Hybrid Public Key Encryption (HPKE) for COSE

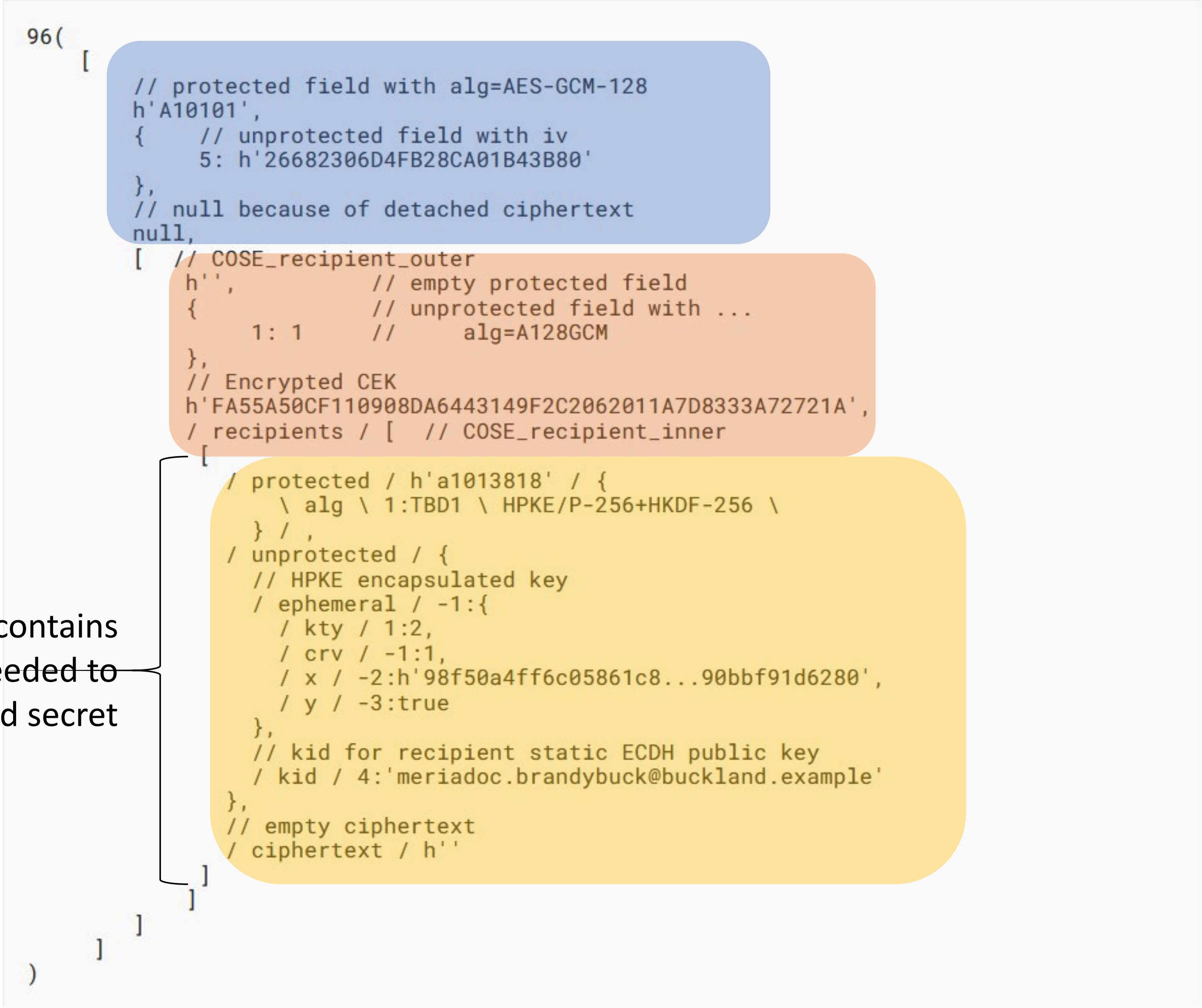
draft-tschofenig-cose-hpke-00

(Russ, Brendan, Hannes)

