eBook Gratuit

APPRENEZ jwt

eBook gratuit non affilié créé à partir des **contributeurs de Stack Overflow.**

#jwt

Table des matières

À propos

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: jwt

It is an unofficial and free jwt ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official jwt.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapitre 1: Commencer avec jwt

Remarques

Un jeton Web JSON (JWT) est un moyen compact, sécurisé par des URL, de représenter des revendications pouvant être échangées entre des parties.

Tous les JWT consistent en un en- **tête** et une **charge utile** , qui sont des hachages JSON. Ces objets sont stringifiés et codés en Base64. L'en-tête et la charge utile codés sont combinés à une signature numérique (JWS) et les trois composants sont concaténés avec "." (période).

Lectures complémentaires

- Cas d'utilisation et conditions requises pour la signature et le chiffrement d'objets JSON (RFC 7165)
- Spécification de signature Web JSON (RFC 7515)
- Spécification JSON Web Encryption (RFC 7516)
- Clé Web JSON (RFC 7517)
- Algorithmes Web JSON (RFC 7518)
- Spécification du jeton Web JSON (RFC 7519)
- Liste IANA des revendications de jeton Web JSON (liste IANA RFC 7519)
- Exemples de protection du contenu à l'aide de la signature et du chiffrement d'objets JSON (RFC 7520)
- Empreinte JWK (clé Web JSON) (RFC 7638)
- Option de charge utile non codée JWS (JSON Web Signature) (RFC 7797)

Examples

JWT non signé

Un JWT non signé a la valeur d'en-tête `alg: none` et un composant JWS (signature) vide:

```
eyJhbGciOiJub25lIn0
.eyJpc3MiOiJqb2UiLA0KICJleHAiOjEzMDA4MTkzODAsDQogImh0dHA6Ly9leGFtcGxlLmNvbS9pc19yb290Ijp0cnVlfQ

.
```

Le point final indique que la signature est vide.

Entête

```
{
  "alg": "none"
}
```

Charge utile

```
{
  "iss": "joe",
  "exp": 1300819380,
  "http://example.com/is_root": true
}
```

JWT signé (JWS)

Un JWT signé inclut une signature codée Url Base64 en tant que troisième composant. L'algorithme utilisé pour générer la signature est indiqué dans l'en-tête.

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9
.eyJzdWIiOiJKb2huIERvZSIsImFkbWluIjp0cnVlLCJpYXQiOjE0NzAzNTM5OTQsImV4cCI6MTQ3MDM1NzYyNywianRpIjoiNmU0MD

.7CfBdVP4uKsb0cogYepCvMLm8rcpjBYW1XZzA-a5e44
```

Entête

```
{
  "typ": "JWT",
  "alg": "HS256"
}
```

Ce JWT a été signé avec l'algorithme HMAC-SHA256, donc `alg: HS256` .

Charge utile

```
{
  "sub": "John Doe",
  "admin": true,
  "iat": 1470353994,
  "exp": 1470357627,
  "jti": "6e404ba8-f885-4d5f-bfa2-e3f5a08380a4"
}
```

Ce JWT peut être vérifié avec le `notsosecret` secret UTF-8.

JSON Web Encryption (JWE)

JSON Web Encryption (JWE) représente le contenu chiffré à l'aide de structures de données basées sur JSON (JavaScript Object Notation). Il définit un moyen de chiffrer vos données de réclamation afin que seul le destinataire puisse lire les informations présentes dans un jeton.

Dans la sérialisation JSON JWE, un JWE est représenté comme un objet JSON contenant certains ou tous ces huit membres:

