Changes to mls-extensions

Rohan Mahy
rohan.ietf@gmail.com

Richard Barnes
rlb@ipv.sx

We were abusing “extensions”

People have been using “extensions” to mean a lot of things.

We should refocus:

> “MLS extension” == “something that changes the behavior of MLS”

Code that would just store data in an “extensions” field is not an extension, it’s application logic, and we should treat it as such.

What we did

- Create Safe Application API for things that don't modify the behavior of MLS

The resulting description is much cleaner, simpler, and more concise.

Application “Components”

Accommodate the idea that an application may have multiple parts

Call these parts “components”

Assign them “component IDs” values that are unique within an application

Use the component IDs for domain separation, data separation

This is just the “extension IDs” from safe extensions, renamed, but it applies to other things like AAD

Safe Applications API - Exposing MLS functions to apps

Sending authenticated+confidential data => PrivateMessage.application_data ✅	
Sending authenticated data => PrivateMessage.authenticated_data ✅	
Confirming group agreement on arbitrary data	AppEphemeral
Confirming agreement on changes made in a Commit	app_data_dictionary AppDataUpdate
Re-using HPKE key pairs in leaf nodes Re-using signature key pairs in leaf nodes Injecting PSKs Exporting secrets	safe Labels using component id
Safe AAD	aad_items include component id
Contents of application_data	determined using content advertisement

Attaching Application Data to MLS Messages

```
struct {
    ComponentID component_id;
    opaque data<V>;
} ComponentData;

struct {
    ComponentData component_data<V>;
} AppDataDictionary;
```

MLS messages have extension fields: GroupContext, GroupInfo, LeafNode, KeyPackage

Just make one extension to shove in all these places.

All of these are ephemeral, **except for GroupContext**

Updating GroupContext.app_data_dictionary

```
enum {
    invalid(0),
    update(1),
    remove(2),
    (255)
} AppDataUpdateOperation;

struct {
    ComponentID component_id;
    AppDataUpdateOperation op;

    select (AppDataUpdate.op) {
        case update: opaque update<V>;
        case remove: struct{};
    };
} AppDataUpdate;
```

Send a proposal with updates to the application data field

Format of the updates is entirely up to the app; could be replace, could be diff, etc.

Presumes an API where the MLS stack asks the application mid-commit-processing how to apply the updates.

UpdatePath is optional

Can be an external proposal

Can be included in external commits

Application-defined proposals

AppEphemeral proposal type.

Opaque to MLS, but gets included in the transcript.

=> Everyone gets the same app data with this Commit

Doesn’t affect MLS at all

=> UpdatePath never required

Does not make sense in External Proposal

Can be in an External Commit

```
struct {
    ComponentID component_id;
    opaque data<V>;
} AppEphemeral;
```

Safe AAD follows the same pattern as app_data_dictionary

aad_items is a vector of Safe AAD prefixed with each items's component_id.

If you have non-safe AAD it goes after the safe AAD.

```
struct {
  ComponentID component_id;
  opaque aad_item_data<V>;
} SafeAADItem;

struct {
  SafeAADItem aad_items<V>;
} SafeAAD;
```

Improved negotiation for both extensions and components

Added `supported_wire_formats` and `required_wire_formats` **extensions**

Added `app_components` **component** to negotiate supported required components

Added `safe_aad` **component** to negotiate which components can/need to appear in the AAD

Added `content_media_types` **component** to negotiation supported/required media types

Remapping existing "extensions"

- AppAck is now a usage of AppEphemeral
- TargetedMessage WireFormat is an extension
- SelfRemove is an extension
- Content Advertisement uses a component
- Last resort KeyPackages implemented as a component.

Recap

“Extension” = “influences how MLS works”

“Doesn’t influence how MLS works” = “Application functionality”

More limited definition of “extensions”, more expansive definition of “app”

Ready for WGLC?

ML-KEM (plain and hybrid) cipher suites for MLS

draft-mahy-mls-pq

Rohan Mahy and Richard Barnes

Wanted cipher suites using PQ and PQ/T ML-KEM

ML-KEM-768 + X25519	128 bits	non-NIST	hybrid
ML-KEM-768 + P-256	128 bits	NIST	hybrid
ML-KEM-1024 + P-384	192 bits	NIST	hybrid
ML-KEM-768	128 bits	NIST	pure PQ
ML-KEM-1024	192 bits	NIST	pure PQ

What components did we use?