Background

- The SUIT WG worked on firmware encryption scheme, (which is also used in TEEP).
- Functionality was recently moved into a dedicated document, see https://datatracker.ietf.org/doc/html/draft-ietf-suit-firmware-encryption-02
- We wanted two features:
    - A pre-shared secret-based key encryption □ AES Key Wrap (offered by COSE)
    - Public key encryption scheme □ Also offered by COSE (in form of the ECDH Ephemeral-Static key agreement)
- Everything great but HPKE (Hybrid Public Key Encryption) emerged in the IETF/IRTF as the prominent public key encryption scheme.
    - See https://datatracker.ietf.org/doc/html/draft-irtf-cfrg-hpke-12
    - Already used in several specifications, such as TLS ESNI and MLS.
    - Code for HPKE available!
- Group decided to re-use HPKE.

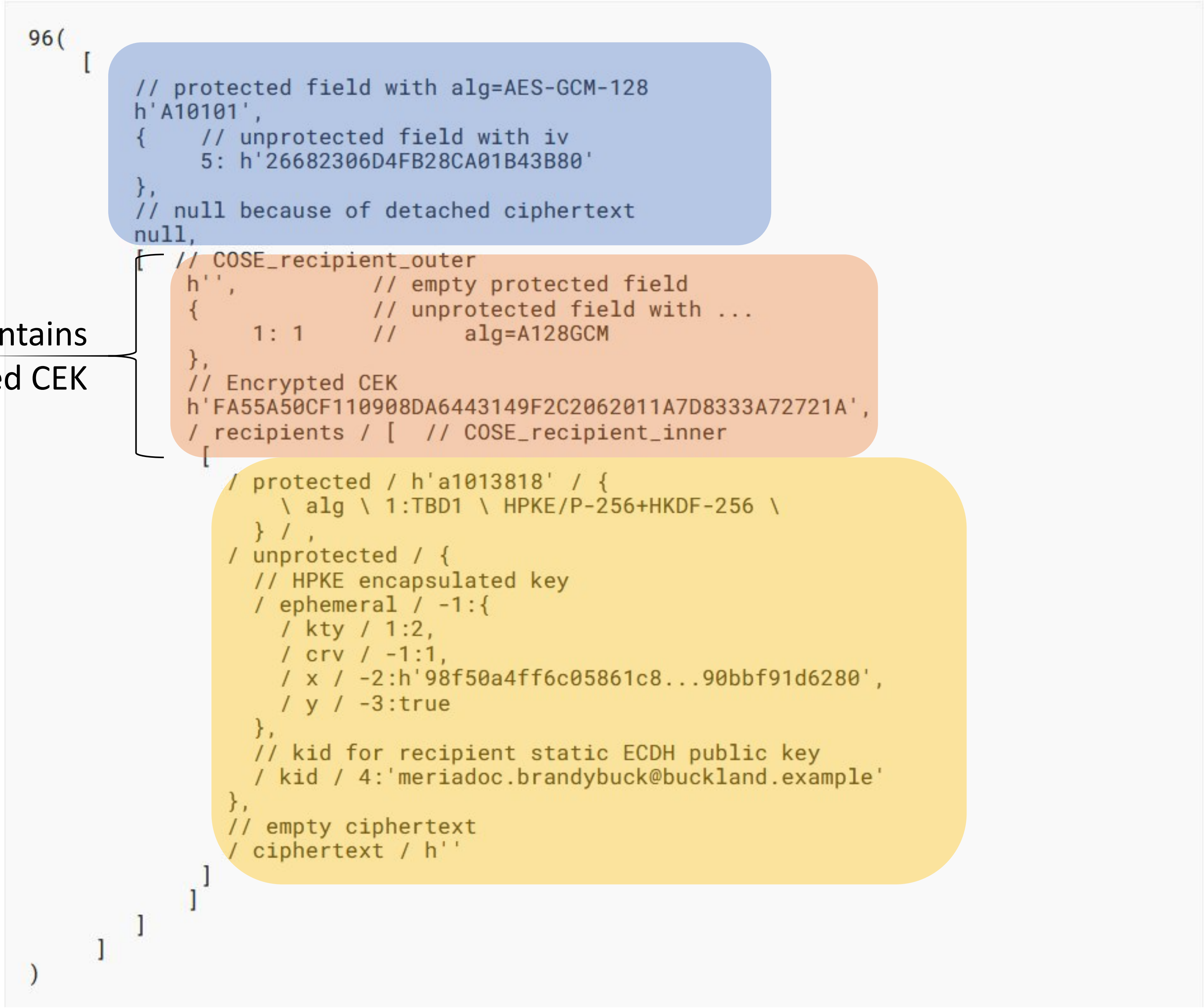
COSE-HPKE



```
96(
    [
        // protected field with alg=AES-GCM-128
        h'A10101',
        {    // unprotected field with iv
             5: h'26682306D4FB28CA01B43B80'
        },
        // null because of detached ciphertext
        null,
        [  // COSE_recipient_outer
            h'',           // empty protected field
            {              // unprotected field with ...
                 1: 1      //      alg=A128GCM
            },
            // Encrypted CEK
            h'FA55A50CF110908DA6443149F2C2062011A7D8333A72721A',
            / recipients / [  // COSE_recipient_inner
             [
               / protected / h'a1013818' / {
                   \ alg \ 1:TBD1 \ HPKE/P-256+HKDF-256 \
                 } / ,
               / unprotected / {
                 // HPKE encapsulated key
                 / ephemeral / -1:{
                   / kty / 1:2,
                   / crv / -1:1,
                   / x / -2:h'98f50a4ff6c05861c8...90bbf91d6280',
                   / y / -3:true
                 },
                 // kid for recipient static ECDH public key
                 / kid / 4:'meriadoc.brandybuck@buckland.example'
               },
               // empty ciphertext
               / ciphertext / h''
             ]
            ]
        ]
     ]
)
```