```
  "protected", with the value BASE64URL(UTF8(JWE Protected Header))
  "unprotected", with the value JWE Shared Unprotected Header
  "header", with the value JWE Per-Recipient Unprotected Header
  "encrypted_key", with the value BASE64URL(JWE Encrypted Key)
  "iv", with the value BASE64URL(JWE Initialization Vector)
  "ciphertext", with the value BASE64URL(JWE Ciphertext)
  "tag", with the value BASE64URL(JWE Authentication Tag)
  "aad", with the value BASE64URL(JWE AAD)
```

Les six chaînes de résultats encodées en base64url et les deux valeurs d'objet JSON non protégées sont représentées en tant que membres dans un objet JSON.

Exemple JWE

L'exemple suivant d'en-tête JWE déclare que:

- la clé de cryptage de contenu est cryptée sur le destinataire à l'aide de l'algorithme RSA-PKCS1_1.5 pour produire la clé cryptée JWE
- le texte en clair est chiffré à l'aide de l'algorithme AES-256-GCM pour produire le texte chiffré JWE
- le vecteur d'initialisation 64 bits spécifié avec l'encodage base64url __79_Pv6-fg a été utilisé
- L'empreinte du certificat X.509 correspondant à la clé utilisée pour chiffrer le JWE a l'encodage base64url 7noOPq-hJ1_hCnvWh6IeYI2w9Q0.

```
{
 "alg":"RSA1_5",
 "enc":"A256GCM",
 "iv":"__79_Pv6-fg",
 "x5t":"7noOPq-hJ1_hCnvWh6IeYI2w9Q0"
}
```

Base64url codant les octets de la représentation UTF-8 de l'en-tête JWE génère cette valeur d'en-tête JWE codée (avec sauts de ligne uniquement à des fins d'affichage):

```
eyJhbGciOiJSU0ExXzUiLA0KICJlbmMiOiJBMjU2R0NNIiwNCiAiaXYiOiJfXzc5
X1B2Ni1mZyIsDQogIng1dCI6Ijdub09QcS1oSjFfaENudldoNkllWUkydzlRMCJ
```

Lisez la spécification JSON Web Encryption (RFC 7516) pour plus d'informations.

Comment savoir si vous avez un JWS ou un JWE?

À partir de la section 9 de la spécification JSON Web Encryption (RFC 7516):

> L'en-tête JOSE d'un JWS peut être distingué de l'en-tête JOSE d'un JWE en examinant la valeur du paramètre d'en-tête "alg" (algorithme). Si la valeur représente une signature numérique ou un algorithme MAC, ou si la valeur est "none", c'est pour un JWS; s'il s'agit d'un algorithme de chiffrement de clé, de verrouillage de clé, de contrat à clé directe, d'accord de clé avec habillage de clé ou de chiffrement direct, il s'agit d'un JWE. (L'extraction de la valeur "alg" à examiner est simple lors de l'utilisation de la sérialisation JWS Compact ou de la sérialisation JWE Compact et

peut s'avérer plus difficile lors de l'utilisation de la sérialisation JSON JWS ou de la sérialisation JWE JWE.)

Et

> L'en-tête JOSE d'un JWS peut également être distingué de l'en-tête JOSE d'un JWE en déterminant s'il existe un membre "enc" (algorithme de chiffrement). Si le membre "enc" existe, c'est un JWE; sinon, c'est un JWS.

JWS (signé)

```
{
  "alg": "HS256"
}
```

JWE (crypté)

```
{
  "alg":"RSA1_5",
  "enc":"A256GCM",
  "iv":"__79_Pv6-fg",
  "x5t":"7noOPq-hJ1_hCnvWh6IeYI2w9Q0"
}
```

Que stocker dans un JWT

Le JWT RFC établit trois classes de revendications:

- **Réclamations enregistrées** comme `sub` , `iss` , `exp` ou `nbf`
- **Réclamations publiques** avec des noms publics ou des noms enregistrés par l'IANA qui contiennent des valeurs qui devraient être uniques comme `email` , `address` ou `phone_number` . Voir la liste complète
- **Les revendications privées** à utiliser dans votre propre contexte et les valeurs peuvent provoquer des collisions

Aucune de ces revendications n'est obligatoire

Un JWT est autonome et doit éviter d'utiliser la session du serveur fournissant les données nécessaires pour effectuer l'authentification (pas besoin de stockage sur le serveur et d'accès à la base de données). Par conséquent, les informations sur les `role` ou les `permissions` peuvent être incluses dans les revendications privées de JWT.

Réclamations enregistrées

Les noms de réclamation suivants sont enregistrés dans le registre IANA "JSON Web Token Claims" établi au paragraphe 10.1.

- `iss` (émetteur): identifie le principal qui a émis le JWT.
- `sub` (sujet): identifie le principal objet de la JWT. Doit être unique
- `aud` (audience): identifie les destinataires auxquels le JWT est destiné (tableau de chaînes / uri)
- `exp` (délai d'expiration): identifie le délai d'expiration (UTC Unix) après lequel vous ne devez plus accepter ce jeton. Il devrait être après le moment de l'émission.
- `nbf` (pas avant): identifie l'heure UTC Unix avant laquelle le JWT ne doit pas être accepté
- `iat` (délivré à): identifie l'heure UTC Unix à laquelle le JWT a été émis
- `jti` (JWT ID): fournit un identifiant unique pour le JWT.

Exemple

