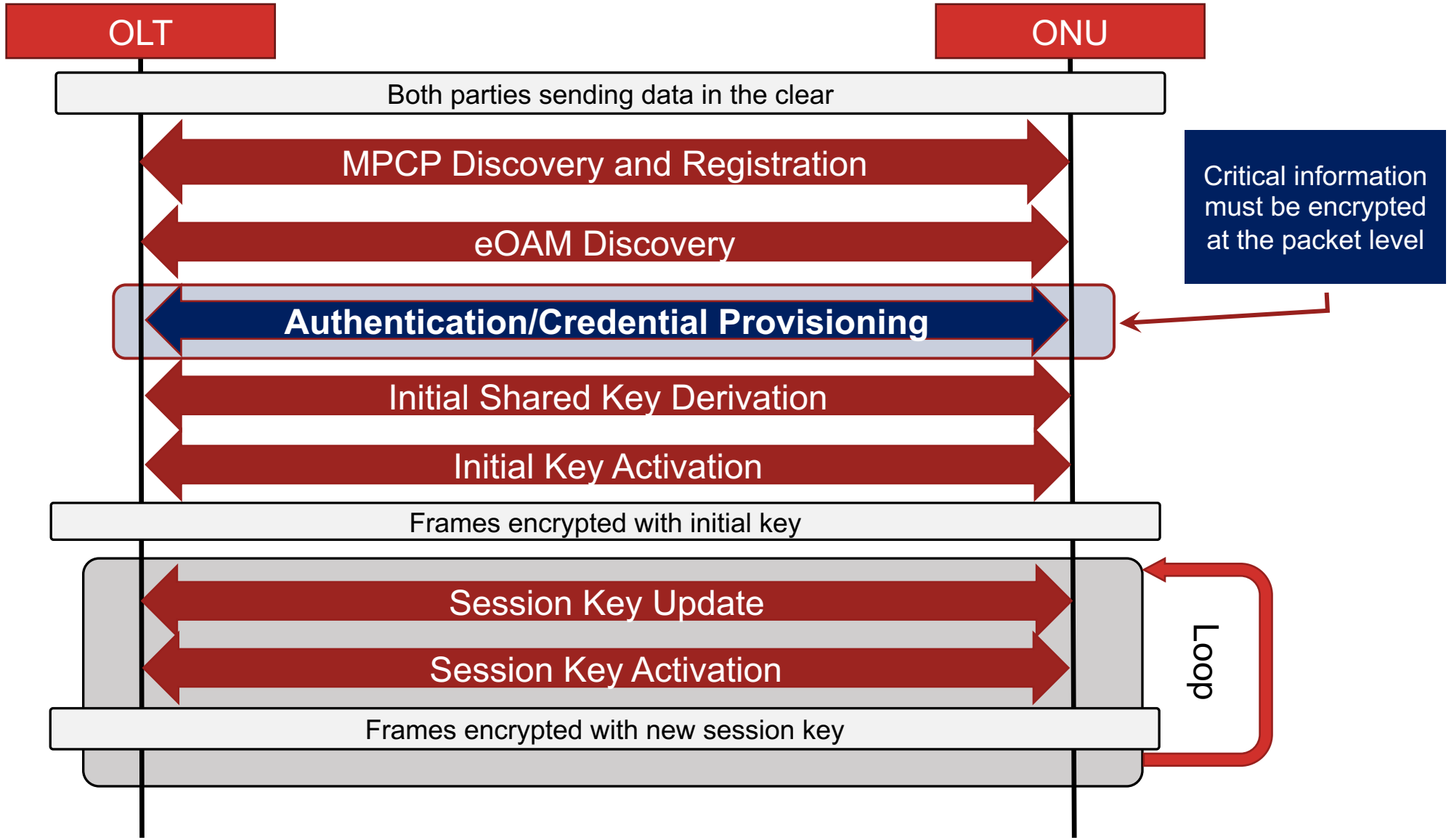
SIEPON.4 Authentication Proposal
v0.5 – 2023-12-14
Part 1: Initial Key Derivation & Credentials

