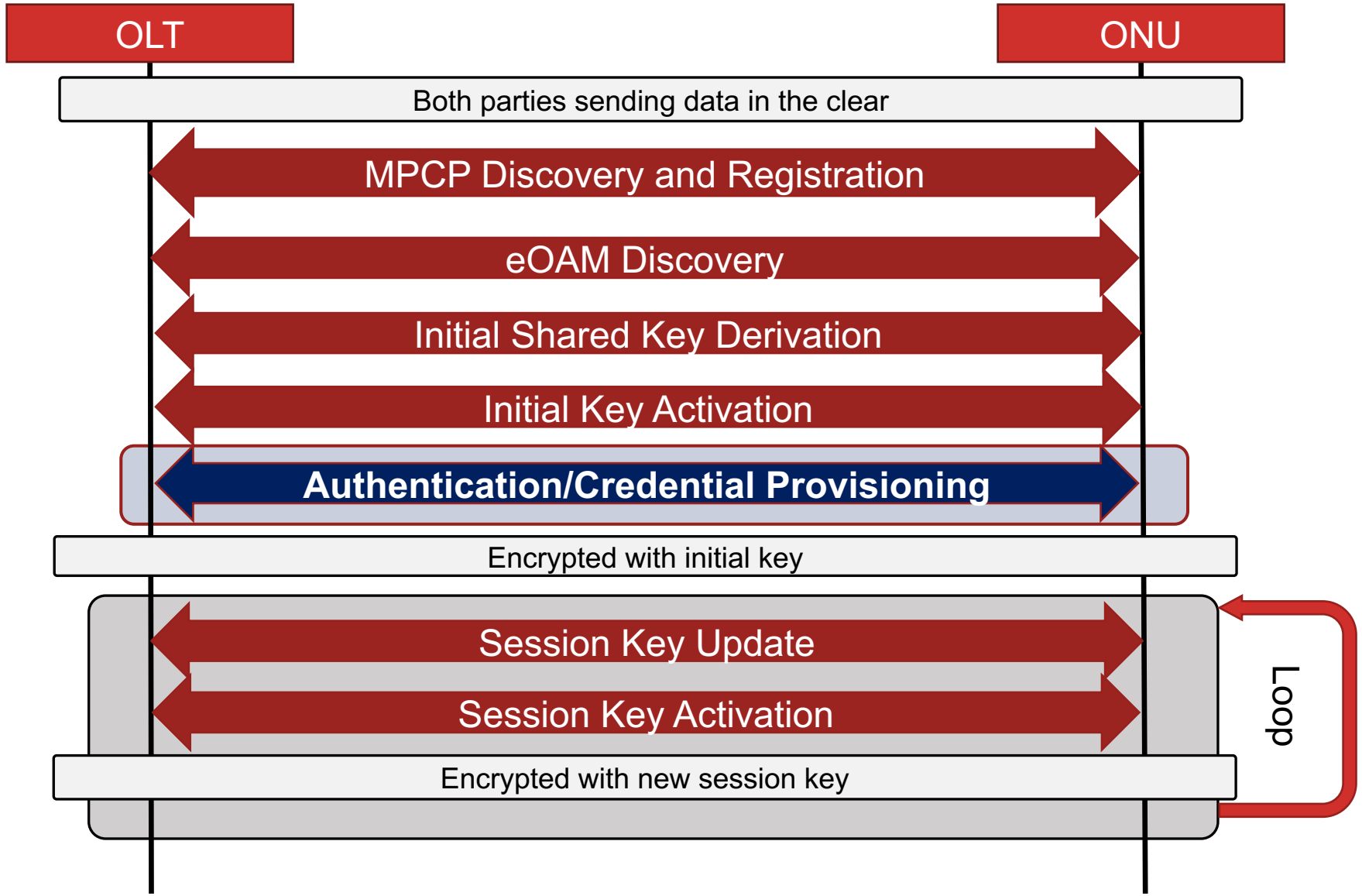
SIEPON.4 Authentication Proposal
v0.5 – 2023-11-30

Craig Pratt | Lead Software Engineer

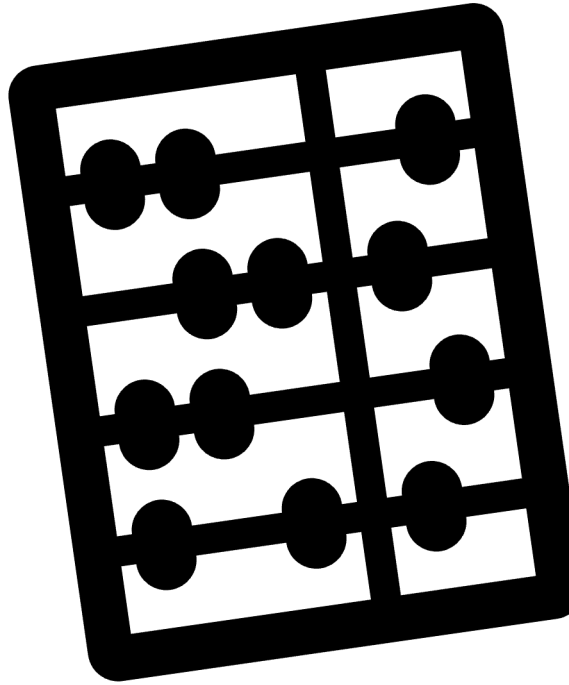
c.pratt@cablelabs.com

ONU Encryption Initialization



SIEPON MA - Approaches/assumptions:

