JW*

by Roland Hedberg

JOSE Working Group

- Integrity-protected object format
- Confidentiality-protected object format
- A format for expressing keys

JW*

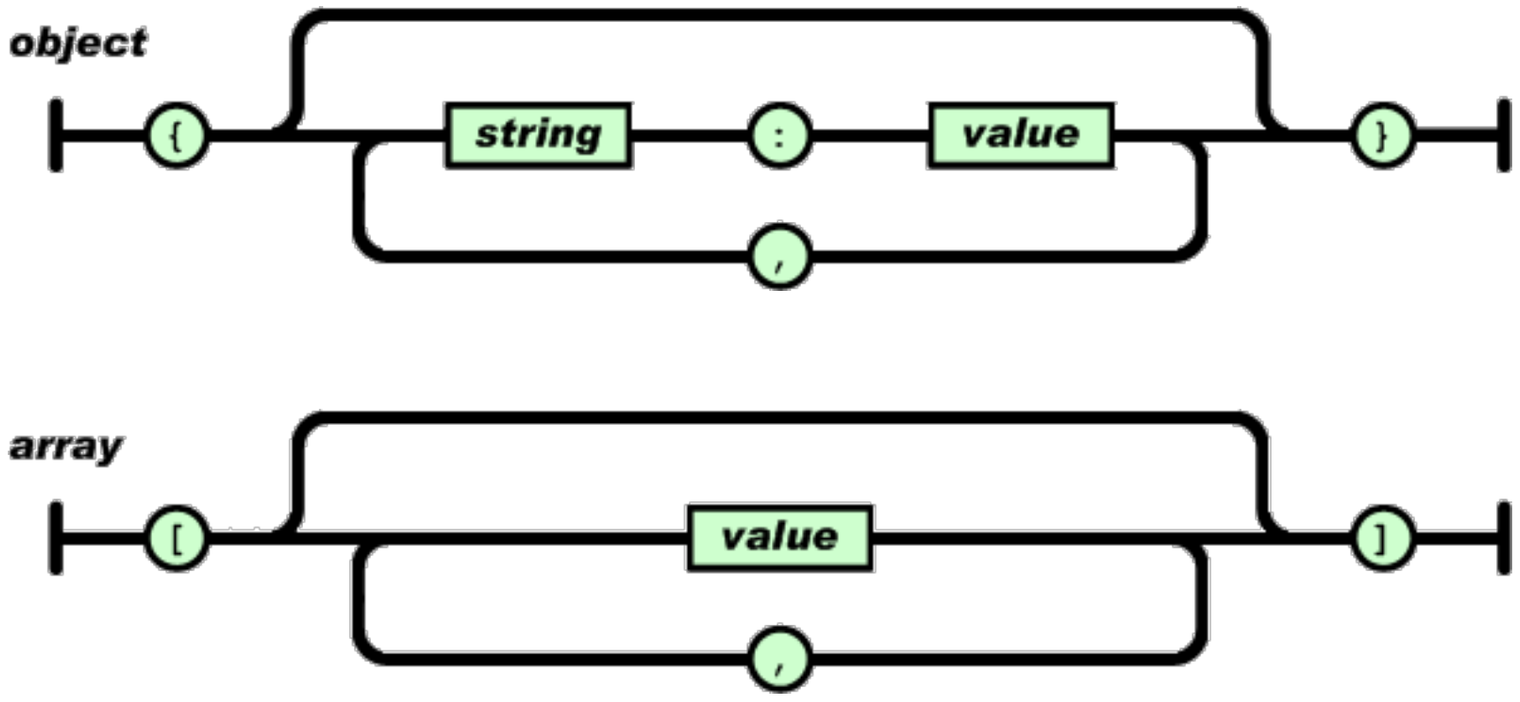
- JWT - JSON Web Token
- JWS - JSON Web Signature
- JWE - JSON Web Encryption
- JWK - JSON Web Key
- JWA - JSON Web Algorithms

JWT - RFC 7519

- JSON Web Token (JWT)
  - Compact
  - URL-safe
  - Transport format
  - JWTs represent a set of claims as a JSON object that is encoded in a JWS and/or JWE structure.

JSON

Java Script Object Notation



JWT Usage

- HTTP Authorization headers

- URI query parameters

Representation

- A sequence of URL-safe parts separated by period ’.’ characters.