```
{
    "iss": "stackoverflow",
    "sub": "joe",
    "aud": ["all"],
    "iat": 1300819370,
    "exp": 1300819380,
    "jti": "3F2504E0-4F89-11D3-9A0C-0305E82C3301"
    "context": {
        "user": {
            "key": "joe",
            "displayName": "Joe Smith"
        },
        "roles":["admin","finaluser"]
    }
}
```

Lire Commencer avec jwt en ligne: https://riptutorial.com/fr/jwt/topic/5213/commencer-avec-jwt

Chapitre 2: Invalider les jetons Web Json

Remarques

Il y a plusieurs raisons d'invalider un jeton JWT avant sa date d'expiration: compte supprimé / bloqué / suspendu, mot de passe ou autorisations modifiés, utilisateur déconnecté par l'administrateur.

JWT est autonome, signé et stocké en dehors du contexte du serveur. La révocation d'un jeton n'est donc pas une action simple.

Examples

Supprimer le jeton du stockage client

Supprimer le jeton du stockage client pour éviter l'utilisation

Les jetons sont émis par le serveur et vous ne pouvez pas forcer les navigateurs à supprimer un cookie / localStorage ou à contrôler la façon dont les clients externes gèrent vos jetons. De toute évidence, **si des attaquants ont volé le jeton avant la déconnexion,** ils pourraient toujours utiliser le jeton. Par conséquent, **des mesures supplémentaires sont nécessaires côté serveur** (voir ci-dessous la stratégie de la liste noire de jetons).

Biscuits

Vous ne pouvez pas forcer les navigateurs à supprimer un cookie. Le client peut configurer le navigateur de telle manière que le cookie persiste, même s'il a expiré. Mais le serveur peut définir la valeur à vider et inclure le champ expire pour invalider la valeur du cookie.

```
Set-Cookie: token=deleted; path=/; expires=Thu, 01 Jan 1970 00:00:00 GMT
```

Supprimer 'token' avec javascript