Craig Pratt | Lead Software Architect

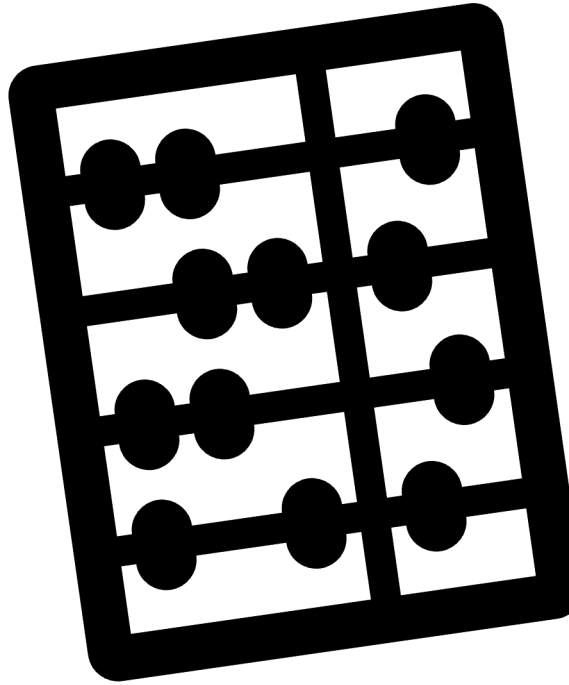
c.pratt@cablelabs.com

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



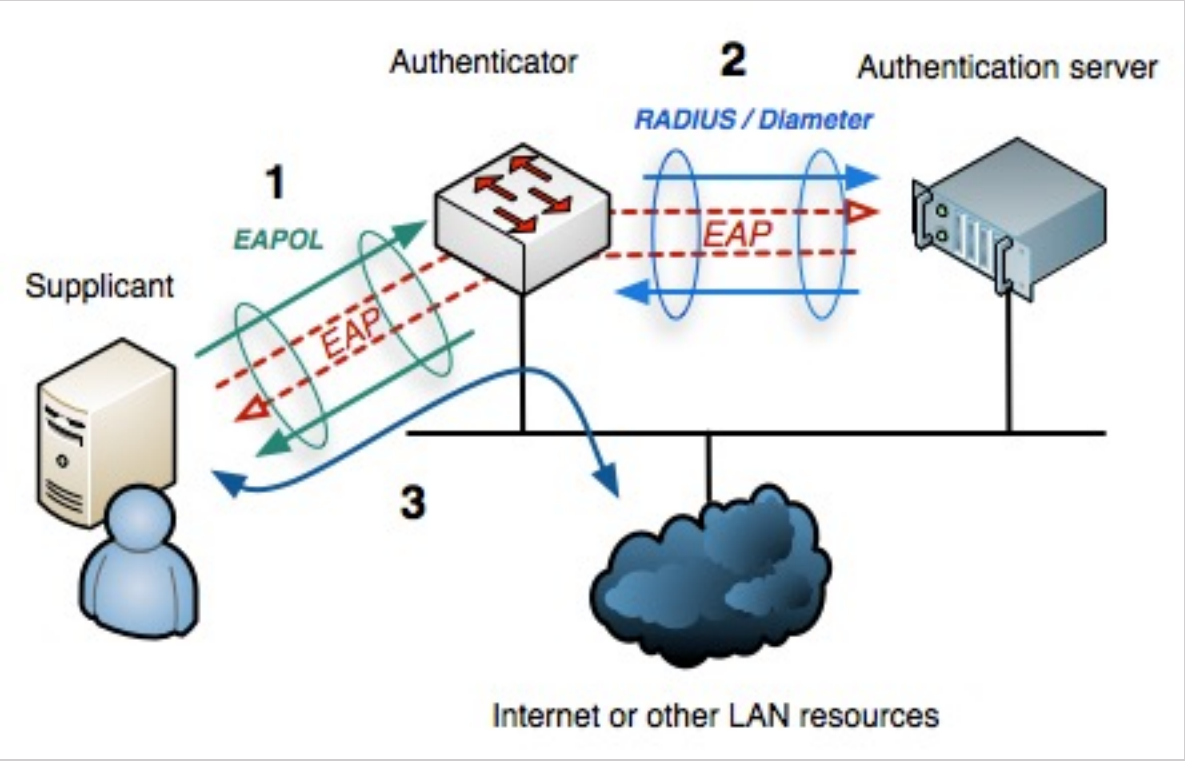
Questions to answer

- **Q1b: How should authentication be performed?**
  - **Have looked into a couple options...**
- Q2b: 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?

Q1b:
How should authentication be performed?

- Proposal: Use 802.1X with EAP-TLS v1.3
  - 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 (pubkey & X.509)
  - Widely supported and maintained/updated technology
  - **TLS 1.3 encrypts sensitive handshake fields (like the cert/public key) and provides key data export**