Registered Claim names

- iss (Issuer)
- sub (Subject)
- aud (Audience)
- exp (Expiration Time)
- nbf (Not Before)
- iat (Issued at)
- jti (JWT ID)

JSON Signing and Encryption (JOSE)

JOSE header parameters

- **alg** (Algorithm) - REQUIRED
- jku (JWK Set URL)
- jwk (JSON Web Key)
- kid (Key ID)
- x5u (X.509 URL)
- x5c (X.509 Certificate Chain)
- x5t (X.509 Certificate SHA-1 Thumbprint)
- x5t#S256 (X.509 Certificate SHA-256 Thumbprint)
- typ (Media type)
- cty (Content type)
- crit (Critical)

Unsecure JWT

- header: {”alg”: ”none”}
- message: payload

<header>.<message>

Create a JWT

A. Create a Claims Set (== create a JSON object)

B. Message = Base64url encoded UTF-8 representation of the JSON object

C. Create a JOSE Header (JWS or JWE header)

D. Create JWS or JWE

E. If nested use the JWS/JWE as message, include cty=”JWT” in header and go from (3)

F. else, resulting JWT is the JWS or JWE

Code example

JWK

A JSON Web Key (JWK) is a JavaScript Object Notation (JSON) [RFC7159] data structure that represents a cryptographic key.

Example JWK


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

*Elliptic curve cryptography

For elliptic-curve-based protocols, it is assumed that finding the discrete logarithm of a random elliptic curve element with respect to a publicly known base point is infeasible.

A 256-bit ECC public key should provide comparable security to a 3072-bit RSA public key.

JWK parameters

- **kty (Key Type) - REQUIRED**
- use (Public Key Use)
    - ’sig’, ’enc’
- key_ops (Key Operations)
    - ’sign’, ’verify’, ’encrypt’, ’decrypt’, ’wrapkey’, ’unwrapKey’, ’deriveKey’, ’deriveBits’
- alg (Algorithm)
- kid (Key ID)
- x5u (X.509 URL)
- x5c (X.509 Certificate Chain)
- x5t (X.509 Certificate SHA-1 Thumbprint)
- x5t#S256 (X.509 Certificate SHA-256 Thumbprint)

JSON Web Key Set (JWKS)

- A JSON object that represents a set of JWKs

- The JSON object must have a ”keys” member.

Code example

JWS

JSON Web Signature (JWS) represents content secured with digital signatures or Message Authentication Codes (MACs) using JavaScript Object Notation (JSON) based data structures.

JWS header parameters

- **alg** (algorithm) - REQUIRED
- jku (JWK Set URL)
- jwk (JSON Web Key)
- kid (Key ID)
- x5u (X.509 ULR)
- x5c (X.509 Certificate Chain)
- x5t (X.509 Certificate SHA-1 thumbprint)
- x5t#S256 (X.509 Certificate SHA-256 Thumbprint)
- typ (Type, MIME Media Type of the JWS)
- cty (Content Type of payload)
- crit (Critical extensions)

JWS components

- header
  - parameters describing the cryptographic operations and parameters employed
- payload
  - message
- signature
    - Digital signature or MAC over the JWS Protected Header and the JWS Payload

- JWS compact serialization:

BASE64URL(UTF8(Protected Header)) ”.” BASE64URL(Payload) ”.” BASE64URL(Signature)

2 serializations

- Compact serialization - compact, URL-safe
- JSON serialization
    - general
    - flattened

JWS JSON Serialization (general)


```
{
  "payload":
    "eyJpc3MiOiJqb2UiLA0KICJleHAiOjEzMDA4MTkzODAsDQogImh0dHA6Ly9leG
     FtcGxlLmNvbS9pc19yb290Ijp0cnVlfQ",
  "signatures":[
   {"protected":"eyJhbGciOiJSUzI1NiJ9",
    "header": {"kid":"2010-12-29"},
    "signature":
     "cC4hiUPoj9Eetdgtv3hF80EGrhuB__dzERat0XF9g2VtQgr9PJbu3XOiZj5RZ
      mh7AAuHIm4Bh-0Qc_lF5YKt_O8W2Fp5jujGbds9uJdbF9CUAr7t1dnZcAcQjb
      KBYNX4BAynRFdiuB--f_nZLgrnbyTyWzO75vRK5h6xBArLIARNPvkSjtQBMHl
      b1L07Qe7K0GarZRmB_eSN9383LcOLn6_dO--xi12jzDwusC-eOkHWEsqtFZES
      c6BfI7noOPqvhJ1phCnvWh6IeYI2w9QOYEUipUTI8np6LbgGY9Fs98rqVt5AX
      LIhWkWywlVmtVrBp0igcN_IoypGlUPQGe77Rw"},
  {"protected":"eyJhbGciOiJFUzI1NiJ9",
   "header": {"kid":"e9bc097a-ce51-4036-9562-d2ade882db0d"},
   "signature":
    "DtEhU3ljbEg8L38VWAfUAqOyKAM6-Xx-F4GawxaepmXFCgfTjDxw5djxLa8I
     SlSApmWQxfKTUJqPP3-Kg6NU1Q"}]
```

