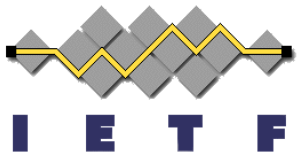
*

Key Issues and Choices for COSE

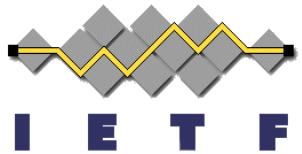


Based on a review of draft-ietf-cose-msg-01

Mike Jones

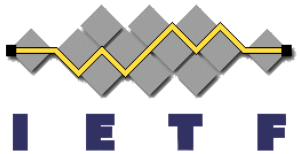
IETF 93
Prague
July 2015

Key Issues and Choices for COSE



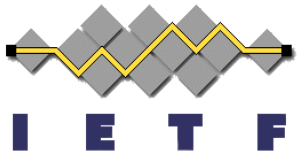
- Our goals should include:
  - Keeping simple things simple
  - Making complex things possible, when necessary
  - Compactness of representations
  - Compactness of implementations
  - *Leading to adoption*
- Presentation identifies potential areas for simplification

Example: Direct MAC Current Representation




```
{ 1 (typ): 3 (MAC),
  2 (protected): h'a1016f4145532d434d41432d3235362f3634',
          ({1 (alg): “AES-CMAC-256/64”})
  4 (payload): h'546869732069732074686520636f6e74656e742e',
          (“This is the content.”)
  10 (tag): h'd9afa663dd740848',
  9 (recipients): [
    { 3 (unprotected): {
        1 (alg): -6 (direct),
        5 (kid): h'6f75722d736563726574‘ (“our-secret”)
      } }
  ]
}
```

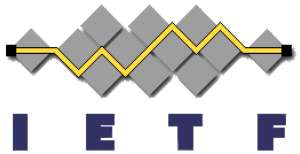
Example: Direct MAC Possible Simplifications




```
{ 1 (typ): 3 (MAC),
  2 (protected): h'encoding TBD',
          ({1 (alg): "AES-CMAC-256/64"
            5 (kid): h'6f75722d736563726574' ("our-secret")
           })
  4 (payload): h'546869732069732074686520636f6e74656e742e',
          ("This is the content.")
  10 (tag): h'd9afa663dd740848'
}
```

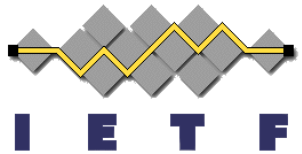
- Simplifications applied
  - Flattened serialization (no "recipient")
  - Removed key management layer -6 (direct)

Choice: Representation of Single-Recipient Content



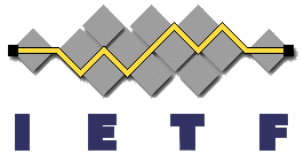
- Current draft always uses recipients array
  - Always a singleton for single recipient
- Even for direct content, currently always two sets of header parameters
  - Those describing the cryptographic operations
  - Those describing the recipient
- In single recipient case we could:
  - Eliminate the “recipients” tag and the array
  - Have only one set of header parameters

Choice: Representation of Key Management



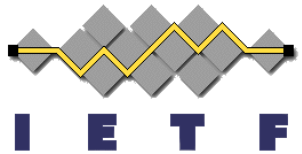
- Current draft always includes key management structure, even when “direct”
- An alternative is to include a key management structure only when needed
  - Omit it in the “direct” case and combine headers
  - This still allows having one “alg” parameter, versus JOSE which required two (“alg”, “enc”)
  - Note: This approach allows multiple levels of key management by nesting, like Jim’s Appendix B