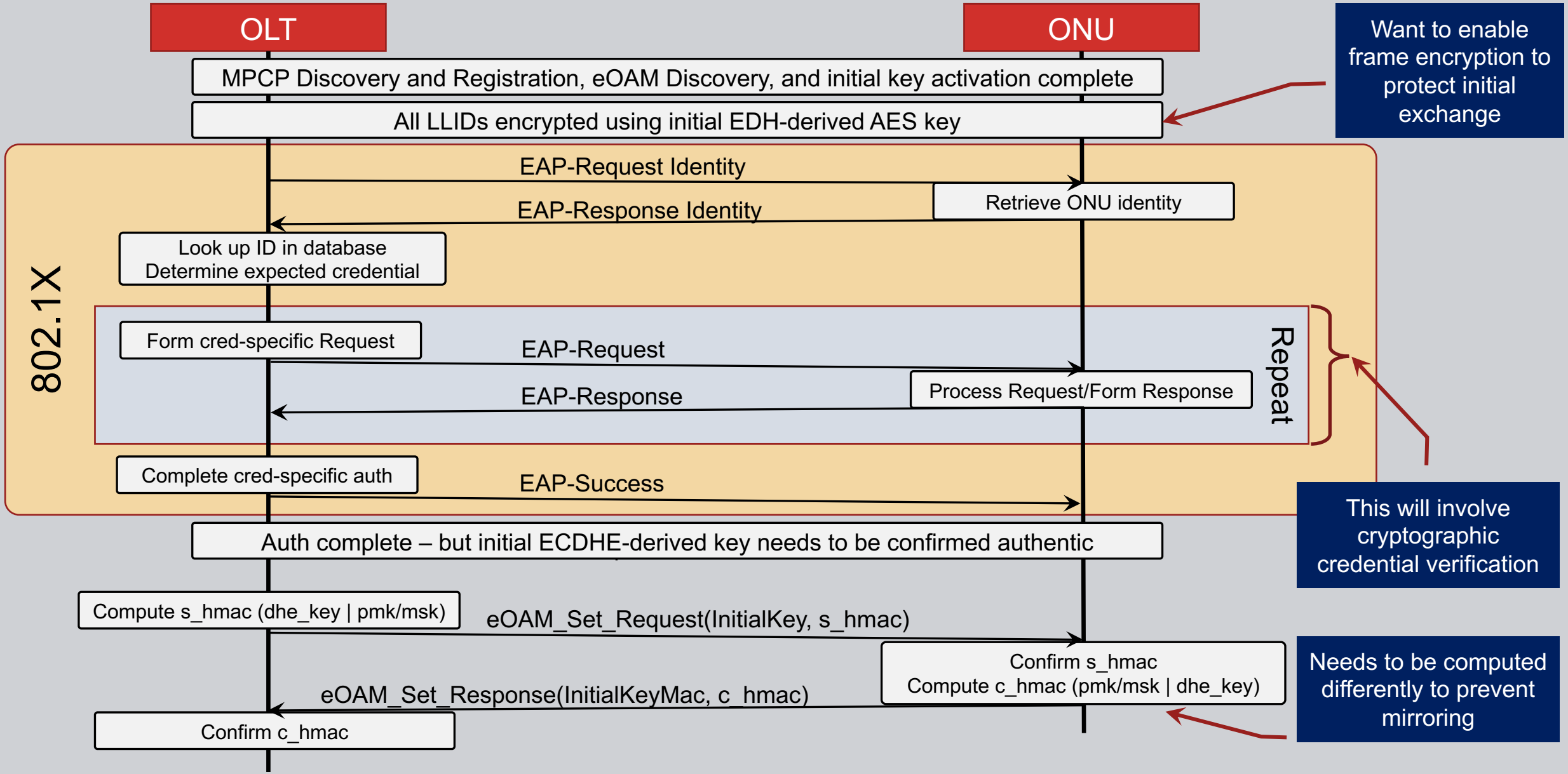
NEW



