RFC 7516 : JSON Web Encryption (JWE)

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 3 juin 2015

Date de publication du RFC : Mai 2015

https://www.bortzmeyer.org/7516.html

L'ensemble de normes de cryptographie JOSE <https://www.bortzmeyer.org/jose.html>, qui permet de sécuriser des textes JSON comprend plusieurs RFC, dont ce RFC 7516 [1], qui normalise le chiffrement, permettant d'assurer la confidentialité des documents JSON.

Un document chiffré est baptisé JWE (pour "*JSON Web Encryption*"). Il peut exister sous deux formes (deux **sérialisations**), une compacte (en Base64, cf. section 3.1), qui peut notamment être utilisée dans les URL, et une en JSON, qui permet de passer par les programmes de traitement du JSON (sections 3.2 et 7.2.1). Elles utilisent les mêmes mécanismes cryptographiques sous-jacents. Les algorithmes utilisables sont normalisés dans le RFC 7518. Il est recommandé de lire également le RFC sur les signatures JOSE (RFC 7515) car beaucoup de règles applicables à la fois à la signature et au chiffrement ne sont décrites que dans ce RFC 7515 (comme la comparaison des chaînes de caractères de la section 5.3).

Les principes de base de JWE sont exposés dans la section 3 du RFC (en s'appuyant que la terminologie de la section 2). Quelle que soit la sérialisation choisie, un JWE peut comprendre :

— L'en-tête JOSE, lui-même composé d'une partie protégée et d'une ou plusieurs parties non protégées,
— La clé cryptographique, chiffrée,
— Le vecteur d'initialisation,
— L'AAD ("*Additional Authenticated Data*"), protégée en intégrité mais pas chiffrée, qui sert lorsqu'on fait de l'AEAD,
— Le texte chiffré (et donc illisible par un indiscret),
— L'"*authentication tag*".

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc7516.txt

Dans la sérialisation en JSON, les composants binaires (comme le texte chiffré) sont encodés en Base64, JSON n'ayant pas de mécanisme pour les données binaires.

Un JWE en sérialisation compacte est la concaténation des différents composants cités ci-dessus, chacun encodé en Base64 (un exemple figure en section 3.3). Un JWE en sérialisation JSON est un objet JSON comportant un membre par composant, par exemple `iv` pour le vecteur d'initialisation, `ciphertext` pour le texte chiffré, `recipients` pour des informations sur les destinataires (optionnel : s'il n'y a qu'un seul destinataire, on peut mettre ces informations dans l'objet JWE, cf. section 7.2.2).

L'en-tête JOSE est détaillé dans la section 4. Il inclut la plupart des paramètres cryptographiques. On le représente par un objet JSON (qui sera ensuite encodé en Base64). Cet objet a des membres comme `alg` (l'algorithme de chiffrement utilisé pour la clé, en général un algorithme de chiffrement asymétrique) ou `enc` (l'algorithme de chiffrement utilisé pour le contenu, en général un algorithme de chiffrement symétrique), etc. Les algorithmes possibles sont décrits dans le RFC 7518. De nombreux autres membres sont possibles comme `zip` pour indiquer l'éventuel algorithme de compression utilisé (par exemple celui du RFC 1951).

Le mécanisme complet de chiffrement et de déchiffrement (ce dernier est l'inverse du chiffrement) est décrit dans la section 5. Voici un exemple, tiré de l'annexe A.4 : il y a deux destinataires, le premier utilisera RSA et le second AES pour chiffrer la clé :

```
"recipients":[
       {"header":
         {"alg":"RSA1_5","kid":"2011-04-29"},
       {"header":
         {"alg":"A128KW","kid":"7"},
 ...
```

Le texte sera chiffré avec AES en mode CBC avec du HMAC sur SHA-256 pour produire l'"*authentication tag*" :

```
  {"enc":"A128CBC-HS256"}
```

(Cet objet formera l'en-tête protégé.) En utilisant cette valeur pour le vecteur d'initialisation :

```
  "iv":       "AxY8DCtDaGlsbGljb3RoZQ",
```

et cette clé de chiffrement :

```
  {"kty":"oct",
      "k":"GawgguFyGrWKav7AX4VKUg"
      }
```

Et le message en clair « *"Live long and prosper."* », on arrive à ce JWE complet :

```
{
    "protected":
     "eyJlbmMiOiJBMTI4Q0JDLUhTMjU2In0",
    "unprotected":
     {"jku":"https://server.example.com/keys.jwks"},
    "recipients":[
     {"header":
       {"alg":"RSA1_5","kid":"2011-04-29"},
      "encrypted_key":
       "UGhIOguC7IuEvf_NPVaXsGMoLOmwvc1GyqlIKOK1nN94nHPoltGRhWhw7Zx0-
        kFm1NJn8LE9XShH59_i8J0PH5ZZyNfGy2xGdULU7sHNF6Gp2vPLgNZ__deLKx
        GHZ7PcHALUzoOegEI-8E66jX2E4zyJKx-YxzZIItRzC5hlRirb6Y5Cl_p-ko3
        YvkkysZIFNPccxRU7qve1WYPxqbb2Yw8kZqa2rMWI5ng8OtvzlV7elprCbuPh
        cCdZ6XDP0_F8rkXds2vE4X-ncOIM8hAYHHi29NX0mcKiRaD0-D-ljQTP-cFPg
        wCp6X-nZZd9OHBv-B3oWh2TbqmScqXMR4gp_A"},
     {"header":
       {"alg":"A128KW","kid":"7"},
      "encrypted_key":
       "6KB707dM9YTIgHtLvtgWQ8mKwboJW3of9locizkDTHzBC2IlrT1oOQ"}],
    "iv":
     "AxY8DCtDaGlsbGljb3RoZQ",
    "ciphertext":
     "KDlTtXchhZTGufMYmOYGS4HffxPSUrfmqCHXaI9wOGY",
    "tag":
     "Mz-VPPyU4RlcuYv1IwIvzw"
     }
```

Pour du code qui met en œuvre le chiffrement JOSE, voir mon article général <https://www.bortzmeyer.org/jose.html>.