What is JOSE

Jim Schaad
Co-chair JOSE
August Cellars

Overview

- Use JSON for data structure representations
- Try and meet the goal of easy to implement and use
- Allow for complex uses
- Allow for arbitrary body content

History

- Came out of the OpenID Forum
- Generalization of the JSON Web Token

Provided Services

- Signature
- Message Authentication Code
- Encryption
- Public Key Format
- Private Key Format

Signing

- Signature and MAC treated the same
- MAC only with pre-shared secret
- No canonicalization
- Multiple Serializations
  - URL safe version
  - JSON object serialization

Signing Structure

- Three sections
  - Header
    - JSON String
    - Short member names for compatness
  - Body
    - Arbitrary Content
    - Attached or Detached Content
  - Signature Value

Signature Header

- Example Header
  - {“alg”:”RS256”,
    “jku”:”http://keys.example.com/~jose/SigningKeys”,
    “kid”:”Key#1”, “typ”:”JWS”
    }
- Header Types
  - Algorithm Information
  - Keys/Key Locator Information
  - Meta Data

Algorithm Headers

- Header members
  - alg
    - Signature Algorithm
    - MAC algorithm
    - String containing all information

Signature/MAC Algorithms

- Signature Algorithms
  - RSA 1.5 with SHA-256, SHA-384, SHA-512
  - ECDS with SHA-256, SHA-384, SHA-512
  - SHOULD – RSA and ECDS with SHA-256
- MAC Algorithms
  - HMAC with SHA-256, SHA-384, SHA-512
  - MUST – HMAC with SHA-256
- Plain Text Algorithms

Key Location Methods

- Header members
  - jku – URL to JSON Web Keyset
  - jwk – Embedded JSON Web Key
  - x5u – URL to X.509 Certificate/Cert Chain
    - PEM encoded
  - x5t – SHA-1 thumbprint of X.509 Certificate
  - x5c – embedded X.509 Certificate/Cert Chain
  - kid – key identifier

Meta Data

- Members
  - typ – "JWS" – JSON Web Signature object
  - crit – array of strings to identify must handle members for extensions
  - ctyp – Inner content identification
- Proposed Members
  - aed – Application specific meta data

JSON Issues

- JSON String Delimitation with parser
  - {“tag”:”value”}ABCD
  - Not all parsers handle correctly
- JSON Allows multiple Members in a lexical scope
  - {“tag”:”value1”,”tag”:”value2”}
  - SHOULD be unique

URL Safe Encoding

- Base64URL encoded Header
- Period character
- Base64URL encoded Body
- Period char
- Base64URL encoded Signature

- Hash the first three elements in the encoding

URL Safe Example

eyJ0eXAiOiJKV1QiLA0KICJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJqb2UiLA0KICJleHAiOjEzMDA4MTkzODAsDQogImh0dHA6Ly9leGFtcGxlLmNvbS9pc19yb290Ijp0cnVlfQ.dBjftJeZ4CVP-mB92K27uhbUJU1p1r_wW1gFWFOEjXk

Line wraps are absent in real world

JSON Based Encoding

- Members
  - recipients – array of signature or MAC headers in an object
    - header – Base64URL encoded JSON header string
    - signature – Base64URL encoded signature value
  - payload – Base64URL encoded content

JSON Based Encoding

{"recipients":[
{"header":"eyJhbGciOiJSUzI1NiJ9",

"signature":"cC4hiUPoj9Eetdgtv3hF80EGrhuB__dzERat0XF9g2
VtQgr9PJbu3XOiZj5RZmh7AAuHIm4Bh0Qc_lF5YKt_O8W2
Fp5jujGbds9uJdbF9CUAr7t1dnZcAcQjbKBYNX4BAynRFdiuB--f_nZLgrnbyTyWzO75vRK5h6xBArLIARNPvkSjtQBMHlb1L07Qe7K0Gar
ZRmB_eSN9383LcOLn6_dO--xi12jzDwusC-eOkHWEsqtFZESc6BfI7n
oOPqvhJ1phCnvWh6IeYI2w9QOYEUipUTI8np6LbgGY9Fs98r

Open Issues

- Inclusion of RSA-PSS in the list of algorithms

Encryption

- Multiple Serializations
  - URL safe version
  - JSON object serialization

Encryption Structure

- Five sections
  - Header
    - JSON String
    - Short member names for compactness
  - Encrypted Key
  - Initialization Vector (IV)
  - Body
    - Arbitrary Content
    - Attached or Detached Content
  - Authentication Tag

Encryption Header

- Example Header
  - {“alg”:”RSA1_5”,
    “enc”:”A128GCM”
    “jku”:”http://keys.example.com/~jose/EncryptionKey”,
    “kid”:”Key#1”, “typ”:”JWE”
    }
- Header Types
  - Algorithm Information
  - Keys/Key Locator Information
  - Meta Data

Is it JWS or JWE

- If present use “typ” member
- Else if present use “alg” member
  - Based on the algorithm, decide if it is JWS or JWE
  - If algorithm is not understood – then exception
- Else if present use “enc” member
  - JWE

Key Management Algorithms

- Member ‘alg’
- Key Transport
  - RSA v1.5, RSA-OAEP
- Key Agreement
  - ECDH-ES + KDF x {none, AES KeyWrap 128, 256}
- Key Encryption (symmetric)
  - AES Key Wrap 128, 256

Content Encryption

- Requires the use of AEAD algorithms
- AES 128 GCM
- AES 256 GCM
- AES 128 CBC + HMAC SHA-256
- AES 256 CBC + HMAC SHA-512

Review where Algorithms Go

- Example Header
  - {“alg”:”RSA1_5”,
    “enc”:”A128GCM”
    “jku”:”http://keys.example.com/~jose/EncryptionKey”,
    “kid”:”Key#1”, “typ”:”JWE”
    }
  - alg – Key Management Algorithm
  - enc – Content Encryption Algorithm

Zip and Key Derivation

- ‘zip’ – currently only support Deflate
- KDF parameters
  - epk – ephemeral public key – embedded JWK
  - apu, apv, epu, epv – used for key agreement algorithm key derivation functions

URL Safe Encoding

- Base64URL encoded Header
- Period character
- Base64URL encoded Encrypted Key
- Period character
- Baes64URL encoded IV
- Period character
- Base64URL encrypted Body
- Period character
- Base64URL authentication tag

- Authenticated Data first 5 elements

JSON Based Encoding

- Members
  - recipients – array of recipient information
    - header – Base64URL encoded JSON header string
    - encrypted_key – base64URL encoded
    - Integrity_value – base64URL encoded
  - enitialization_vector – base64URL encoded
  - ciphertext – base64URL encoded

JSON Keys

- Not doing certificates
- Keys can have attributes
- Allows for single keys and arrays of keys
- Allow for private key fields

JSON Structure

- Members
  - kty – key type
    - RSA, ECDS, binary
  - use – ‘sign’, ‘enc’
  - alg – which algorithm to use
  - kid – key identifier

Example

```
{"keys": [
   {"kty":"EC", "crv":"P-256",
    "x":“base64 value",    "y":“base64
value",          "use":"enc", "kid":"1"},
   {"kty":"RSA", "n": “base64 value",
      "e":"AQAB", "alg":"RS256",
      "kid":"2011-04-29"}
] }
```

Questions