```
document.cookie = 'token=; Path=/; Expires=Thu, 01 Jan 1970 00:00:01 GMT;';
localStorage.removeItem('token')
sessionStorage.removeItem('token')
```

Liste noire des jetons

Marquez les jetons invalides, stockez jusqu'à leur date d'expiration et vérifiez-la dans chaque requête.

La liste noire brise l'apatridie de la JWT parce qu'elle nécessite le maintien de l'état. L'un des avantages de JWT n'est pas le stockage sur serveur, donc si vous devez révoquer des jetons

sans attendre l'expiration, réfléchissez également aux inconvénients.

Gérer la liste noire

La liste noire peut être facilement gérée dans votre propre service / base de données. La taille de stockage ne serait probablement pas grande car elle est uniquement nécessaire pour stocker les jetons entre la déconnexion et la date d'expiration.

Incluez le jeton complet ou simplement l'identifiant unique `jti` . Définissez le `iat` (émis à) pour supprimer les anciens jetons.

Pour révoquer tous les jetons après la mise à jour des données critiques sur l'utilisateur (mot de passe, autorisations, etc.), définissez une nouvelle entrée avec `sub` et `iat` lorsque `currentTime - maxExpiryTime < last iss` . L'entrée peut être supprimée lorsque `currentTime - maxExpiryTime > lastModified` (plus de jetons non expirés envoyés).

Faire pivoter les jetons

Définissez un **délai d'expiration court et faites pivoter les jetons** . Émettez un nouveau **jeton d'accès à** chaque demande. Utilisez des **jetons d'actualisation** pour permettre à votre application d'obtenir de nouveaux jetons d'accès sans avoir à effectuer une nouvelle authentification

Actualiser et accéder aux jetons

- **jeton d'accès** : Autoriser l'accès à une ressource protégée. Durée de vie limitée. Doit être gardé secret, les considérations de sécurité sont moins strictes en raison de leur durée de vie plus courte.
- **Actualiser le jeton** : permet à votre application d'obtenir de nouveaux jetons d'accès sans avoir à se ré-authentifier. Longue durée de vie Stocker dans un stockage sécurisé à long terme

Recommandations d'utilisation:

- **Applications Web** : actualisez le jeton d'accès avant son expiration, chaque fois que l'utilisateur ouvre l'application et à intervalles fixes. Vous pouvez également renouveler le jeton d'accès lorsqu'un utilisateur effectue une action. Si l'utilisateur utilise un jeton d'accès expiré, la session est considérée comme inactive et un nouveau jeton d'accès est requis. Ce nouveau jeton peut être obtenu avec un jeton d'actualisation ou nécessitant des informations d'identification.
- Applications **mobiles / natives** : connexion à une application une seule fois. Le jeton d'actualisation n'expire pas et peut être échangé contre un JWT valide. Prendre en compte des événements spéciaux comme changer le mot de passe

Autres techniques courantes

- Autoriser le changement d'identifiant unique de l'utilisateur si le compte est compromis avec un nouvel identifiant d'utilisateur et de mot de passe
- Pour invalider les jetons lorsque l'utilisateur modifie son mot de passe ou ses autorisations, signez le jeton avec un hachage de ces champs. Si l'un de ces champs change, tous les jetons précédents ne sont pas vérifiés automatiquement. L'inconvénient est qu'il nécessite un accès à la base de données
- Modifier l'algorithme de signature pour révoquer tous les jetons actuels dans un problème de sécurité majeur

Lire Invalider les jetons Web Json en ligne: https://riptutorial.com/fr/jwt/topic/6224/invalider-les-jetons-web-json

Chapitre 3: Sérialisations

Examples

Sérialisation JWS Compact

La sérialisation compacte est le format de sérialisation le plus courant et est conçu pour être utilisé dans un contexte Web.

Les JWS sont représentés dans une chaîne qui contient des informations codées Url Base64 Safe séparées par un point ".".

Ce mode ne prend pas en charge les en-têtes non protégés.

Les sauts de ligne ajoutés pour plus de lisibilité

```
BASE64URL(UTF8(JWS Protected Header)) || '.' ||
BASE64URL(JWS Payload) || '.' ||
BASE64URL(JWS Signature)
```

Exemple

```
eyJhbGciOiJQUzM4NCIsImtpZCI6ImJpbGJvLmJhZ2dpbnNAaG9iYml0b24uZX
hhbXBsZSJ9
.
SXTigJlzIGEgZGFuZ2Vyb3VzIGJ1c2luZXNzLCBGcm9kbywgZ29pbmcgb3V0IH
lvdXIgZG9vci4gWW91IHN0ZXAgb250byB0aGUgcm9hZCwgYW5kIGlmIHlvdSBk
b24ndCBrZWVwIHlvdXIgZmVldCwgdGhlcmXigJlzIG5vIGtub3dpbmcgd2hlcm
UgeW91IG1pZ2h0IGJlIHN3ZXB0IG9mZiB0by4
.
cu22eBqkYDKgIlTpzDXGvaFfz6WGoz7fUDcfT0kkOy42miAh2qyBzk1xEsnk2I
pN6-tPid6VrklHkqsGqDqHCdP6O8TTB5dDDItllVo6_1OLPpcbUrhiUSMxbbXU
vdvWXzg-UD8biiReQFlfz28zGWVsdiNAUf8ZnyPEgVFn442ZdNqiVJRmBqrYRX
e8P_ijQ7p8Vdz0TTrxUeT3lm8d9shnr2lfJT8ImUjvAA2Xez2Mlp8cBE5awDzT
0qI0n6uiP1aCN_2_jLAeQTlqRHtfa64QQSUmFAAjVKPbByi7xho0uTOcbH510a
6GYmJUAfmWjwZ6oD4ifKo8DYM-X72Eaw
```

Sérialisation compacte JWE

La sérialisation compacte est le format de sérialisation le plus courant et est conçu pour être utilisé dans un contexte Web.

JWE sont représentés dans une chaîne qui contient les informations codées Url Base64 Safe séparées par un point ".".

Ce mode ne prend pas en charge les en-têtes non protégés ou AAD.

Les sauts de ligne ajoutés pour plus de lisibilité