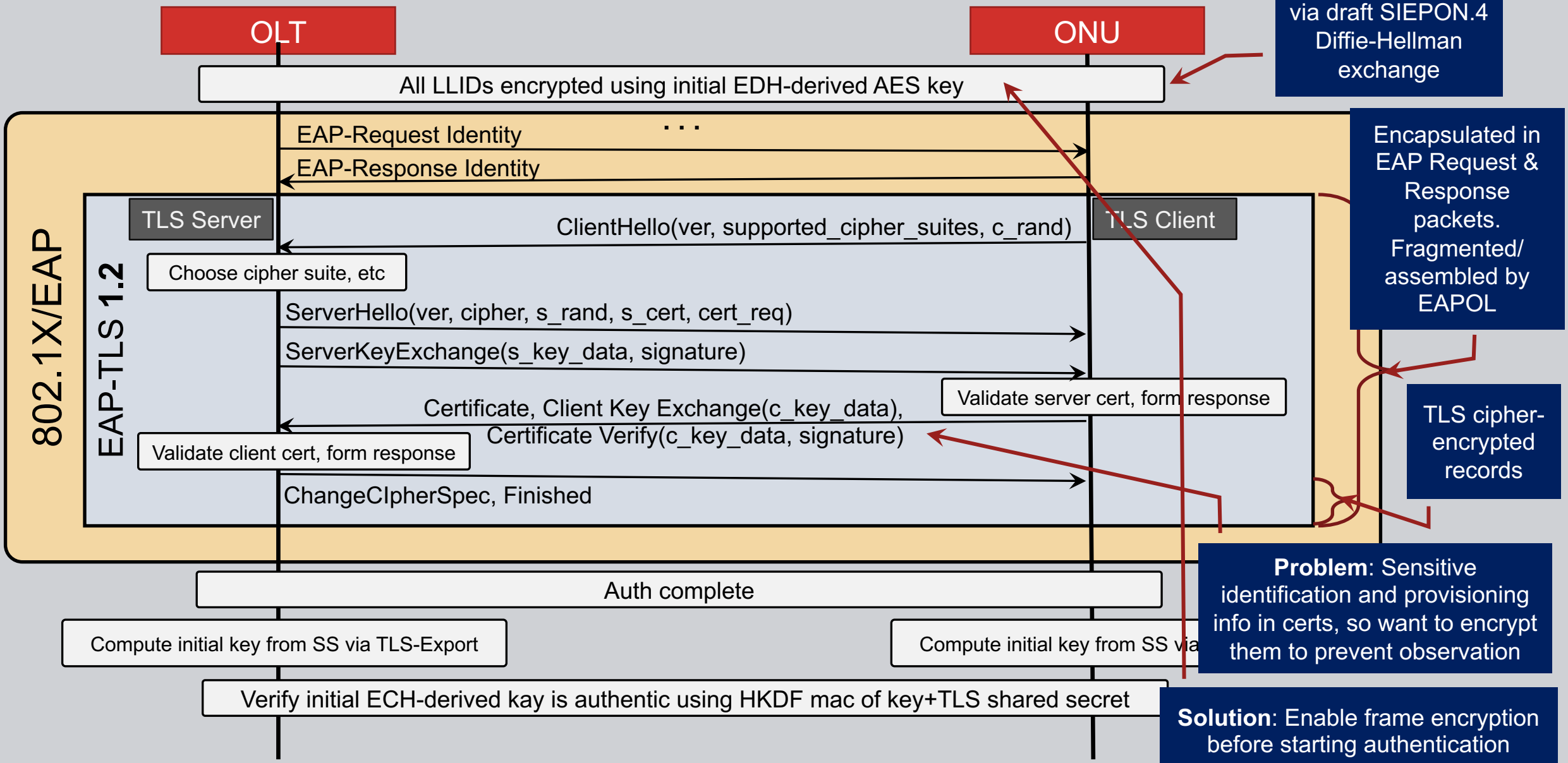
Q1b:
How should authentication be performed?