KEM	KDF	AEAD	Hash	Signature
XWing (ML-KEM-768 + X25519)	XOF(SHAKE256)	AES256	SHA384	Ed25519
QSF(ML-KEM-768 + P-256)	XOF(SHAKE256)	AES256	SHA384	P256
QSF(ML-KEM-1024 + P-384)	XOF(SHAKE256)	AES256	SHA384	P384
ML-KEM-768	XOF(SHAKE256)	AES256	SHA384	P256
ML-KEM-1024	XOF(SHAKE256)	AES256	SHA384	P384

Open Issues

- Do we want to use SHA3 instead of SHA2 in any of these cipher suites?
  - SHA3 is already in ML-KEM
    - If implemented in pure software, then it is easy to reuse
    - If ML-KEM is implemented in a module then it might require adding another SHA3 implementation for the hash
  - SHA3 is slower
  - SHA3 is stronger

- What KDF to use? Was XOF(SHAKE256) the correct choice?

- There was no draft to register XOF(SHAKE256) as an HPKE KDF, so we added the registration.
- At the same time Deirdre wrote a draft to do it.

- Why use XWing instead of KitchenSink? XWing is faster, deployed for TLS, and is safe in MLS

- **Most of these will be sorted out by the CFRG KEM combiners work**
- **MLS should go ahead and adopt this draft to get started**

Web token credentials for MLS

draft-mahy-mls-sd-cwt-creds

Rohan Mahy

Intro to Selective Disclosure (SD) Web Tokens

- CBOR Web Tokens (CWTs - ex: webauthn) and JSON Web Tokens (JWTs)
- Solution statement
    - Allow a **Holder** to get a CBOR or JSON Web Token from an **Issuer** and then safely blind ("redact") or reveal ("disclose") some of its claims to a **Verifier**
- Holder can disclose *all, some, or none* of the disclosable claims.
- Issuer dictates which Claims can be blinded or disclosed
- Can include **decoys**

- SD-JWT format includes a regular JWT, 0 or more disclosures, and *optional* key binding.
- SD-CWT is wrapped in a *mandatory* SD-KBT (Key Binding Token)
- How Issuer decides which claims to selectively disclose is out-of-scope. Issuer does not have to make claims disclosable.
- Includes Holder's public key or reference to it (`cnf` claim)

What can be disclosed?

No: Any CWT claims that are part of the validation of the CWT (iss, aud, exp, nbf, iat, cti, cnf, nonce), **except** the subject.

Yes: The subject; Any other claims at *any level of hierarchy*; Any array items of any CWT claims (ex: `swversion` below).

```
{
   / sub / 1: "https://device.holder.example",
   / cnf / 8 : {
      / ckt / 5 : h'496bd8a…3bda88fadd1669da253ec'
   },
   …
   / inspection passed /  500 : true,
   / inspector license /  501 : “ABCD-123456”,
   / swversion / 271 : [
      "3.5.5",
      "4.1.7"
   ],
   / inspection location /  503 : {
      "country"     : "us",      / United States /
      "region"          : "ca",              / California /
      "locality"    : "San Francisco",
      "postal_code" : "94188"
}
```

Forming the disclosures

Disclosures are formed using a per-disclosure salt, the value, and claim name for map/object entries

```
<<[
  /salt/ h'8d5c15fa86265d8ff77a0e92720ca837',
  /claim/  500,  / inspection_passed /
  /value/  true
]>>
```

Hashing the Disclosures

The **hash** of each disclosure becomes the **redacted form** of that claim, array element (or decoy). These hashes are placed in the *payload*, which is signed by the issuer. (Example CBOR syntax)

```
/ cose-sign1 / 18([
  / protected   / <<{
    / key / 4 : h'496b…53ec', # public key thumbprint
    / algorithm / 1 : -7,     # ES256
  }>>,
  / unprotected / {},
  / payload     /<<{
    / iss / 1 : "https://issuer.example",
    / sub / 2 : "https://subject.example",
    / iat / 6: 1729706150,
    / redacted_map_keys / simple(59) : [
      / hash of disclosure for inspection_passed / h'db6e…76e2',
      / hash of disclosure for license_number  / h'c234…3b04'
    ],
    /inspection_dates / 502: [ 1549560720, 60(h’7ad6…092f’)
  }>>,
  / signature   / h'2544f2ed...5840893b'
])
```

Disclosing to a Verifier

Holder presents to a Verifier:

- the Issuer CWT
- any disclosures
- the Holder Key Binding
    - Key Binding Token (KBT) required for SD-CWT,
    - SD-JWT+KB (Key Binding) optional for SD-JWT

Can include an *audience* specific to the Verifier, and/or *nonce* scoped to the Verifier