```
BASE64URL(UTF8(JWE Protected Header)) || '.' ||
BASE64URL(JWE Encrypted Key) || '.' ||
BASE64URL(JWE Initialization Vector) || '.' ||
BASE64URL(JWE Ciphertext) || '.' ||
BASE64URL(JWE Authentication Tag)
```

Exemple

```
eyJhbGciOiJSU0EtT0FFUCIsImtpZCI6InNhbXdpc2UuZ2FtZ2VlQGhvYmJpdG
9uLmV4YW1wbGUiLCJlbmMiOiJBMjU2R0NNIn0
.
rT99rwrBTbTI7IJM8fU3Eli7226HEB7IchCxNuh7lCiud48LxeolRdtFF4nzQi
beYOl5S_PJsAXZwSXtDePz9hk-BbtsTBqC2UsPOdwjC9NhNupNNu9uHIVftDyu
cvI6hvALeZ6OGnhNV4v1zx2k7O1D89mAzfw-_kT3tkuorpDU-CpBENfIHX1Q58
-Aad3FzMuo3Fn9buEP2yXakLXYa15BUXQsupM4A1GD4_H4Bd7V3u9h8Gkg8Bpx
KdUV9ScfJQTcYm6eJEBz3aSwIaK4T3-dwWpuBOhROQXBosJzS1asnuHtVMt2pK
IIfux5BC6huIvmY7kzV7W7aIUrpYm_3H4zYvyMeq5pGqFmW2k8zpO878TRlZx7
pZfPYDSXZyS0CfKKkMozT_qiCwZTSz4duYnt8hS4Z9sGthXn9uDqd6wycMagnQ
fOTs_lycTWmY-aqWVDKhjYNRf03NiwRtb5BE-tOdFwCASQj3uuAgPGrO2AWBe3
8UjQb0lvXn1SpyvYZ3WFc7WOJYaTa7A8DRn6MC6T-xDmMuxC0G7S2rscw5lQQU
06MvZTlFOt0UvfuKBa03cxA_nIBIhLMjY2kOTxQMmpDPTr6Cbo8aKaOnx6ASE5
Jx9paBpnNmOOKH35j_QlrQhDWUN6A2Gg8iFayJ69xDEdHAVCGRzN3woEI2ozDR
s
.
-nBoKLH0YkLZPSI9
.
o4k2cnGN8rSSw3IDo1YuySkqeS_t2m1GXklSgqBdpACm6UJuJowOHC5ytjqYgR
L-I-soPlwqMUf4UgRWWeaOGNw6vGW-xyM01lTYxrXfVzIIaRdhYtEMRBvBWbEw
P7ua1DRfvaOjgZv6Ifa3brcAM64d8p5lhhNcizPersuhw5f-pGYzseva-TUaL8
iWnctc-sSwy7SQmRkfhDjwbz0fz6kFovEgj64X1I5s7E6GLp5fnbYGLa1QUiML
7Cc2GxgvI7zqWo0YIEc7aCflLG1-8BboVWFdZKLK9vNoycrYHumwzKluLWEbSV
maPpOslY2n525DxDfWaVFUfKQxMF56vn4B9QMpWAbnypNimbM8zVOw
.
UCGiqJxhBI3IFVdPalHHvA
```

Syntaxe de sérialisation JSON JSON générale

La sérialisation JSON JWS représente le contenu signé numériquement ou MACed en tant qu'objet JSON. Cette représentation n'est ni optimisée pour la compacité ni la sécurité des URL.

Cette syntaxe est optimisée pour plusieurs signatures numériques et / ou opérations MAC.

Les sauts de ligne ajoutés pour plus de lisibilité