1. SIEPON should *enable* authentication methods, while allowing the *policy* to be dictated/described by the operator
2. Credentials must be attested/verified
   - e.g. via challenge/response and hash/signatures
3. Trust store/lists must be operator-configurable (on OLT and ONU) and initialization/updates to the ONU trust store should be securely updatable by the operator via the OLT.
4. Initial AES key must ephemeral and mutually verified
   - To provide forward secrecy and prevent Machine in the Middle (MITM) attacks
5. Having mandatory authentication with simplified credentials is better than having optional/no authentication

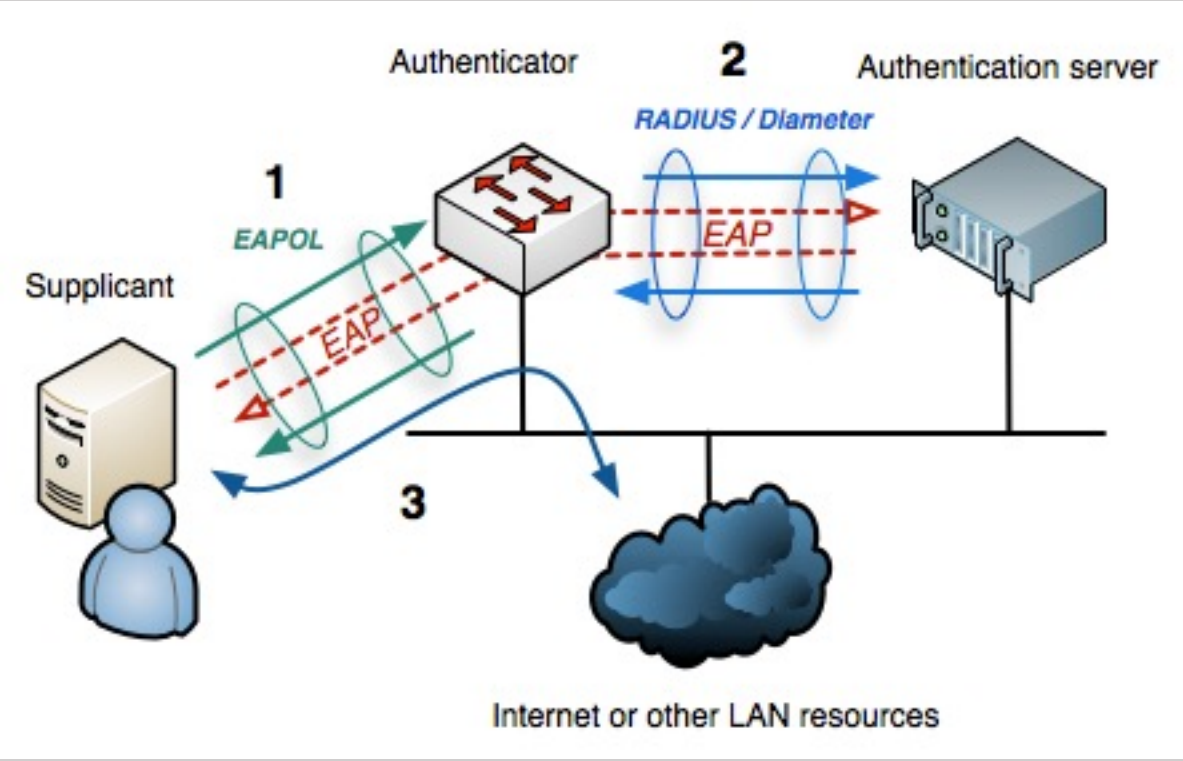


Questions to answer

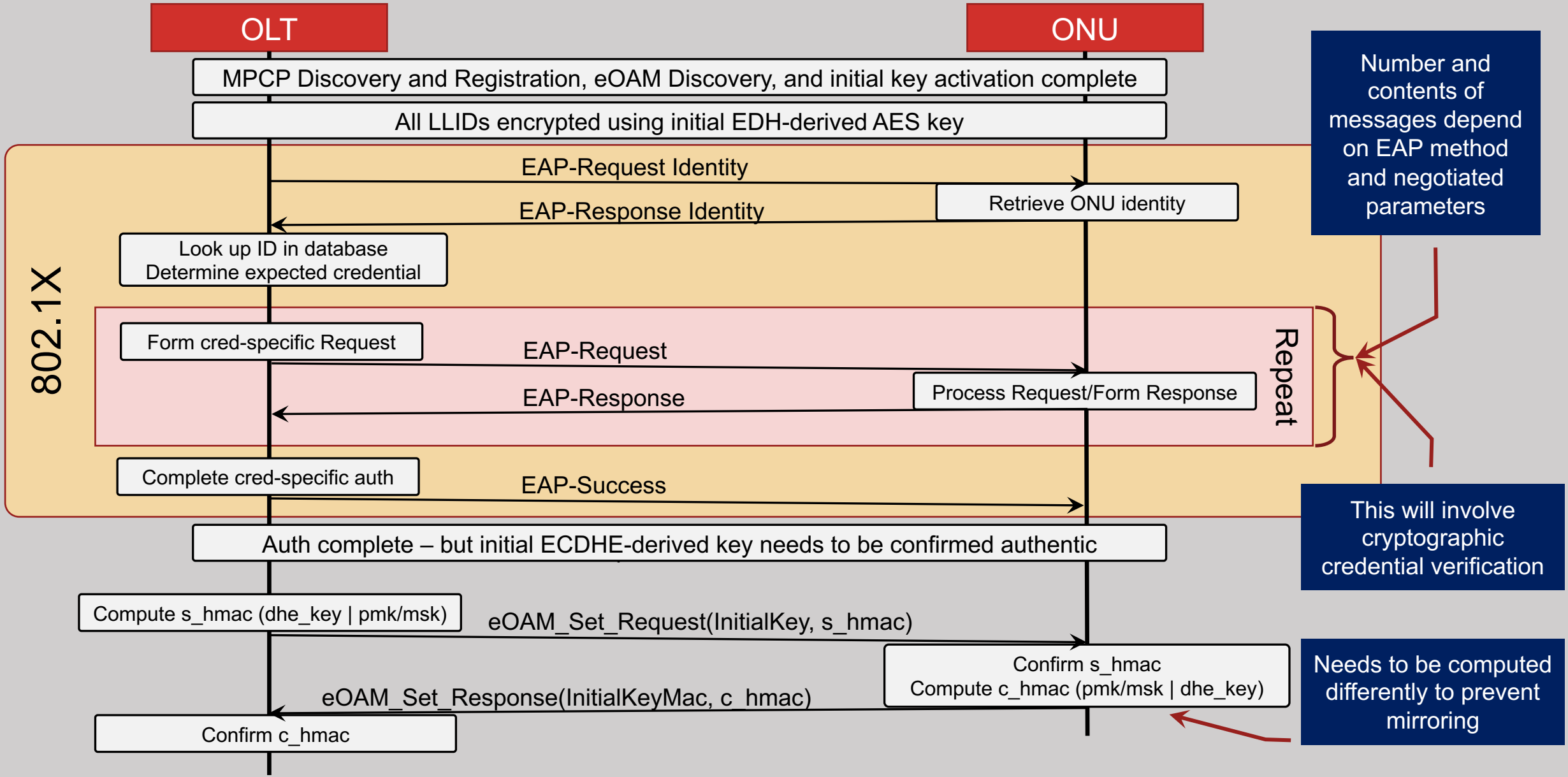
- Q1: How should authentication be performed?
  - Have looked into a couple options...
- Q2: What formats of credentials are allowed?
  - X.509 is widely supported and supports a wide variety of PKI systems, but has some complexities. Should we support more than one credential type and if so, which?
- Q3: How should initial authentication be performed?
  - What credential/key(s) should be built into the ONU for authentication?
  - What information should be provided by the installer/operator during onboarding to enable initial authentication?
- Q4: How to enable and configure OLT authentication?
  - ONU must have a way to validate the OLT to provide full mutual authentication, but how?

Q1:
How should authentication be performed?

