres sont notamment :

— Les paramètres des clés <https://www.iana.org/assignments/jose/jose.xhtml#web-key-parameter (comme use pour l'usage prévu, ou kid pour l'identificateur). Comme toujours dans le monde JSON, les membres inconnus d'un objet sont ignorés. Ainsi, un nouveau membre enregistré ici pourra être utilisé sans risque puisque les mises en œuvre de JOSE plus anciennes ignoreront ce nouveau membre.
— Les utilisations des clés <https://www.iana.org/assignments/jose/jose.xhtml#web-key-use> comme sig pour la signature ou bien enc pour le chiffrement.

Outre ces nouveaux registres, ce RFC ajoute au registre des types de média <https://www.iana.org/assignments/media-types/media-types.xhtml#application> le type application/jwk+json qui identifie une clé encodée en JSON et application/jwk-set+json qui identifie un ensemble de clés.

La section 9 de notre RFC se penche sur la sécurité, évidemment un sujet important quand on manipule des clés. Les règles à suivre sont les règles classiques de la cryptographie. Par exemple, les clés privées doivent être gardées à l'abri des regards indiscrets, comme rappelé dans les RFC 3447 et RFC 6030.

Autre piège, risquer de croire que, parce qu'un texte est signé, il a davantage de valeur. Cela n'est vrai que si on peut s'assurer de l'origine de la clé. Autrement, cela ne signifie rien, n'importe qui ayant pu se créer une clé. Un des moyens de vérifier cette origine est de valider le certificat PKIX dans le membre x5c. Au passage, les exemples de certificats dans le RFC sont, comme le veut le format JWK de JOSE, du DER encodé en Base64. Pour les lire, on peut par exemple, mettre le contenu en Base64 (après avoir retiré les espaces et sauts de ligne qui se trouvent dans le RFC) dans un fichier, le décoder avec base64 puis le lire avec OpenSSL :

```
% base64 -d < rfc.b64 > rfc.der
% openssl x509 -text -inform der -in rfc.der
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            01:3c:ff:16:e2:e2
    Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=US, ST=CO, L=Denver, O=Ping Identity Corp., CN=Brian Campbell
        Validity
            Not Before: Feb 21 23:29:15 2013 GMT
            Not After : Aug 14 22:29:15 2018 GMT
        Subject: C=US, ST=CO, L=Denver, O=Ping Identity Corp., CN=Brian Campbell
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
...
```