Choice: Use Maps or Arrays at Top Level



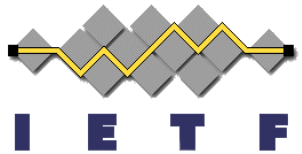
- Current draft uses maps
- Alternative is to define array representations of signed, MACed content, encrypted
  - Analogous to JOSE compact serializations
  - May make representing key management messier
    - Would key management maps also become arrays?
    - Or would headers for levels be combined, requiring different "alg" parameters like JOSE's "alg" and "enc"?
  - How to identify the different types?
    - CBOR type prefix or first array element?
- I'm personally OK staying with maps
  - Seems like there's fewer special cases that way

Choice: Overloaded or Single Use Label Values



- Current draft overloads map labels with different meanings onto same value
  - E.g., 4 for both payload and ciphertext
- No obvious disadvantage to using different labels when meanings different
  - Some advantages, such as more comprehensibility of encoding
  - Also may avoid conflicts that aren't apparent now but may occur when extensions defined
- I'd personally recommend single use labels

Choice: Concatenate Tag to Ciphertext or Keep Separate

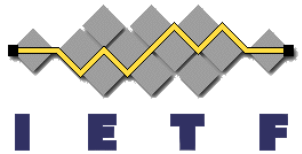


- Do we represent authenticated encryption output with one or two parameters?
  - “ciphertext”: ciphertext, “tag”: authentication tag *or*
  - “ciphertext”: ciphertext || authentication tag
- AES GCM [SP 800-38D] specified as providing two output parameters
- JOSE kept the separate parameters separate
- TLS and some other specs concatenate them
- Already a “tag” parameter used by MACs

Issue: Confusing Header Parameter Descriptions

- Some names copied from JOSE should be changed:
  - “jku” to “cku” (COSE Key URL)
  - “jwk” to “ck” (COSE Key)

Choice: Which Header Parameters to Standardize

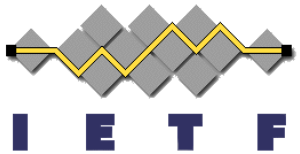


- Issue 1 in the draft: “Which of the following items do we want to have standardized in this document: jku, jwk, x5c, x5t, x5t#S256, x5u, zip”
- I’d advocate cku, ck, x5c, x5t, x5t#S256, x5u, zip
- Related choice:
  - Do we also want to have “jku” (JWK URL) to point to keys in JWK format in addition to “cku”?

Choice: Include JOSE Alg Names in COSE Alg Registry

- Advantages of doing so:
  - Ability to reuse JOSE alg registrations by just defining short labels for them
  - Clearer documentation when same algs can be used in both JOSE and COSE
  - Encourages registration of algs defined for use by COSE to also be registered for use with JOSE
    - For example, AES-CMAC
  - Reduces duplication
- Don’t see much downside in doing so

Issue: Why the asymmetry between sig & mac structs?

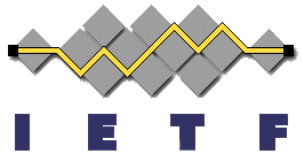



```
Sig_structure = [
      body_protected: bstr,
      sign_protected: bstr,
      payload: bstr
  ]
```

- versus

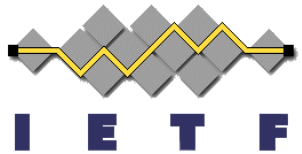
```
MAC_structure = [
       protected: bstr,
       external_aad: bstr,
       payload: bstr
  ]
```

Choice: Define “use” Key Member



- JOSE “use” has two values: “sig”, “enc”
  - Based on XML DSIG/ENC key use definition
  - Useful for public keys
  - Single valued
- JOSE “key_ops” value an array
  - Based on WebCrypto API
    - WebCrypto API does define how “use” works as well
  - Useful for public and private keys
- Semantic compatibility with other systems argues for keeping it

Request: Add Symbolic Annotations to Examples



{

1 (typ): 3 (MAC),

2 (protected): h'a1016f4145532d434d41432d3235362f3634',

({1 (alg): “AES-CMAC-256/64”})

…

- versus

{

1: 3,

2: h'a1016f4145532d434d41432d3235362f3634',

…