- Proposal: Use 802.1X
  - Can deal with limited frame sizes
  - Concept of ”Controlled Port” and “Supplicant” matches up well with OLT and ONU, respectively
  - Allows for use of different credential types
  - Widely supported and maintained/updated technology



Authentication Flow using 802.1X



Q1 Discussion: How should authentication be performed?

- Had looked into straight encapsulation of TLS messages into OAM PDUs, but quickly ran into complexities
  - Many credentials and signatures are too large – so would have to come up with a fragmentation system
  - Would potentially be simpler, but more custom code (no library support)
- EAP can be a can of worms if underspecified
  - Open-ended options will reduce interoperability
  - Spec text should specify what can be allowed/disallowed and require ONUs to support both credential types (to provide operator flexibility)

Q2:
What formats of credentials should be supported?

- Proposal: Support 2 credential forms to give operators options
  - X.509 certificates: RFC-5280
    - Widely supported legacy technology, especially with existing PKI (Public Key Infrastructure) systems
    - Structure and coding are complicated
    - Requires a well-designed PKI system for robust authentication
  - Java Web Tokens with signatures (JWT/JWS): RFC-7519/RFC-7515
    - Widely supported in web applications and used in some IoT protocols
    - Structure and coding are simple
    - Can interoperate with PKI systems for robust authentication