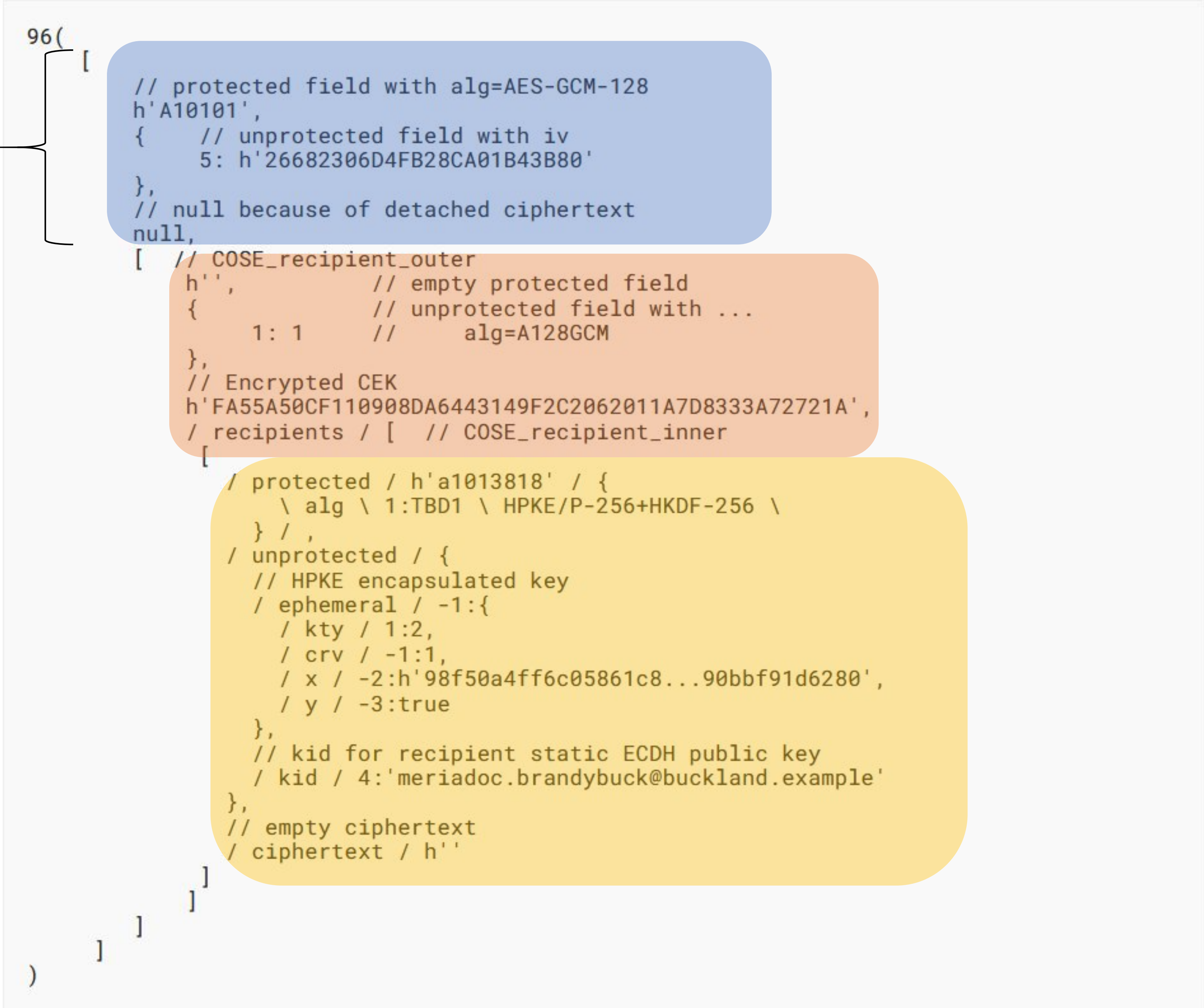
Layer 3 contains parameters needed to generate a shared secret

