SUIT Manifest

draft-ietf-suit-manifest-12

Brendan Moran, Hannes Tschofenig, Henk Birkholz, Koen Zandberg

March 11th, 2021, notinprague

Changes from v11

- Examples contained two errors:
  - Manifest digest was present in COSE objects
  - One digest

Delete component (Request from TEEP)

- Delete can be problematic as an imperative:
  - If permissions are wrong, could break dependencies
  - Who actually has authority to delete a TC?
  - Might not really mean “delete”:
    - what if two TAs depend on the same component and one deletes it?
  - Might already have been deleted
  - May break atomic nature of updates
    - Especially if used & deleted in same manifest
- Maybe Unlink or Garbage-Collect would be a better idiom

Garbage-Collect Component

- Marks a component as unused by the current manifest tree
- Manifest Processor applies marks the component
  - E.g. decrement a reference count
- Once the current section is complete, manifest processor checks for marked components that can be deleted.

Encryption in SUIT

Firmware Encryption

AES 128 Key Wrap (KW)

- AES KW described in RFC 3394
- Symmetric *Key Encryption Key (KEK)* is used to encrypt a randomly generated *Content Encryption Key (CEK)*.

ECDH Ephemeral-Static + AES KW

- Sender creates an ephemeral ECDH key pair (E-Pub/E-Priv).
- Sender uses the receiver's static public key (S-Pub) with the private key (E-Priv) to derive a symmetric key (ECDH-Shared)
- Sender applies HKDF on ECDH-Shared to produce KEK
- Sender generates a random CEK
- Sender encrypts the CEK with KEK.

AES 128 KW in COSE

```
96(                                                     // COSE ENCRYPT
    [
        h'A10101',                                      // protected field with alg=AES-GCM-128
        {                                               // unprotected field with...
            5: h'26682306D4FB28CA01B43B80'              //      iv
        },
        null,                                           // detached ciphertext
        [                                               // recipients array
            h'',                                        // empty protected field
            {                                           // unprotected field with...
                1: -3,                                  //     alg=AES-128-KW
                4: h'6B69642D31'                        //     kid
            },
            h'2AD7307BCB5EBDDD…4669D4DF13F46945'        // CEK encrypted with KEK
        ]
    ]
)
```

Notes on AES 128 KW

- Additional Data Structure needs clarifications:

```
Enc_structure = [
  context : "Encrypt",
  protected : empty_or_serialized_map,
  external_aad : bstr
]
```

- Protected refers to outer protected field – not inner.
- Suggestion: external_aad = null

ECDH Ephemeral-Static + AES KW in COSE

```
96(
    [
        h'A10101',
        {
            5: h'26682306D4FB28CA01B43B80'
        },
        null,
        [
            h'',
            {
                1: -3
            },
            h'FA55A50CF110908DA6443149F2C2062011A7D8333A72721A',
            [
                h'A1013818',
                {
                    -1: h'A4010220012158205F...979D5168718766510C445',
                     4: h'6B69642D31'
                },
                null
            ]
        ]
    ]
)
```

```
// COSE ENCRYPT

// protected field with alg=AES-GCM-128
// unprotected field with...
//     iv

// detached ciphertext
// recipients array
// empty protected field
// unprotected field with ...
//     alg=AES-128-KW

// CEK encrypted with KEK
// recipients array
// protected field with alg=ECDH-ES + HKDF-256
// unprotected field with ...
//    ephemeral structure
//    kid

// empty ciphertext
```

Ephemeral

- The ephemeral structure contains the public ECDHE key + meta-data:

```
{
    1: 2,                                                  // key type (kty) parameter –> EC2
   -1: 1,                                                  // curve identifier (crv) parameter –> P-256
   -2: h'5FA28AA979D51E570E621C69F3C57C76608B21EECF2696629E65A0B4772A1174',  // x
   -3: h'60F29EA947048EFECA06F6DBEDF185CA559B181DE9EB6D80E68718766510C445'   // y
}
```

The “Context”

```
PartyInfo = (
     identity : bstr / nil,
     nonce : bstr / int / nil,
     other : bstr / nil
 )

 COSE_KDF_Context = [
     AlgorithmID : int / tstr,
     PartyUInfo : [ PartyInfo ],
     PartyVInfo : [ PartyInfo ],
     SuppPubInfo : [
         keyDataLength : uint,
         protected : empty_or_serialized_map,
         ? other : bstr
     ],
     ? SuppPrivInfo : bstr
 ]
```

KEK = HKDF(ECDH-Shared, context)

encryptedCEK = KeyWrap(KEK, CEK)

- `PartyUInfo.Identity` => ?
- `PartyVInfo.Identity` => kid
- `Nonce` => Always nil
- `SuppPubInfo`
  - `Protected`, `AlgorithmID` and `keyDataLength` => algorithm used to encrypt the CEK (?)
- No `other (?)`
- `SuppPrivInfo => null`

General Recommendations

- Only use Encrypt structure
- Specify a small set of mechanisms in detail for interoperability and to limit code size. More key exchange techniques can be added later.
- Use only detached mode for ciphertext.

- Q: Does the same description also apply to encryption of the manifest?

Next Steps

- Create a PR to add examples and text.
- Need someone to verify the content.
- Describe example(s) for multiple recipients.
- Mcuboot uses (some) Elliptic Curve Integrated Encryption Scheme (ECIES)
  - Planning to specify hybrid public key encryption based on draft-irtf-cfrg-hpke.
  - Looks less complicated than the currently specified COSE public key encryption techniques.
  - Probably a better story long-term.