Example of creating an ONU JWT/JWS

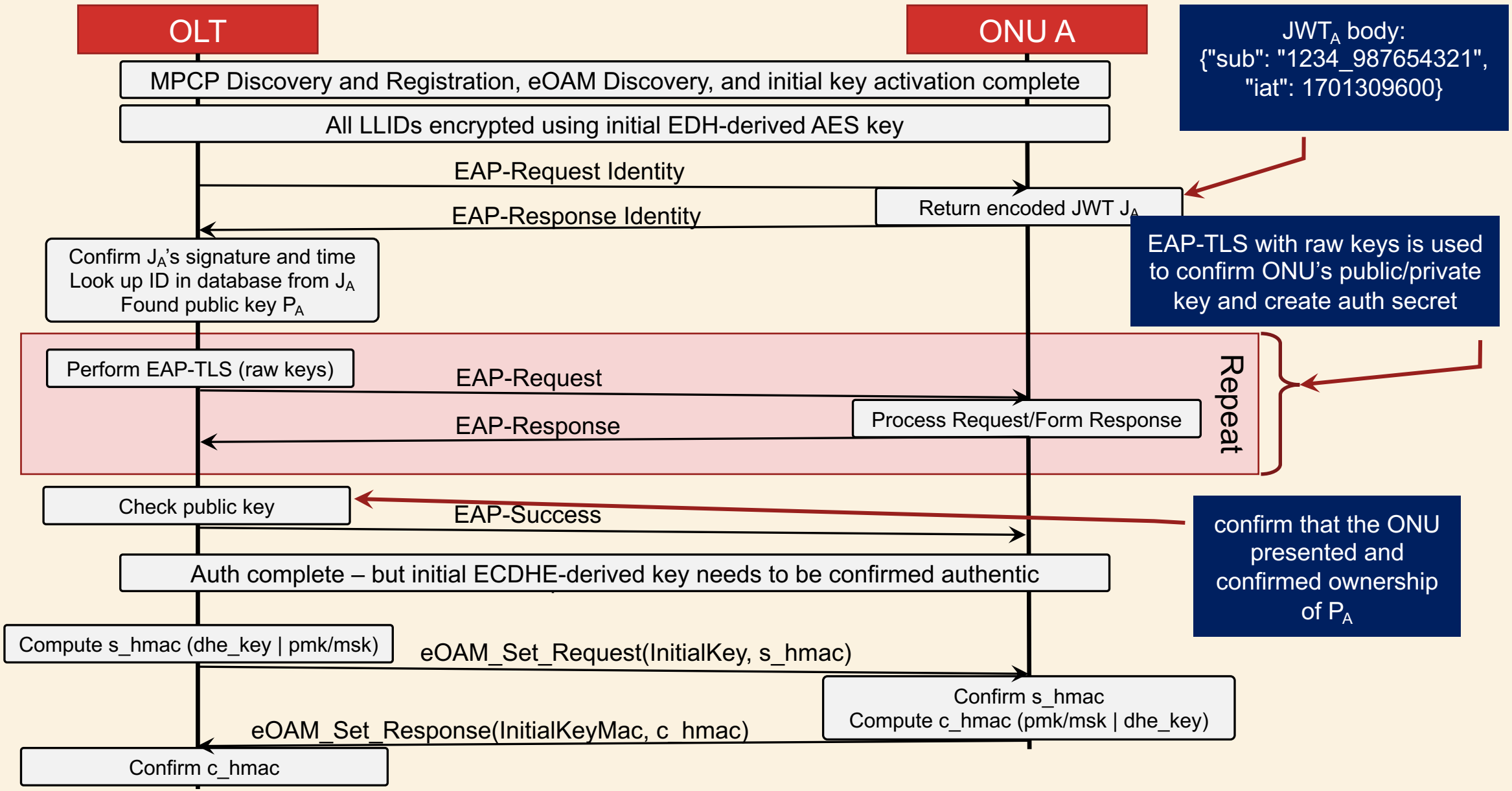
```
# Manufacturer code 1234, Serial number 987654321
header = {"alg": "ES256", "typ": "JWT", "kid": ”onu_1234_987654321_key"}
payload = {"sub": "1234_987654321", "iat": datetime.datetime.utcnow()}

jwt_token = jwt.encode(payload, ca_private_key, algorithm="ES256", headers=header)

print(f"Generated JWT ({len(jwt_token)} bytes):", jwt_token)
```

```
Generated JWT (220 bytes):
eyJhbGciOiJFUzI1NiIsImtpZCI6IjEyMzRfOTg3NjU0MzIxX2tleSIsInR5cCI6IkpXVCJ9.
eyJzdWIiOiIxMjM0Xzk4NzY1NDMyMV9rZXkiLCJpYXQiOjE3MDE1NjM0Nzh9.
TOp6n-zNDlTKoU9yykxllWFpsz1SRM9YVkVBrVydxNd3SGRI1nFrIUcy76juOMykVJ2kcOIzO91tLdqtM-A1Jg
```