Key Binding

Key Binding prevents copy and paste or replay attacks on selective disclosure.
It includes the Issued CWT and disclosures in the KBT protected header
The Holder proves possession of their private key and choice of disclosures.

```
/ sd_kbt    / 18([
  / protected / << {
            / alg / 1 : -7 / ES256 /,
            / typ / 16 : "application/kb+cwt"
            / kcwt / 13: 18([
                / protected / h’0fd2…6aa3’,
                / unprotected / {
                      /sd_claims/ 17: /disclosures go here /
                },
                / payload / h’ad49…1473’,
                /signature / h’28a9…83fd’
      ])
  } >>,
  / unprotected / {},
  / payload / << {
            / cnonce / 39          : h'e0a156...bb3f',
            / aud      / 3         : "https://verifier.example",
            / iat      / 6         : 1783000000,
  } >>,
  / signature / h'1237af2e...678945'
])
```

Nested disclosures

Disclosures can refer to nested data

```
{
  / name / 170: "Alice Smyth",
  / address / 187: {
    / region   / 2: "jp",
    / locality / 3: "Tokyo",
    / physical / 8: {
      / block / 1: "2 Chome-5-2",
      / bldg /  2: "Mitsubishi Building"
      / floor / 3: "B1F"
    }
  },
  / nationalities / 197: [
    "jp", "ie"
  ]
}
```

Nested disclosures

Disclosures can refer to nested data

```
{
  / name / 170: "Alice Smyth",
  / address / 187: {
    / region   / 2: "jp",
    / locality / 3: "Tokyo",
    / physical / 8: { simple(59) : [
      h'f41a...8246',
      h'1562...228d',
      h'277c...839f' ]
    }
  },
  / nationalities / 197: [
    "jp", "ie"
  ]
}
```

Nested disclosures

Disclosures can refer to nested data

```
{
  / name / 170: "Alice Smyth",
  / address / 187: { simple(59) : [
    h'ce5a...45fe',
    h'4d5a...cf14',
    h'08fc...a01b'




    ]
  },
  / nationalities / 197: [
    60(h'7ef9...2388'), 60(h'2782...65af')
  ]
}
```

Nested disclosures

Disclosures can refer to nested data

```
{ simple(59) : [
  h'6194...d892',
  h'56b8...0389',









  h'3ebc...c52e' ]



}
```

SD-CWTs and SD-JWTs as MLS Credentials

- This is relatively straightforward.
- Currently uses the SD-KBT for SD-CWT. Might be able be to use the SD-CWT directly instead in MLS.
- Works great with pseudonyms.

```
struct {
    CredentialType credential_type;
    select (Credential.credential_type) {
        case basic:
            opaque identity<V>;
        case x509:
            Certificate certificates<V>;
        ...
        case sd_cwt:
            opaque sd_kbt<V>;
        case sd_jwt:
            SdJwt sd_jwt;
    };
} Credential;
```

SD-JWT as Credential

- the canonical format of an SD-JWT looks like this:
- Can convey it "as is", or decomposed and unbase64ed to make it smaller