- Secondary discoveries
  - TLS 1.3 performs ECDH first to provide ”early data” feature and encrypts the handshake data – including certificates and public keys. (RFC-8446 4.2.10)
    - Was working under the assumption that we need to enable frame-level encryption to encrypt the handshake (to protect certs and pubkeys) → not necessary in TLS 1.3
    - Also means we would be performing 2 ECDH exchanges if/when SIEPON did its own → twice as much computation and extra round trips
    - Need to not disclose sensitive data in EAP Identity, since it’s not protected
  - RFC-5705 (TLS Exporters) provides a standard way to extract keying material
    - Can use this instead of specifying HKDF to derive keys from the premaster secret
    - Getting to the premaster secret is not well supported by most TLS libraries
- Recommendation:
  - Take advantage of TLS 1.3’s ECDH early_data/PSK for handshake
  - Use TLS Exporter for establishing initial key

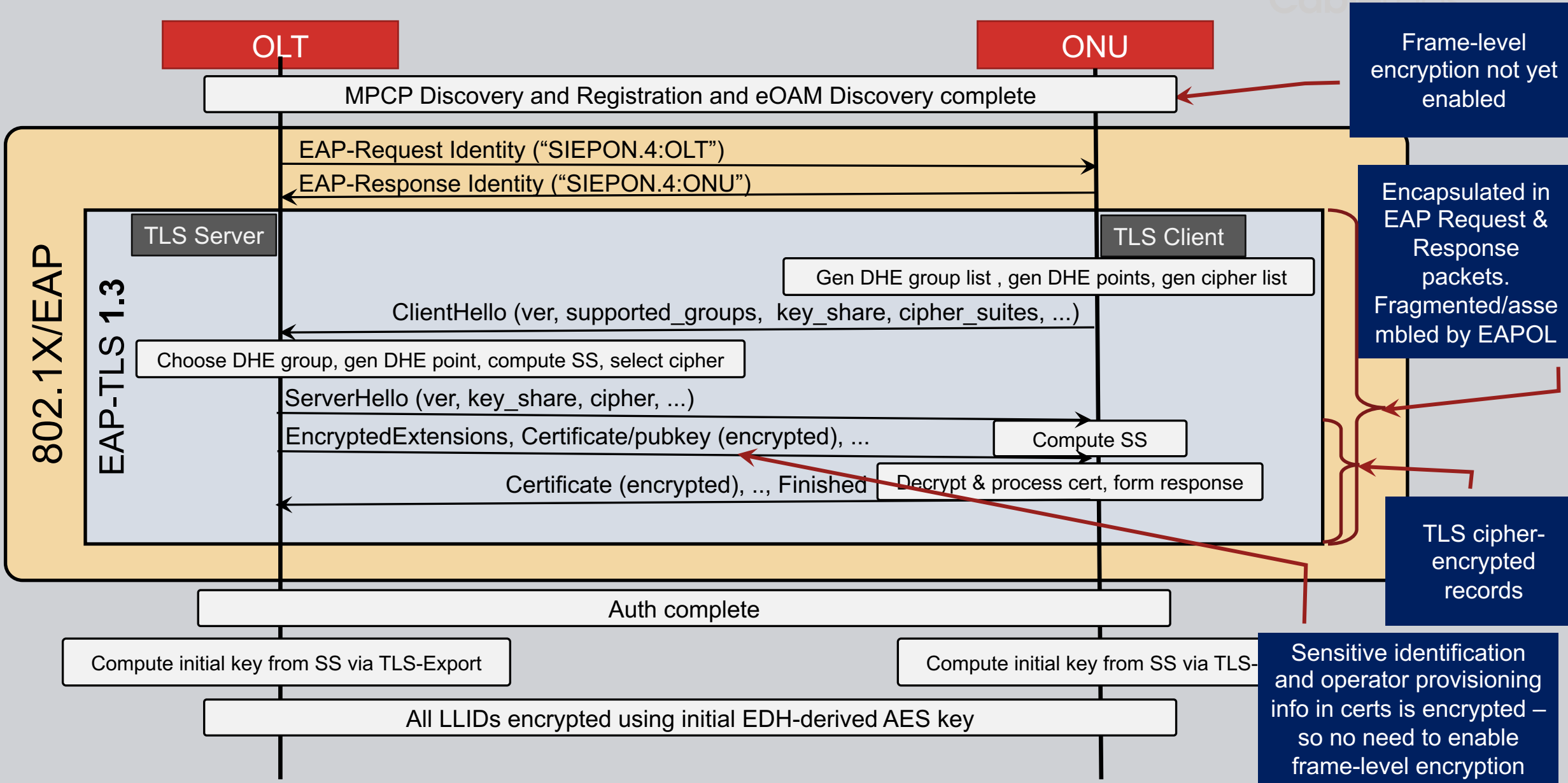
Authentication Flow using 802.1X (initial proposal)



Details we didn't get into last meeting...