Auth Flow with EAP-TLS JWT

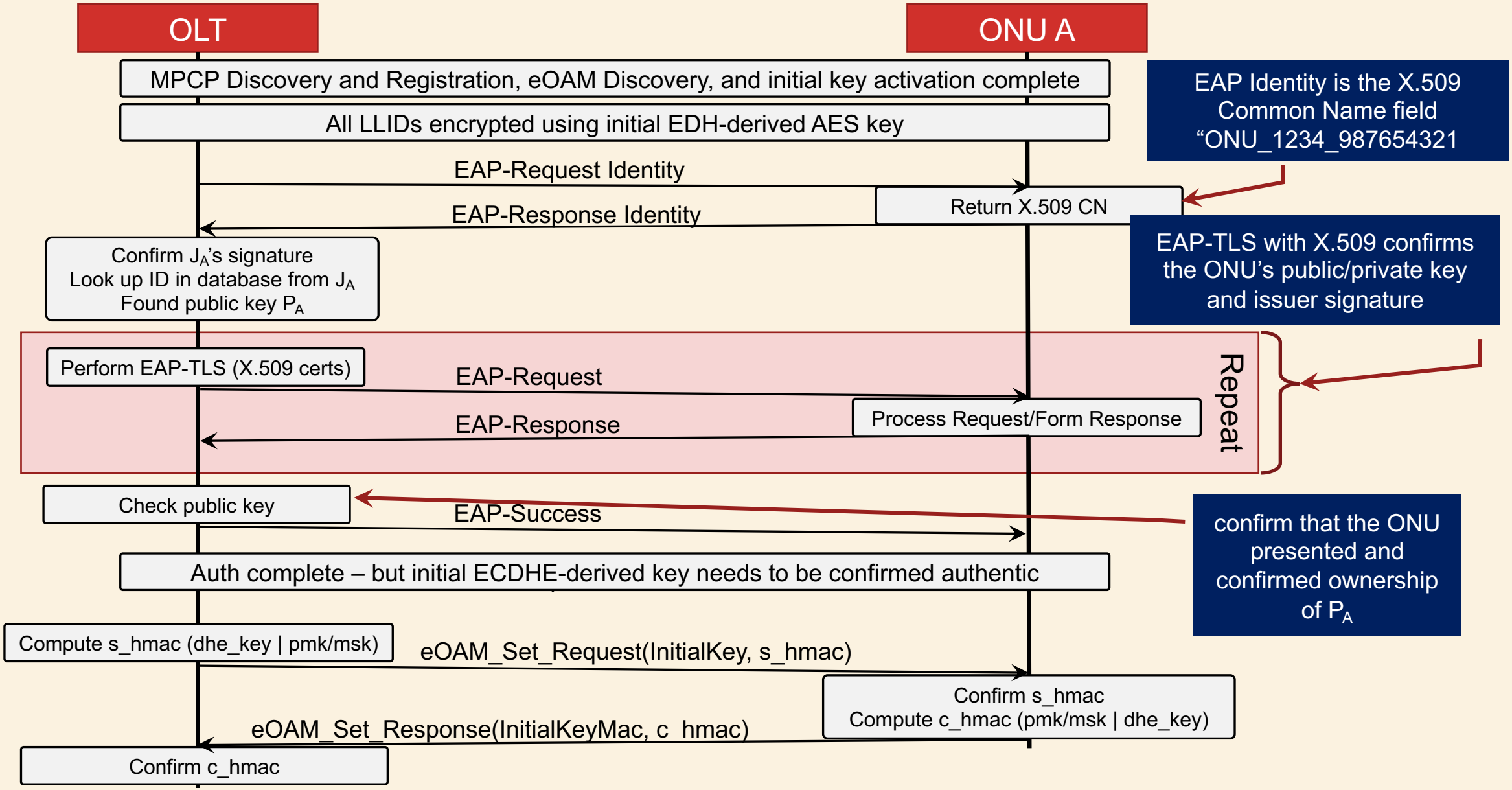


Example of creating an ONU X.509

```
# Manufacturer code 1234, Serial number 987654321
builder = x509.CertificateBuilder()
builder = builder.subject_name(x509.Name([
        x509.NameAttribute(x509.NameOID.COMMON_NAME, "onu_1234_987654321_key")]))
builder = builder.issuer_name(x509.Name([
        x509.NameAttribute(x509.NameOID.COMMON_NAME, "ACME CORP")]))
not_valid_before = datetime.utcnow()
not_valid_after = not_valid_before + timedelta(days=365)  # Valid for 1 year
builder = builder.not_valid_before(datetime.utcnow())
builder = builder.not_valid_after(datetime.utcnow() + timedelta(days=365))
builder = builder.serial_number(serial_number)
builder = builder.public_key(ca_private_key.public_key())
```

```
-----BEGIN CERTIFICATE-----
MIIBIzCBy6ADAgECAgQSNFZ4MAoGCCqGSM49BAMCMBQxEjAQBgNVBAMMCUFDTUUg
Q09SUDAeFw0yMzEyMDQwMjE2MzJaFw0yNDEyMDMwMjE2MzJaMCExHzAdBgNVBAMM
Fm9udV8xMjM0Xzk4NzY1NDMyMV9rZXkwWTATBgcqhkjOPQIBBggqhkjOPQMBBwNC
AAR+1QQz1s/6hU4dWxUeql8xkFp731Gh/NBn+d2E9gHj5bnOnznAS1MGoDYbJh40
ty2jQa+86IktACXlv+jQ87P8MAoGCCqGSM49BAMCA0cAMEQCIAuM41m8IYkm1WsT
7GpxOnPM4j9AEuuJGhjvtBDUSM05AiAxzB3bZWdU69xKE75OVSwTnr8oK+gXU7PQ
vsZ9qDJzGQ==
-----END CERTIFICATE-----
```

Auth Flow with EAP-TLS with X.509



Q2 Discussion: What formats of credentials should be supported?

- Proposed solution notes:
  - Both the JWT and X.509 auth essentially do the same thing:
    - Confirm ownership of public key and associated identity
    - Confirm issuer
    - Confirm valid time period
  - Credential issuance isn't covered here (discussed below), but the type and makeup of the credential is entirely at the discretion of the OLT
    - Can switch from JWT to X.509, and vice-versa
    - Necessarily should issue new credentials to deal with dates
  - Authentication of the OLT must be performed using the same mechanism as the ONU – no mixing and matching
    - To enable JWT MA, OLT must provide a trust store to validate OLT JWTs
    - To enable X.509 MA, OLT must provide a trust store to validate X.509 certificates

Intent here is to require authentication using one form or the other – "no auth" should not be an option.