```
     {
      "payload":"<payload contents>",
      "signatures":[
       {"protected":"<integrity-protected header 1 contents>",
        "header":<non-integrity-protected header 1 contents>,
        "signature":"<signature 1 contents>"},
       ...
       {"protected":"<integrity-protected header N contents>",
        "header":<non-integrity-protected header N contents>,
        "signature":"<signature N contents>"}]
     }
```

Exemple

```
{
  "payload": "SXTigJlzIGEgZGFuZ2Vyb3VzIGJ1c2luZXNzLCBGcm9kbywg
      Z29pbmcgb3V0IHlvdXIgZG9vci4gWW91IHN0ZXAgb250byB0aGUgcm9h
      ZCwgYW5kIGlmIHlvdSBkb24ndCBrZWVwIHlvdXIgZmVldCwgdGhlcmXi
      gJlzIG5vIGtub3dpbmcgd2hlcmUgeW91IG1pZ2h0IGJlIHN3ZXB0IG9m
      ZiB0by4",
  "signatures": [
    {
      "protected": "eyJhbGciOiJSUzI1NiJ9",
      "header": {
        "kid": "bilbo.baggins@hobbiton.example"
      },
      "signature": "MIsjqtVlOpa71KE-Mss8_Nq2YH4FGhiocsqrgi5Nvy
          G53uoimic1tcMdSg-qptrzZc7CG6Svw2Y13TDIqHzTUrL_lR2ZFc
          ryNFiHkSw129EghGpwkpxaTn_THJTCglNbADko1MZBCdwzJxwqZc
          -1RlpO2HibUYyXSwO97BSe0_evZKdjvvKSgsIqjytKSeAMbhMBdM
          ma622_BG5t4sdbuCHtFjp9iJmkio47AIwqkZV1aIZsv33uPUqBBC
          XbYoQJwt7mxPftHmNlGoOSMxR_3thmXTCm4US-xiNOyhbm8afKK6
          4jU6_TPtQHiJeQJxz9G3Tx-083B745_AfYOnlC9w"
    },
    {
      "header": {
        "alg": "ES512",
        "kid": "bilbo.baggins@hobbiton.example"
      },
      "signature": "ARcVLnaJJaUWG8fG-8t5BREVAuTY8n8YHjwDO1muhc
          dCoFZFFjfISu0Cdkn9Ybdlmi54ho0x924DUz8sK7ZXkhc7AFM8Ob
          LfTvNCrqcI3Jkl2U5IX3utNhODH6v7xgy1Qahsn0fyb4zSAkje8b
          AWz4vIfj5pCMYxxm4fgV3q7ZYhm5eD"
    },
    {
      "protected": "eyJhbGciOiJIUzI1NiIsImtpZCI6IjAxOGMwYWU1LT
          RkOWItNDcxYi1iZmQ2LWVlZjMxNGJjNzAzNyJ9",
      "signature": "s0h6KThzkfBBBkLspW1h84VsJZFTsPPqMDA7g1Md7p
          0"
    }
  ]
}
```

Syntaxe de sérialisation JSON JSON aplatie

Comme la syntaxe de sérialisation JSON JWS générale, la sérialisation JSON JWS représente le contenu signé numériquement ou MACed en tant qu'objet JSON. Cette représentation n'est ni optimisée pour la compacité ni la sécurité des URL.

La syntaxe aplatie est optimisée pour la signature numérique unique ou le boîtier MAC.

Les sauts de ligne ajoutés pour plus de lisibilité

```
  {
   "payload":"<payload contents>",
   "protected":"<integrity-protected header contents>",
   "header":<non-integrity-protected header contents>,
   "signature":"<signature contents>"
```

