Use cases + JSMS

Richard Barnes

Use Cases Reminder

- JSON Web Token
  - Mainly signing, and encyption
  - Compact, URL-safe serialization
- XMPP end to end security
  - Mainly encryption, and signing
  - Separation of wrapped keys from encrypted content
  - Base64 != base64url
- ALTO[?]: Signing over JSON objects
- WebCrypto[?]: Cryptographic structures that go over the wire (public keys, wrapped keys)
- Other suggestions welcome!

JWT Security Token Use Case [MJ]

- URL-safe representation required
  - Tokens may be used as URI query parameters
- Compactness required
  - Some browsers limit URLs to 2048 chars or less
- Simplicity required
  - Goal is widespread adoption

JSMS

- Written to highlight some design choices in JW*
  - In particular, several that differ from CMS
- Simplified profile of CMS, encoded in JSON
  - No signed/authenticated attributes
  - No password-based key wrapping
  - Lots of optional features stripped
- Crypto-compatible with (a profile of) CMS

JSMS Example (Encrypted)

```
{
  "version": 1,
  "type": "encrypted",
  "content": "0nkXCLOVxM2oNJOsDCwASLTODIMVZQE=",
  "algorithm": {
    "name": "aes128-ccm",
    "n": "LTR8s7KKbd1QlQ==",
    "m": 8
  },
  "keys": [{
    "type": "transport",
    "algorithm": "rsaes-oaep",
    "encryptedKey": "AbAx...mgOKJv-",
    "recipientKey": {
      "type": "rsa",
      "n": "AfWGin...yq7_v_c_",
      "e": "AQAB",
    }
  }]
}
```

JSMS Example (Authenticated)

```
{
  "version": 1,
  "type": "authenticated",
  "algorithm": "hs256",
  "content": "QXR0YWNrIGF0IGRhd24h",
  "mac": "990xwhrsX-COXUN0uF09HUHLU2CjdneeMqTtM4sGVDY=",
  "keys": [{
    "type": "encryption",
    "algorithm": "aes",
    "encryptedKey": "Dbf2O_ZIX0_Zfj-0aU6zQjn3xixj6vm7LVX
                     XFDdX4xqie5bZUS1nnstIPYOyzxNx9Udt-J
                     LZZh-zM8A_FbsZ8zAibdJ3EPyd",
    "KEKIdentifier": "HK1RA8AQwcI="
  }]
}
```

```
{"v":1,"t":"au","a":"hs256","ki":"HK1RA8AQwcI=",
     "mac": "PMVmhmrgbj-KNybfMqHu4ySJ0GnVrwe11MKpiuuGlIQ="}
```

Differences from JW*

- It's JSON ☺
- MAC is handled separately from signing
  - Gets wrapped keys
- Clearer structure and processing instructions
- No integrity protection for parameters

Processing <--> Structure

- JW* objects are flat lists of fields, with requirements not clearly specified
- Processing requires looking for collections of fields being present
  - Should I do key unwrapping or key agreement?
- JSMS organizes data elements according to processing
  - Algorithm parameters, public keys, wrapped keys
  - Whenever there's a branch point, isolate an object

Header Integrity

- JW* applies integrity protection to all header parameters
  - Unnecessary complexity, esp. for multiple recipients
- Survey of prior art:
  - No protection: TLS, IKE
  - Algorithm protection: CMS, PKIX
  - Full parameter protection: XML-Dsig
- No clear benefit to protecting parameters other than algorithms

Proposals

- Enable primary format to be JSON
  - "base64 agility"
- Add internal structure and clarify requirements / processing of JOSE objects
  - Algorithm+parameters, public keys, wrapped keys
  - Separate MAC from signing, allowing wrapped keys
- Change from whole-header protection to protection of designated attributes