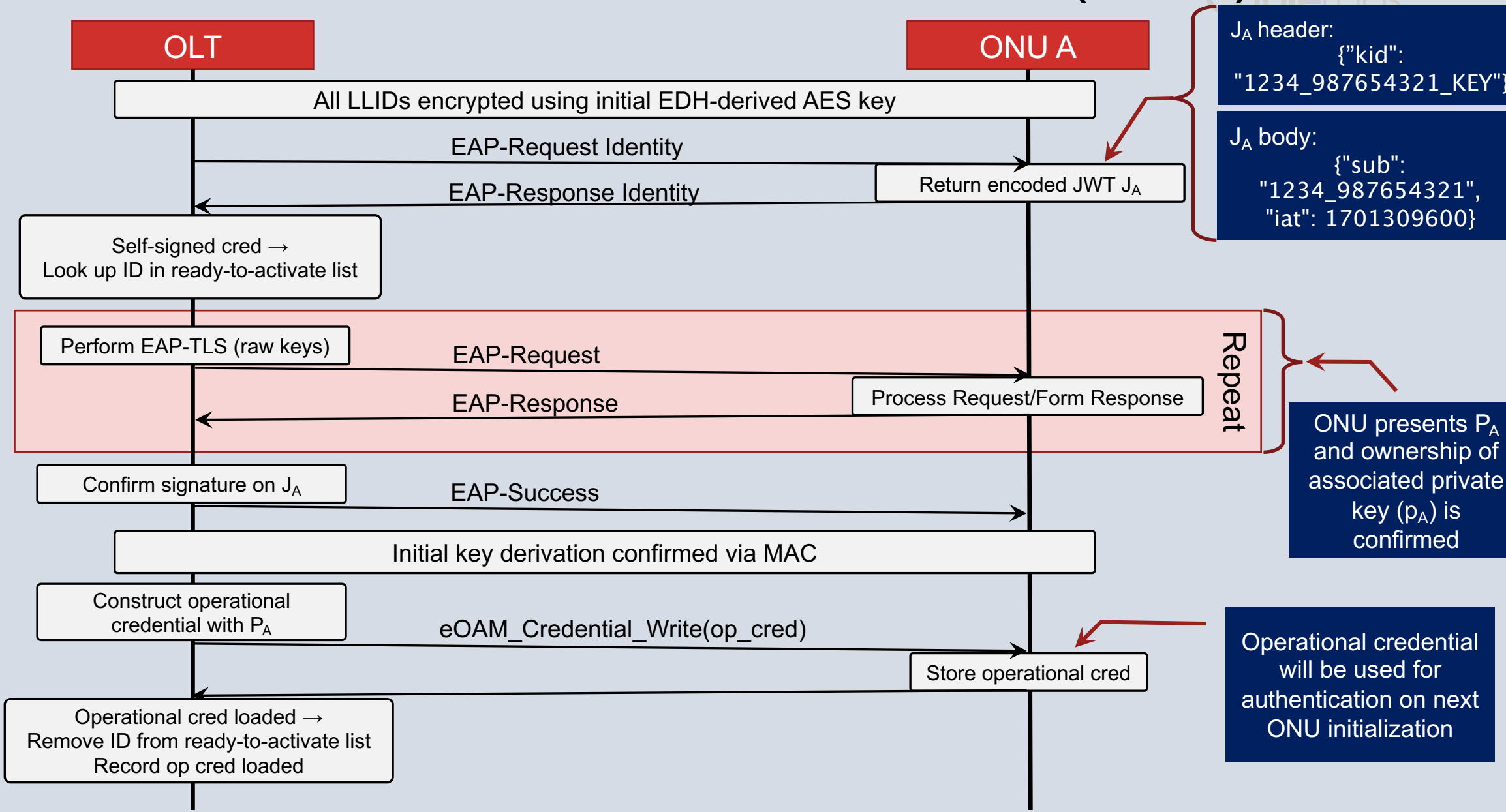
Q3:
How should ONU initial authentication be performed?

- ***Initial authentication***: When ONU is first installed/delivered
- ***operational credential***: Credential loaded onto ONU by operator for daily operation (via the OLT)
  - signed using operator- or operator-outsourced key/CA
- ***activation credential***: Credential provided by ONU to enable it to be put into operation
  - ONU-signed key (self-signed) with two-factor authentication, or
  - Operator-signed if preloaded out-of-channel
  - Has a special token out of large address space (non-sequential)
- **Assumptions**:
  - All credentials bind the ONU SN to a public key
  - public/private authorization keypair is initialized at factory or on first power-on – with private key well protected (already specified)
  - Device serial number (SN) will be put into a “ready to activate” (RtA) list when installed by technician, sent to a user, or activated via phone/app/etc
  - Activation time period should be limited