```
  }
```

Exemple

```
{
  "payload": "SXTigJlzIGEgZGFuZ2Vyb3VzIGJ1c2luZXNzLCBGcm9kbywg
      Z29pbmcgb3V0IHlvdXIgZG9vci4gWW91IHN0ZXAgb250byB0aGUgcm9h
      ZCwgYW5kIGlmIHlvdSBkb24ndCBrZWVwIHlvdXIgZmVldCwgdGhlcmXi
      gJlzIG5vIGtub3dpbmcgd2hlcmUgeW91IG1pZ2h0IGJlIHN3ZXB0IG9m
      ZiB0by4",
  "protected": "eyJhbGciOiJIUzI1NiJ9",
  "header": {
    "kid": "018c0ae5-4d9b-471b-bfd6-eef314bc7037"
  },
  "signature": "bWUSVaxorn7bEF1djytBd0kHv70Ly5pvbomzMWSOr20"
}
```

Syntaxe de sérialisation JSON JSON générale

La sérialisation JWE JSON représente le contenu chiffré en tant qu'objet JSON. Cette représentation n'est ni optimisée pour la compacité ni la sécurité des URL.

Cette syntaxe est optimisée pour plusieurs destinataires.

Les sauts de ligne ajoutés pour plus de lisibilité

```
  {
   "protected":"<integrity-protected shared header contents>",
   "unprotected":<non-integrity-protected shared header contents>,
   "recipients":[
    {"header":<per-recipient unprotected header 1 contents>,
     "encrypted_key":"<encrypted key 1 contents>"},
    ...
    {"header":<per-recipient unprotected header N contents>,
     "encrypted_key":"<encrypted key N contents>"}],
   "aad":"<additional authenticated data contents>",
   "iv":"<initialization vector contents>",
   "ciphertext":"<ciphertext contents>",
   "tag":"<authentication tag contents>"
  }
```

Exemple

```
{
  "recipients": [
    {
      "encrypted_key": "dYOD28kab0Vvf4ODgxVAJXgHcSZICSOp8M51zj
          wj4w6Y5G4XJQsNNIBiqyvUUAOcpL7S7-cFe7Pio7gV_Q06WmCSa-
          vhW6me4bWrBf7cHwEQJdXihidAYWVajJIaKMXMvFRMV6iDlRr076
          DFthg2_AV0_tSiV6xSEIFqt1xnYPpmP91tc5WJDOGb-wqjw0-b-S
          1laS11QVbuP78dQ7Fa0zAVzzjHX-xvyM2wxj_otxr9clN1LnZMbe
          YSrRicJK5xodvWgkpIdkMHo4LvdhRRvzoKzlic89jFWPlnBq_V4n
          5trGuExtp_-dbHcGlihqc_wGgho9fLMK8JOArYLcMDNQ",
```

```
        "header": {
          "alg": "RSA1_5",
          "kid": "frodo.baggins@hobbiton.example"
        }
      },
      {
        "encrypted_key": "ExInT0io9BqBMYF6-maw5tZlgoZXThD1zWKsHi
            xJuw_elY4gSSId_w",
        "header": {
          "alg": "ECDH-ES+A256KW",
          "kid": "peregrin.took@tuckborough.example",
          "epk": {
            "kty": "EC",
            "crv": "P-384",
            "x": "Uzdvk3pi5wKCRc1izp5_r0OjeqT-I68i8g2b8mva8diRhs
                E2xAn2DtMRb25Ma2CX",
            "y": "VDrRyFJh-Kwd1EjAgmj5Eo-CTHAZ53MC7PjjpLioy3ylEj
                I1pOMbw91fzZ84pbfm"
          }
        }
      },
      {
        "encrypted_key": "a7CclAejo_7JSuPB8zeagxXRam8dwCfmkt9-Wy
            TpS1E",
        "header": {
          "alg": "A256GCMKW",
          "kid": "18ec08e1-bfa9-4d95-b205-2b4dd1d4321d",
          "tag": "59Nqh1LlYtVIhfD3pgRGvw",
          "iv": "AvpeoPZ9Ncn9mkBn"
        }
      }
    ],
    "unprotected": {
      "cty": "text/plain"
    },
    "protected": "eyJlbmMiOiJBMTI4Q0JDLUhTMjU2In0",
    "iv": "VgEIHY20EnzUtZFl2RpB1g",
    "ciphertext": "ajm2Q-OpPXCr7-MHXicknb1lsxLdXxK_yLds0KuhJzfWK
        04SjdxQeSw2L9mu3a_k1C55kCQ_3xlkcVKC5yr__Is48VOoK0k63_QRM
        9tBURMFqLByJ8vOYQX0oJW4VUHJLmGhF-tVQWB7Kz8mr8zeE7txF0MSa
        P6ga7-siYxStR7_G07Thd1jh-zGT0wxM5g-VRORtq0K6AXpLlwEqRp7p
        kt2zRM0ZAXqSpe1O6FJ7FHLDyEFnD-zDIZukLpCbzhzMDLLw2-8I14FQ
        rgi-iEuzHgIJFIJn2wh9Tj0cg_kOZy9BqMRZbmYXMY9YQjorZ_P_JYG3
        ARAIF3OjDNqpdYe-K_5Q5crGJSDNyij_ygEiItR5jssQVH2ofDQdLCht
        azE",
    "tag": "BESYyFN7T09KY7i8zKs5_g"
  }
```

Syntaxe de sérialisation JWE JSON aplatie

La syntaxe de la sérialisation JWE JSON aplatie est basée sur la syntaxe générale, mais l'aplatit, l'optimisant pour le cas du destinataire unique.

Les sauts de ligne ajoutés pour plus de lisibilité

