python-jose Documentation

Release 0.2.0

Michael Davis

September 16, 2020

Contents

A JOSE implementation in Python

The JavaScript Object Signing and Encryption (JOSE) technologies - JSON Web Signature (JWS), JSON Web Encryption (JWE), JSON Web Key (JWK), and JSON Web Algorithms (JWA) - collectively can be used to encrypt and/or sign content using a variety of algorithms. While the full set of permutations is extremely large, and might be daunting to some, it is expected that most applications will only use a small set of algorithms to meet their needs.

CHAPTER 1

Contents

1.1 JSON Web Signature

JSON Web Signatures (JWS) are used to digitally sign a JSON encoded object and represent it as a compact URL-safe string.

1.1.1 Supported Algorithms

The following algorithms are currently supported.

Algorithm Value	Digital Signature or MAC Algorithm
HS256	HMAC using SHA-256 hash algorithm
HS384	HMAC using SHA-384 hash algorithm
HS512	HMAC using SHA-512 hash algorithm
RS256	RSASSA using SHA-256 hash algorithm
RS384	RSASSA using SHA-384 hash algorithm
RS512	RSASSA using SHA-512 hash algorithm
ES256	ECDSA using SHA-256 hash algorithm
ES384	ECDSA using SHA-384 hash algorithm
ES512	ECDSA using SHA-512 hash algorithm

1.1.2 Examples

Signing tokens

```
>>> from jose import jws
>>> signed = jws.sign({'a': 'b'}, 'secret', algorithm='HS256')
'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhIjoiYiJ9.jiMyrsmD8AoHWeQgmxZ5yq8z0lXS67_QGs52AzC8Ru8'
```

Verifying token signatures

```
>>> jws.verify(signed, 'secret', algorithms=['HS256'])
{'a': 'b'}
```

1.2 JSON Web Token

JSON Web Tokens (JWT) are a JWS with a set of reserved claims to be used in a standardized manner.

1.2.1 JWT Reserved Claims

Claim	Name	Format	Usage
'exp'	Expiration	int	The time after which the token is invalid.
'nbf'	Not Before	int	The time before which the token is invalid.
'iss'	Issuer	str	The principal that issued the JWT.
'aud'	Audience	str or list(str)	The recipient that the JWT is intended for.
'iat'	Issued At	int	The time at which the JWT was issued.

1.3 JSON Web Key

JSON Web Keys (JWK) are a JSON data structure representing a cryptographic key.

1.3.1 Examples

Verifying token signatures

```
>>> from jose import jwk
>>> from jose.utils import base64url_decode
>>>
>>> token = "eyJhbGciOiJIUzI1NiIsImtpZCI6IjAxOGMwYWU1LTRkOWItNDcxYi1iZmQ2LWVlZjMxNGJjNzAzNyJ9.SXTigJ
>>> hmac_key = {
    "kty": "oct",
    "kid": "018c0ae5-4d9b-471b-bfd6-eef314bc7037",
    "use": "sig",
    "alg": "HS256",
    "k": "hJtXIZ2uSN5kbQfbtTNWbpdmhkV8FJG-Onbc6mxCcYg"
}
>>>
>>> key = jwk.construct(hmac_key)
>>>
>>> message, encoded_sig = token.rsplit('.', 1)
>>> decoded_sig = base64url_decode(encoded_sig)
>>> key.verify(message, decoded_sig)
```

1.3.2 Note

python-jose requires the use of public keys, as opposed to X.509 certificates. If you have an X.509 certificate that you would like to convert to a public key that python-jose can consume, you can do so with openssl.

```
> openssl x509 -pubkey -noout < cert.pem
```

1.4 JSON Web Encryption

JSON Web Encryption (JWE) are used to encrypt a payload and represent it as a compact URL-safe string.

1.4.1 Supported Content Encryption Algorithms

The following algorithms are currently supported.

Encryption Value	Encryption Algorithm, Mode, and Auth Tag
A128CBC_HS256	AES w/128 bit key in CBC mode w/SHA256 HMAC
A192CBC_HS384	AES w/128 bit key in CBC mode w/SHA256 HMAC
A256CBC_HS512	AES w/128 bit key in CBC mode w/SHA256 HMAC
A128GCM	AES w/128 bit key in GCM mode and GCM auth tag
A192GCM	AES w/192 bit key in GCM mode and GCM auth tag
A256GCM	AES w/256 bit key in GCM mode and GCM auth tag

1.4.2 Supported Key Management Algorithms

The following algorithms are currently supported.

Algorithm Value	Key Wrap Algorithm
DIR	Direct (no key wrap)
RSA1_5	RSAES with PKCS1 v1.5
RSA_OAEP	RSAES OAEP using default parameters
RSA_OAEP_256	RSAES OAEP using SHA-256 and MGF1 with SHA-256
A128KW	AES Key Wrap with default IV using 128-bit key
A192KW m	AES Key Wrap with default IV using 192-bit key
A256KW	AES Key Wrap with default IV using 256-bit key

1.4.3 Examples

Encrypting Payloads

```
>>> from jose import jwe
>>> jwe.encrypt('Hello, World!', 'asecret128bitkey', algorithm='dir', encryption='A128GCM')
'eyJhbGciOiJkaXIiLCJlbmMiOiJBMTI4R0NNIn0..McILMB3dYsNJSuhcDzQshA.OfX9H_mcUpHDeRM4IA.CcnTWqaqxNsjT4eCa
```

Decrypting Payloads

```
>>> from jose import jwe
>>> jwe.decrypt('eyJhbGciOiJkaXIiLCJlbmMiOiJBMTI4R0NNIn0..McILMB3dYsNJSuhcDzQshA.OfX9H_mcUpHDeRM4IA.C
'Hello, World!'
```

CHAPTER 2

APIs

2.1 JWS API

2.2 JWT API

2.3 JWK API

2.4 JWE API

CHAPTER 3

Principles

This is a JOSE implementation that is fully compatible with Google App Engine which requires the use of the PyCrypto library.

CHAPTER 4

Thanks

This library was originally based heavily on the work of the guys over at PyJWT.