eyJhbGciOiAiRVMyNTYiLCAidHlwIjogImV4YW1wbGUrc2Qtand0In0.eyJfc2QiOiBb
IkNyUWU3UzVrcUJBSHQtbk1ZWGdjNmJkdDJTSDVhVFkxc1VfTS1QZ2tqUEkiLCAiSnpZ
akg0c3ZsaUgwUjNQeUVNZmVadTZKdDY5dTVxZWhabzdGN0VQWWxTRSIsICJQb3JGYnBL
dVZ1Nnh5bUphZ3ZrRnNGWEFiUm9jMkpHbEFVQTJCQTRvN2NJIiwgIlRHZjRvTGJnd2Q1
SlFhSHlLVlFaVTlVZEdFMHc1cnREc3JaemZVYW9tTG8iLCAiWFFfM2tQS3QxWHlYN0tB
TmtxVlI2eVoyVmE1TnJQSXZQWWJ5TXZSS0JNTSIsICJYekZyendzY002R242Q0pEYzZ2
Vks4QmtNbmZHOHZPU0tmcFBJWmRBZmRFIiwgImdiT3NJNEVkcTJ4Mkt3LXc1d1BFemFr
b2I5aFYxY1JEMEFUTjNvUUw5Sk0iLCAianN1OXlWdWx3UVFsaEZsTV8zSmx6TWFTRnpn
bGhRRzBEcGZheVF3TFVLNCJdLCAiaXNzIjogImh0dHBzOi8vaXNzdWVyLmV4YW1wbGUu
Y29tIiwgImlhdCI6IDE2ODMwMDAwMDAsICJleHAiOiAxODgzMDAwMDAwLCAic3ViIjog
InVzZXJfNDIiLCAibmF0aW9uYWxpdGllcyI6IFt7Ii4uLiI6ICJwRm5kamtaX1ZDem15
VGE2VWpsWm8zZGgta284YUlLUWM5RGxHemhhVllvIn0sIHsiLi4uIjogIjdDZjZKa1B1
ZHJ5M2xjYndIZ2VaOGtoQXYxVTFPU2xlclAwVmtCSnJXWjAifV0sICJfc2RfYWxnIjog
InNoYS0yNTYiLCAiY25mIjogeyJqd2siOiB7Imt0eSI6ICJFQyIsICJjcnYiOiAiUC0y
NTYiLCAieCI6ICJUQ0FFUjE5WnZ1M09IRjRqNFc0dmZTVm9ISVAxSUxpbERsczd2Q2VH
ZW1jIiwgInkiOiAiWnhqaVdXYlpNUUdIVldLVlE0aGJTSWlyc1ZmdWVjQ0U2dDRqVDlG
MkhaUSJ9fX0.aOEWBAzhRHVHxFYicngQrI9rgX-coljX0PCcVinRe0xyInH3RpvGbwxP
xHiISXXBarHB3StKbW4G40LMHDSw4g~WyIyR0xDNDJzS1F2ZUNmR2ZyeU5STjl3IiwgI
mdpdmVuX25hbWUiLCAiSm9obiJd~WyJlbHVWNU9nM2dTTklJOEVZbnN4QV9BIiwgImZh
bWlseV9uYW1lIiwgIkRvZSJd~WyI2SWo3dE0tYTVpVlBHYm9TNXRtdlZBIiwgImVtYWl
sIiwgImpvaG5kb2VAZXhhbXBsZS5jb20iXQ~WyJlSThaV205UW5LUHBOUGVOZW5IZGhR
IiwgInBob25lX251bWJlciIsICIrMS0yMDItNTU1LTAxMDEiXQ~WyJRZ19PNjR6cUF4Z
TQxMmExMDhpcm9BIiwgInBob25lX251bWJlcl92ZXJpZmllZCIsIHRydWVd~WyJBSngt
MDk1VlBycFR0TjRRTU9xUk9BIiwgImFkZHJlc3MiLCB7InN0cmVldF9hZGRyZXNzIjog
IjEyMyBNYWluIFN0IiwgImxvY2FsaXR5IjogIkFueXRvd24iLCAicmVnaW9uIjogIkFu
eXN0YXRlIiwgImNvdW50cnkiOiAiVVMifV0~WyJQYzMzSk0yTGNoY1VfbEhnZ3ZfdWZR
IiwgImJpcnRoZGF0ZSIsICIxOTQwLTAxLTAxIl0~WyJHMDJOU3JRZmpGWFE3SW8wOXN5
YWpBIiwgInVwZGF0ZWRfYXQiLCAxNTcwMDAwMDAwXQ~WyJsa2x4RjVqTVlsR1RQVW92T
U5JdkNBIiwgIlVTIl0~WyJuUHVvUW5rUkZxM0JJZUFtN0FuWEZBIiwgIkRFIl0~


```
struct {
    Bool compacted;
    select (compacted) {
        case true:
            opaque protected<V>;
            opaque payload<V>;
            opaque signature<V>;
            SdJwtDisclosure disclosures<V>;
            opaque sd_jwt_key_binding<V>;
        case false:
            opaque sd_jwt_kb<V>;
    };
} SdJwt;
```

Encrypted disclosures in SD-CWT

- Allows a disclosure to be ephemerally AEAD encrypted
- In the MLS context, allows a disclosure to be hidden from DS but shared with all (or a subset) of members.

```
<<[
    /salt/   h'bae611067bb823486797da1ebbb52f83',
    /claim/  501,   / inspector_license_number /
    /value/  "ABCD-123456"
]>>
```

```
/ sd_encrypted_claims / 19 : [ / encrypted disclosures /
    <<[
        / nonce /      h'95d0040fe650e5baf51c907c31be15dc',
        / ciphertext / h'208cda279ca86444681503830469b705
                         89654084156c9e65ca02f9ac40cd62b5
                         a2470d',
        / mac /        h'1c6e732977453ab2cacbfd578bd238c0'
    ]>>,
    ...
]
```

Questions?