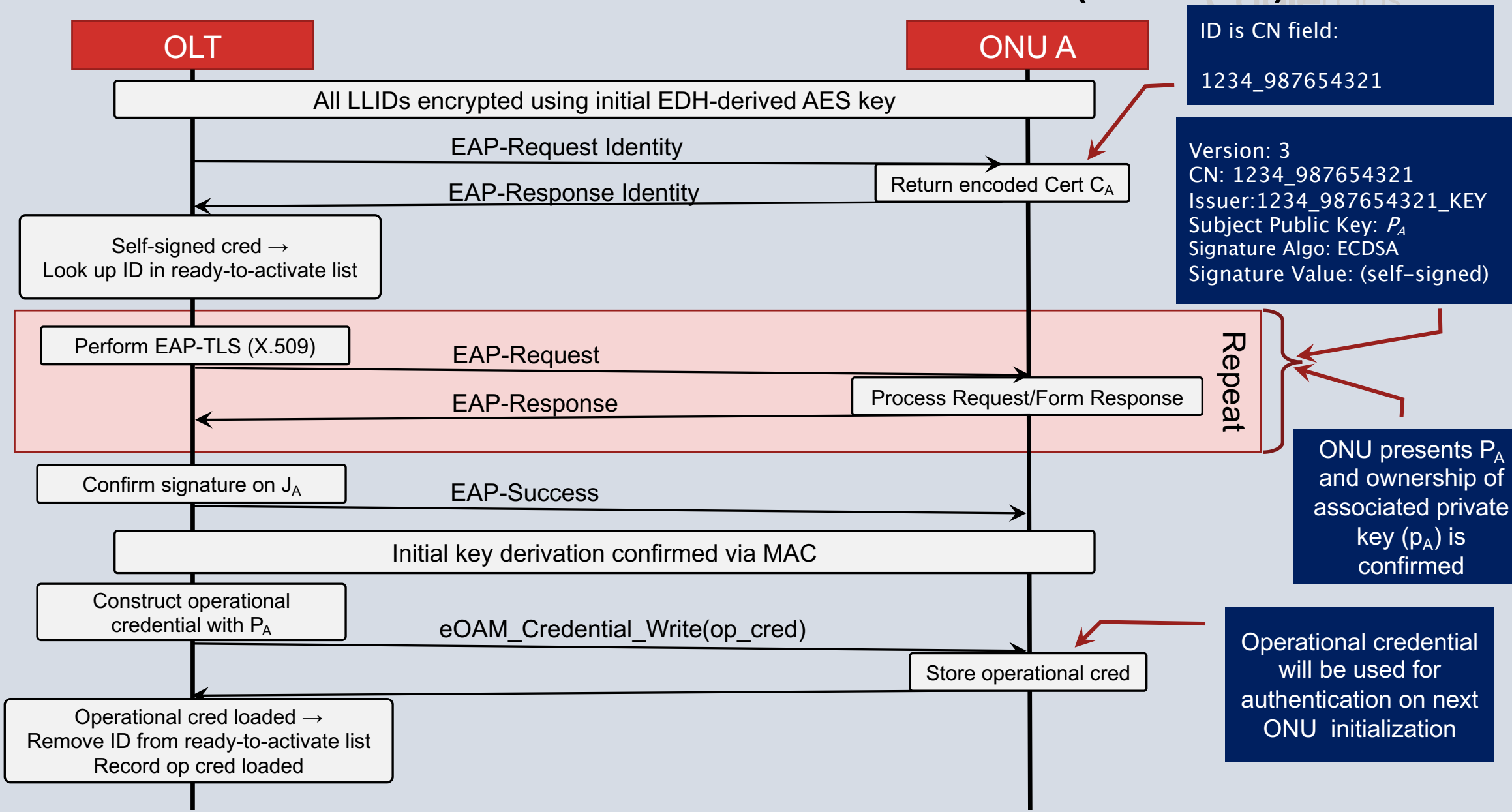
Operational vs Activation Credentials

Operational Credential	Activation Credential
For ongoing authentication of the ONU	Just for onboarding the ONU
Certificate or JWT containing the ONU ID, public key, and anything else the operator wants	Certificate or JWT containing the ONU ID, public key, and onboarding token
Signed by operator or third-party	Self-signed
Provided by operator to provide robust, ongoing trust	Provided by manufacturer to provide initial trust
Robust and attestable	Not robust on its own – depends on multi-factor authentication

Auth Flow for initial authentication (JWT)



Auth Flow for initial authentication (X.509)



Q3 Discussion: How should ONU initial authentication be performed?

- Self-signed credentials with 2-factor authentication
  - PROs:
    - Simple for vendors – just have to have a persistent public/private key to generate
    - Doesn't require powering on or physical interaction with the ONU
    - Could be augmented with a passphrase – would require a bit more specification
  - CONs:
    - Theoretical window of opportunity for rogue ONU to get onboarded *
    - Some logic required to support updating of credentials from activation credential to operational credential **

Q4:
How to enable and configure OLT authentication?

- To prevent possible impersonation of an OLT on a PON network, the ONU can be configured to authenticate the OLT it connects to (Mutual Authentication)
- Using the same mechanisms described for ONU authentication, an OLT can assert an identity/credential and associated public key
  - The public key is verified by verifying ownership of the private key.
  - Metadata/identity is verified via signature verified using the public key

Q4:
How to enable and configure OLT authentication?