JWS JSON Serialization (Flattened)

```
{
  "payload":
   "eyJpc3MiOiJqb2UiLA0KICJleHAiOjEzMDA4MTkzODAsDQogImh0dHA6Ly9
   leGFtcGxlLmNvbS9pc19yb290Ijp0cnVlfQ”,
  "protected":"eyJhbGciOiJFUzI1NiJ9",
  "header":
   {"kid":"e9bc097a-ce51-4036-9562-d2ade882db0d"},
  "signature":
   "DtEhU3ljbEg8L38VWAfUAqOyKAM6XxF4GawxaepmXFCgfTjDxw5djxLa8IS
   lSApm WQxfKTUJqPP3-Kg6NU1Q"
}
```

optimized for single digital signature

Mandatory to implement algorithms

- Only HMAC SHA-256 ("HS256") and "none" MUST be implemented by conforming JWT implementations
- It is RECOMMENDED that implementations also support RSASSA-PKCS1-v1_5 with the SHA-256 hash algorithm ("RS256") and ECDSA using the P-256 curve and the SHA-256 hash algorithm ("ES256").

Code example

JWE

JSON Web Encryption (JWE) represents encrypted content using JavaScript Object Notation (JSON) based data structures.

JWE Header parameters

- **alg** (Algorithm)
- **enc (Encryption Algorithm)**
- **zip (Compression Algorithm)**
- jku (JWK Set URL)
- jwk (JSON Web Key)
- kid (Key ID)
- x5u (X.509 ULR)
- x5c (X.509 Certificate Chain)
- x5t (X.509 Certificate SHA-1 thumbprint)
- x5t#S256 (X.509 Certificate SHA-256 Thumbprint)
- typ (Type, MIME Media Type of the JWS)
- cty (Content Type of payload)
- crit (Critical extensions)

Difference between alg and enc

- CEK

  A symmetric key for the AEAD algorithm used to encrypt the plaintext to produce the ciphertext and the Authentication Tag.

- JWE Encrypt Key

  Encrypted Content Encryption Key value.

- **alg**

  Encryption algorithm for encrypting the CEK to produce the JEK

- **enc**

  Content encryption algorithm

JWE components

- Header
  - parameters describing the cryptographic operations and parameters employed
- Encrypted key
  - Encrypted Content Encryption Key (CEK) value
- Initialization vector
  - Initialization Vector value used when encrypting the plaintext
- *AAD*
  - Additional value to be integrity protected (header+possible extra)
- Cipher text
  - Ciphertext value resulting from authenticated encryption of the plaintext with additional authenticated data
- Authentication tag
  - Authentication Tag value resulting from authenticated encryption of the plaintext with additional authenticated data

Implementation requirements

- Support for encrypted JWTs is OPTIONAL
- RSAES-PKCS1-v1_5 with 2048-bit keys ("RSA1_5"), AES Key Wrap with 128- and 256-bit keys ("A128KW" and "A256KW"), and the composite authenticated encryption algorithm using AES-CBC and HMAC SHA-2 ("A128CBC-HS256" and "A256CBC-HS512") MUST be implemented by conforming implementations.

Code example

JWA

The JSON Web Algorithms (JWA) specification registers cryptographic algorithms and identifiers to be used with the JSON Web Signature (JWS), JSON Web Encryption (JWE), and JSON Web Key (JWK) specifications.

Links to documents

- The JavaScript Object Notation (JSON) Data Interchange Format
- JSON Web Token (JWT)
- JSON Web Key (JWK)
- JSON Web Signature (JWS)
- JSON Web Encryption (JWE)
- JSON Web Algorithms (JWA)