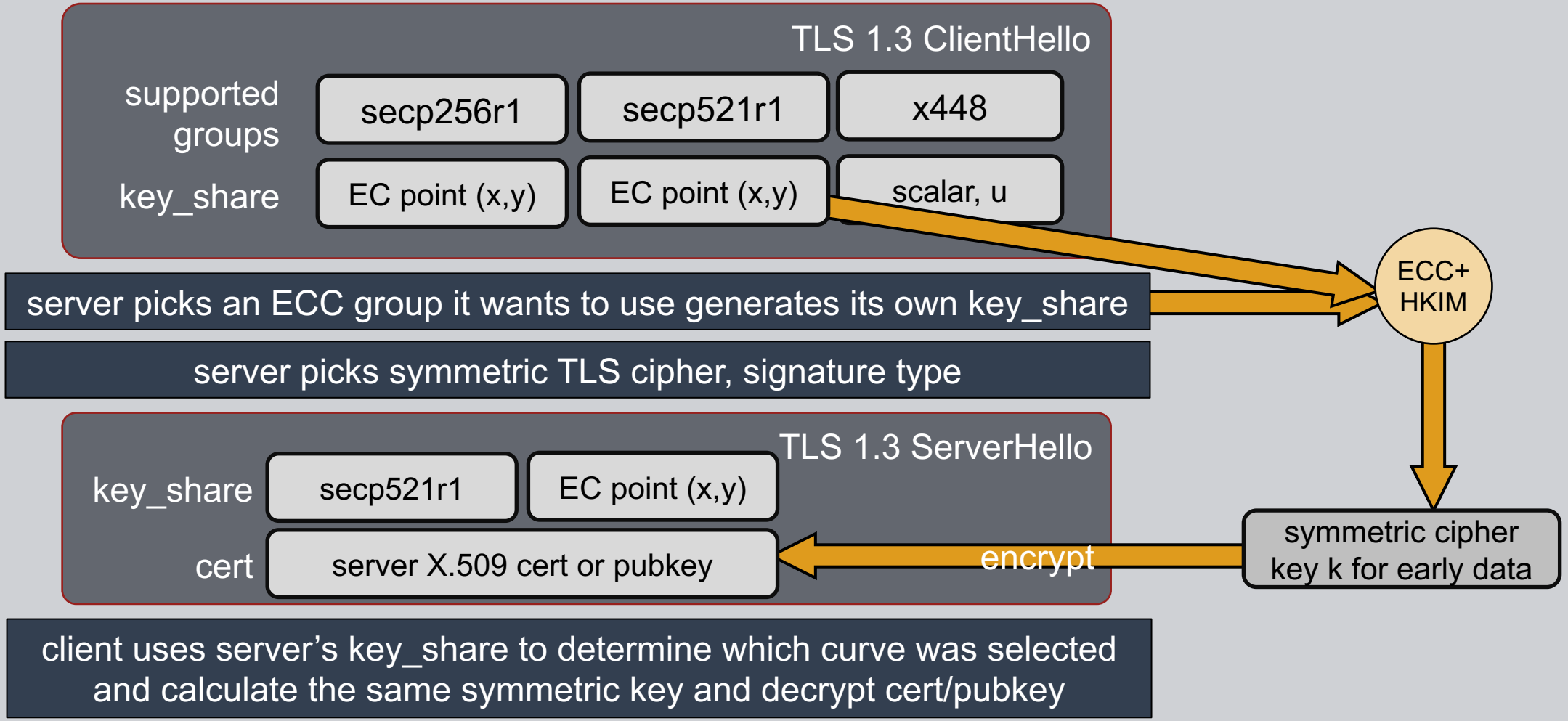


Authentication Flow using 802.1X (“new and improved”)