- For the ONU to authenticate the connected OLT, it needs to be provided some criteria about what OLTs are “authentic”. This can take different forms:
  - ONU can be given an explicit list of what OLT IDs and public key signatures are legitimate (presumably a small number)
  - ONU can be given a value that must be present in a secondary field of the credential (e.g. “GROUP_A”)
  - ONU can be given a certificate(s) which the OLT credential must be signed by (e.g. X.509 CA certificates)
- The database of what credentials/credential forms can be trusted by an entity during authentication is called a “trust store”.

Q4:
How to enable and configure OLT authentication?

- Assumptions:
  - The number of OLTs an ONU should be authorized to connect to is small
  - Operators will want to phase in OLT authentication
  - Operators will want to be able to update ONU's OLT trust store on-demand
  - Trust store needs to be empty when supplied by the vendor – but may be operator-loaded (similar to operational credential)
  - Trust store needs to be persistent across power cycles, cleared on factory reset
  - It's acceptable for the ONU to trust the first OLT it connects to – by virtue of the activation credential
    - "Trust On First Use – But Verify" (via 2-factor authentication)

Q4:
How to enable and configure OLT authentication?

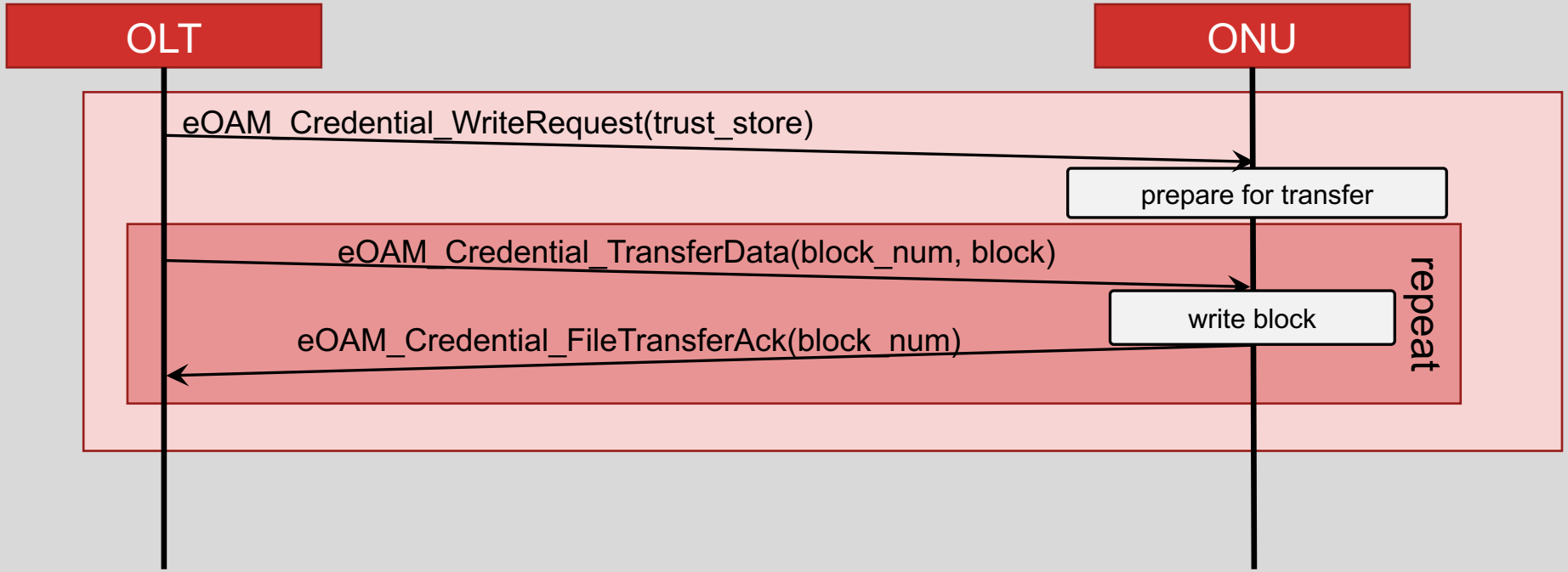
- Proposal:
  - Allow for different forms of trust specification as list of:
    - Explicit OLT ID + pubkey signature.
      - e.g. “OLT.12345678”: ”0x505387c4688063ff...2af552d”
    - Explicit name/value pair + pubkey signature
      - e.g. “olt_group”: “1A”, “pubkey_signature”: ”0x505387c4688063ff...2af552d”
    - X.509 CA certificate:

      e.g.

```
Certificate:
    Data:
        Version: 3
        Serial Number: 12345 (0x3039)
    Signature Algorithm: ecdsa-with-SHA256
        Issuer: CN=ACME INTERNET
        Validity
            Not Before: Dec 1 08:22:19 2023 GMT
            Not After : Dec 9 08:22:19 2033 GMT
        Subject: CN=ACME OLT Network
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
0001: 04 63 D8 A3 E8 65 8A 8E 83 30 94 9E DC 60 1E 53 ...
    Signature Algorithm: ecdsa-with-SHA256
         30:44:02:20:65:76:c3:02:3b:47:59:f9:4e:4d:b7:91:7d:e2...
```

Q4:
How to enable and configure OLT authentication?

- Proposal:
  - Provide a file transfer paradigm updating trust store by the OLT



Q4 Discussion:
How to enable and configure OLT authentication?

BACKUP MATERIAL