```
    {
     "protected":"<integrity-protected header contents>",
     "unprotected":<non-integrity-protected header contents>,
```

```
   "header":<more non-integrity-protected header contents>,
   "encrypted_key":"<encrypted key contents>",
   "aad":"<additional authenticated data contents>",
   "iv":"<initialization vector contents>",
   "ciphertext":"<ciphertext contents>",
   "tag":"<authentication tag contents>"
  }
```

Exemple

```
{
  "protected": "eyJhbGciOiJBMTI4S1ciLCJraWQiOiI4MWIyMDk2NS04Mz
      MyLTQzZDktYTQ2OC04MjE2MGFkOTFhYzgiLCJlbmMiOiJBMTI4R0NNIn
      0",
  "encrypted_key": "4YiiQ_ZzH76TaIkJmYfRFgOV9MIpnx4X",
  "aad": "WyJ2Y2FyZCIsW1sidmVyc2lvbiIse30sInRleHQiLCI0LjAiXSxb
      ImZuIix7fSwidGV4dCIsIk1lcmlhZG9jIEJyYW5keWJ1Y2siXSxbIm4i
      LHt9LCJ0ZXh0IixbIkJyYW5keWJ1Y2siLCJNZXJpYWRvYyIsIk1yLiIs
      IiJdXSxbImJkYXkiLHt9LCJ0ZXh0IiwiVEEgMjk4MiJdLFsiZ2VuZGVy
      Iix7fSwidGV4dCIsIk0iXV1d",
  "iv": "veCx9ece2orS7c_N",
  "ciphertext": "Z_3cbr0k3bVM6N3oSNmHz7Lyf3iPppGf3Pj17wNZqteJ0
      Ui8p74SchQP8xygM1oFRWCNzeIa6s6BcEtp8qEFiqTUEyiNkOWDNoF14
      T_4NFqF-p2Mx8zkbKxI7oPK8KNarFbyxIDvICNqBLba-v3uzXBdB89fz
      OI-Lv4PjOFAQGHrgv1rjXAmKbgkft9cB4WeyZw8MldbBhc-V_KWZslrs
      LNygon_JJWd_ek6LQn5NRehvApqf9ZrxB4aq3FXBxOxCys35PhCdaggy
      2kfUfl2OkwKnWUbgXVD1C6HxLIlqHhCwXDG59weHrRDQeHyMRoBljoV3
      X_bUTJDnKBFOod7nLz-cj48JMx3SnCZTpbQAkFV",
  "tag": "vOaH_Rajnpy_3hOtqvZHRA"
}
```

Lire Sérialisations en ligne: https://riptutorial.com/fr/jwt/topic/5988/serialisations

Crédits

S. No	Chapitres	Contributeurs
1	Commencer avec jwt	Alex, Community, Florent Morselli, Nate Barbettini, pedrofb, RamenChef, Set
2	Invalider les jetons Web Json	pedrofb
3	Sérialisations	Florent Morselli