JOSE Key Wrapping

draft-barnes-jose-key-wrapping

Two requirements from WebCrypto

Provide a format for an encrypted key, possibly with attributes

Allow key wrapping with general AEAD algorithms (e.g., GCM)

Two requirements from WebCrypto

Provide a format for an encrypted key, possibly with attributes

Wrap JWK in JWE

Allow key wrapping with general AEAD algorithms (e.g., GCM)

???

A tale of two wrapped keys

JWE wrapped key

```
{
  "recipients": [{
    "alg": "A128KW",
    "kid": "thatone",
    "encrypted_key": /* key */
  }],
  /* Remainder of JWE */
}
```

JWK within JWE

```
{
  "alg": "dir",
  "recipients": [{
    "enc": "A128KW",
    "kid": "thatone",
  }],
  "ciphertext": /* key */
}
```

A tale of two wrapped keys

JWE wrapped key

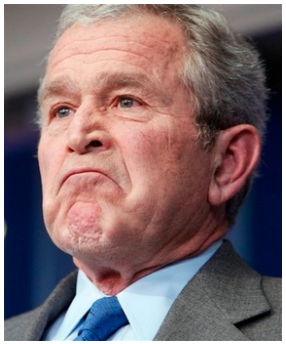
```
{
  "recipients": [{
    "alg": "A128KW",
    "kid": "thatone",
    "encrypted_key": /* key */
  }],
  /* Remainder of JWE */
}
```

JWK within JWE

```
{
  "alg": "dir",
  "recipients": [{
    "enc": "A128KW",
    "kid": "thatone",
  }],
  "ciphertext": /* key */
}
```

No [obvious way to do] AEAD Key Wrap

Distinction without difference!



Wrapped Key == JWE(JWK)

... always and everywhere
... even in JWE

Basic Example

JWE wrapped key

```
{
  "enc": "A128GCM",
  "alg": "A128KW",
  "kid": "thatone",
  "encrypted_key": /* key */
  "initialization_vector": "...",
  "ciphertext": "...",
  "authentication_tag": "..."
}
```

JWK within JWE

```
{
  "enc": "A128GCM",
  "key": {
    "enc": "A128KW",
    "kid": "thatone",
    "ciphertext": /* key */
  },
  "initialization_vector": "...",
  "ciphertext": "...",
  "authentication_tag": "..."
}
```

GCM Key Wrapping

JWE wrapped key

```
/* Not allowed by syntax */
```

JWK within JWE

```
{
  "enc": "A128GCM",
  "key": {
    "enc": "A128GCM",
    "kid": "thatone",
    "initialization_vector": "...",
    "ciphertext": /* key */
    "authentication_tag": "..."
  },
  "initialization_vector": "...",
  "ciphertext": "...",
  "authentication_tag": "..."
}
```

Compact Serialization

JWE wrapped key

```
{
  "enc": "A128GCM",
  "alg": "A128KW",
  "kid": "thatone",
  "encrypted_key": /* key */
  "initialization_vector": "...",
  "ciphertext": "...",
  "authentication_tag": "..."
}
```

```
base64({"enc":"A128GCM","alg":"
A128KW","kid":"thatone"})
. encrypted_key
. initialization_vector
. ciphertext
. authentication_tag
```

JWK within JWE

```
{
  "enc": "A128GCM",
  "key": {
    "enc": "A128KW",
    "kid": "thatone",
    "ciphertext": /* key */
  },
  "initialization_vector": "...",
  "ciphertext": "...",
  "authentication_tag": "..."
}
```

```
base64({"enc":"A128GCM","key":
{"enc":"A128KW","kid":"thatone"}})
. key.ciphertext
. initialization_vector
. ciphertext
. authentication_tag
```

Compact Serialization [Overhead]

```
base64({"enc":"A128GCM","alg":"
A128KW","kid":"thatone"})
. encrypted_key
. initialization_vector
. ciphertext
. authentication_tag
```

```
base64({"enc":"A128GCM","key":
{"enc":"A128KW","kid":"thatone"}})
. key.ciphertext
. initialization_vector
. ciphertext
. authentication_tag
```

```
  eyJlbmMiOiJBMTI4R0NNIiwiYWxnIjoiQTEyOEtXIiwia2lkIjoidGhhdG9uZSJ9Cg
- eyJlbmMiOiJBMTI4R0NNIiwia2V5Ijp7ImVuYyI6IkExMjhLVyIsImtpZCI6InRoYXRvbmUifX0K
==========================================================================
```

10 octets

Compact Serialization [GCM]

JWE wrapped key

```
/* Not allowed by syntax */
```

JWK within JWE

```
{
  "enc": "A128GCM",
  "key": {
    "enc": "A128GCM",
    "kid": "thatone",
    "initialization_vector": "...",
    "ciphertext": /* key */,
    "authentication_tag": "..."
  },
  "initialization_vector": "...",
  "ciphertext": "...",
  "authentication_tag": "..."
}
```

```
base64({
  "enc": "A128GCM",
  "key": {
    "enc":"A128GCM",
    "kid":"thatone",
    "initialization_vector":
"...",
    "authentication_tag": "..."
  }
})
. key.ciphertext
. initialization_vector
. ciphertext
. authentication_tag
```

Large header, but now possible

Incremental complexity

Use "cty" to scale complexity / compactness

No "cty": octet string => symmetric key, !attrs

"cty": "application/jwk+json"

"cty": "application/jwk+cbor" [?]

Key Management for MAC

Having a consolidated wrapped key object makes this trivial

Just add a field "key": { ... } to JWS/MAC

Multiple recipients == multiple "key" values
(for both JWE and JWS/MAC)

Proposal Summary

Wrapped key in JWE == JWE(JWK)

Add "key" field to JWE (and possibly JWS/MAC)

Adjust compactification algorithm to use key.ciphertext instead of encrypted_key

Remove "alg" field from JWE (since "alg" becomes "key.enc")

Further down the rabbit hole...

Object Model

```
key := JWE / kid / jwk / jku / x5c / x5t
algorithm := name (parameter)+
JWE   := algorithm authenticated_attrs? key+ data icv
JWMAC := JWE
JWS   := data (algorithm signed_attrs? key icv)+


/*
-- Define the JSON format based on an object model
-- Define a compactification of JWE / JWMAC / JWS
   -- For single recipient, essentially no change
*/
```