Layer 2 contains the encrypted CEK



```
96(
    [
        // protected field with alg=AES-GCM-128
        h'A10101',
        {    // unprotected field with iv
             5: h'26682306D4FB28CA01B43B80'
        },
        // null because of detached ciphertext
        null,
        [  // COSE_recipient_outer
            h'',              // empty protected field
            {                 // unprotected field with ...
                 1: 1         //      alg=A128GCM
            },
            // Encrypted CEK
            h'FA55A50CF110908DA6443149F2C2062011A7D8333A72721A',
            / recipients / [  // COSE_recipient_inner
             [
               / protected / h'a1013818' / {
                   \ alg \ 1:TBD1 \ HPKE/P-256+HKDF-256 \
                 } / ,
               / unprotected / {
                 // HPKE encapsulated key
                 / ephemeral / -1:{
                   / kty / 1:2,
                   / crv / -1:1,
                   / x / -2:h'98f50a4ff6c05861c8...90bbf91d6280',
                   / y / -3:true
                 },
                 // kid for recipient static ECDH public key
                 / kid / 4:'meriadoc.brandybuck@buckland.example'
               },
               // empty ciphertext
               / ciphertext / h''
             ]
            ]
        ]
     ]
)
```

Layer 1 contains
the encrypted plaintext
(unless it is detached)



```
96(
    [
        // protected field with alg=AES-GCM-128
        h'A10101',
        {    // unprotected field with iv
             5: h'26682306D4FB28CA01B43B80'
        },
        // null because of detached ciphertext
        null,
        [  // COSE_recipient_outer
            h'',          // empty protected field
            {             // unprotected field with ...
                 1: 1     //      alg=A128GCM
            },
            // Encrypted CEK
            h'FA55A50CF110908DA6443149F2C2062011A7D8333A72721A',
            / recipients / [  // COSE_recipient_inner
             [
               / protected / h'a1013818' / {
                   \ alg \ 1:TBD1 \ HPKE/P-256+HKDF-256 \
                 } / ,
               / unprotected / {
                 // HPKE encapsulated key
                 / ephemeral / -1:{
                   / kty / 1:2,
                   / crv / -1:1,
                   / x / -2:h'98f50a4ff6c05861c8...90bbf91d6280',
                   / y / -3:true
                 },
                 // kid for recipient static ECDH public key
                 / kid / 4:'meriadoc.brandybuck@buckland.example'
               },
               // empty ciphertext
               / ciphertext / h''
             ]
            ]
        ]
     ]
)
```

Ask to the group

We would like the COSE WG to adopt this document.

We believe it is of generic use beyond firmware encryption

Background Material

HPKE Implementation

- https://github.com/ARMmbed/mbedtls/pull/5078
- Based on Stephen Farrells “HappyKey” code, see https://github.com/sftcd/happykey.
- HappyKey relies on OpenSSL. Above linked implementation uses the PSA Crypto API and is tailored to constrained devices.
- Code with integration into COSE will be released soon.