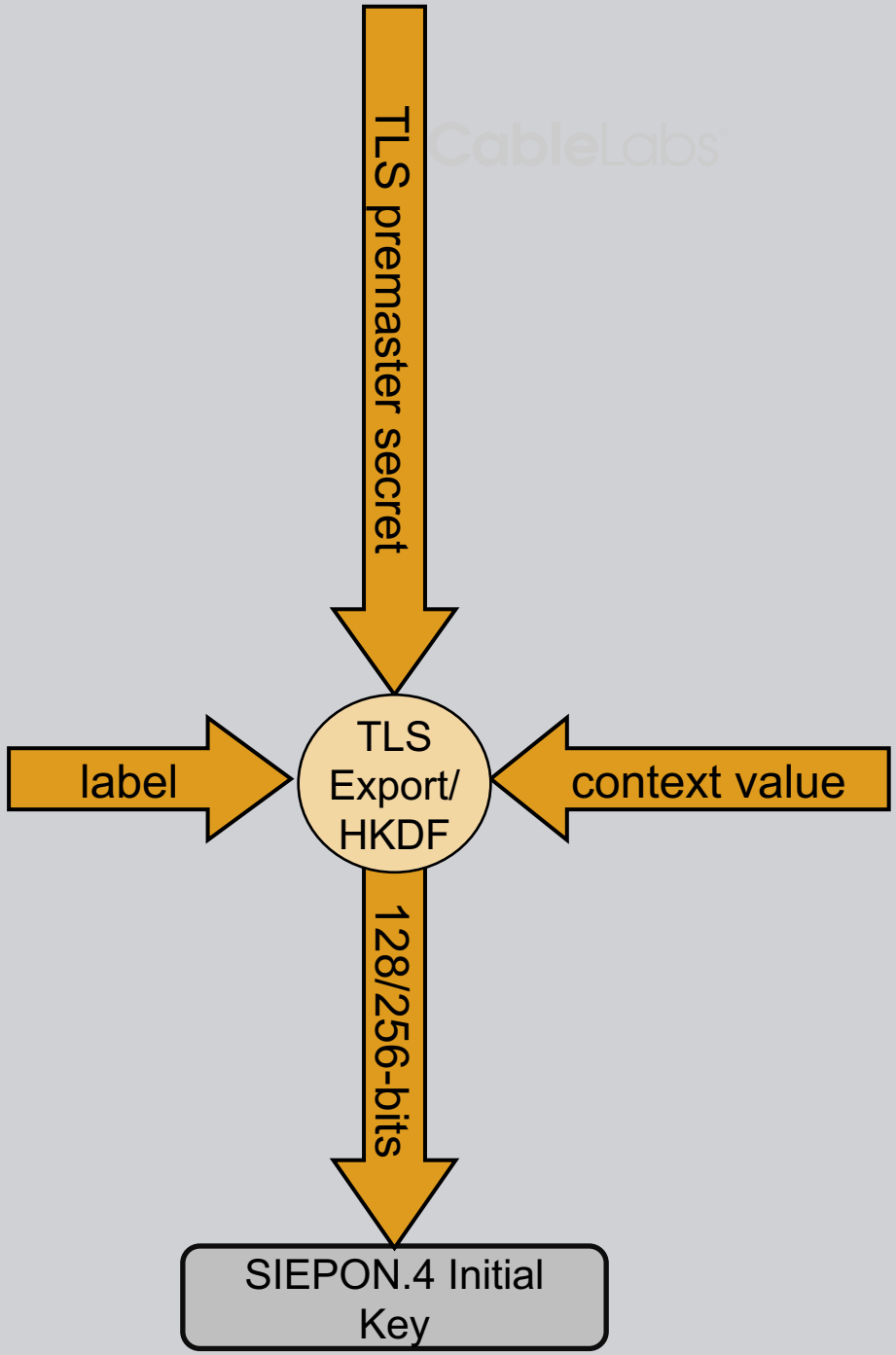
How it works - TLS 1.3 ”early data”

- TLS 1.3 client enumerates capabilities separately instead of a flat list of “cipher suites” (RFC-8446 7.5, RFC-5705)



SIEPON.4 Initial Key Derivation

- TLS 1.3 Early Data provides ephemeral key management, but now how do we derive the SIEPON.4 initial key?
  - A mandatory TLS extension called “TLS Exporter”
  - Uses HKDF (HMAC-based Key Derivation Function), which can derive a key of any length from a shared secret – like the SS you get from a EC Diffie-Hellman exchange
  - Doesn’t expose the TLS master secret/pre-master secret
  - Well supported by existing TLS libraries
  - Example...



SIEPON.4 Initial Key Derivation via TLS Export

- OpenSSL w/ TLS 1.3 (server)

```
if (SSL_accept(ssl) <= 0) {
        ERR_print_errors_fp(stderr);
    } else {
        printf("TLS handshake successful\n");

        // Export keying material
        const char* label = "EXPORTER_SIEPON4";
        const size_t len = 32;  // Length of keying material
        unsigned char keying_material[len];

        SSL_export_keying_material(ssl, keying_material, len, label, strlen(label), NULL, 0, 0);
        printf("Exported Keying Material: ");
        for (size_t i = 0; i < len; i++) {
            printf("%02x", keying_material[i]);
        }
```

```
$ ./tls-server-exporter-1
Server listening on port 8080...
Connection accepted from 127.0.0.1:57773
TLS handshake successful
Exported Keying Material: 81a764a880d3505772ee6907cdd8161be9e6024fdcfd3e96a972c52376191c7a
```

SIEPON.4 Initial Key Derivation via TLS Export

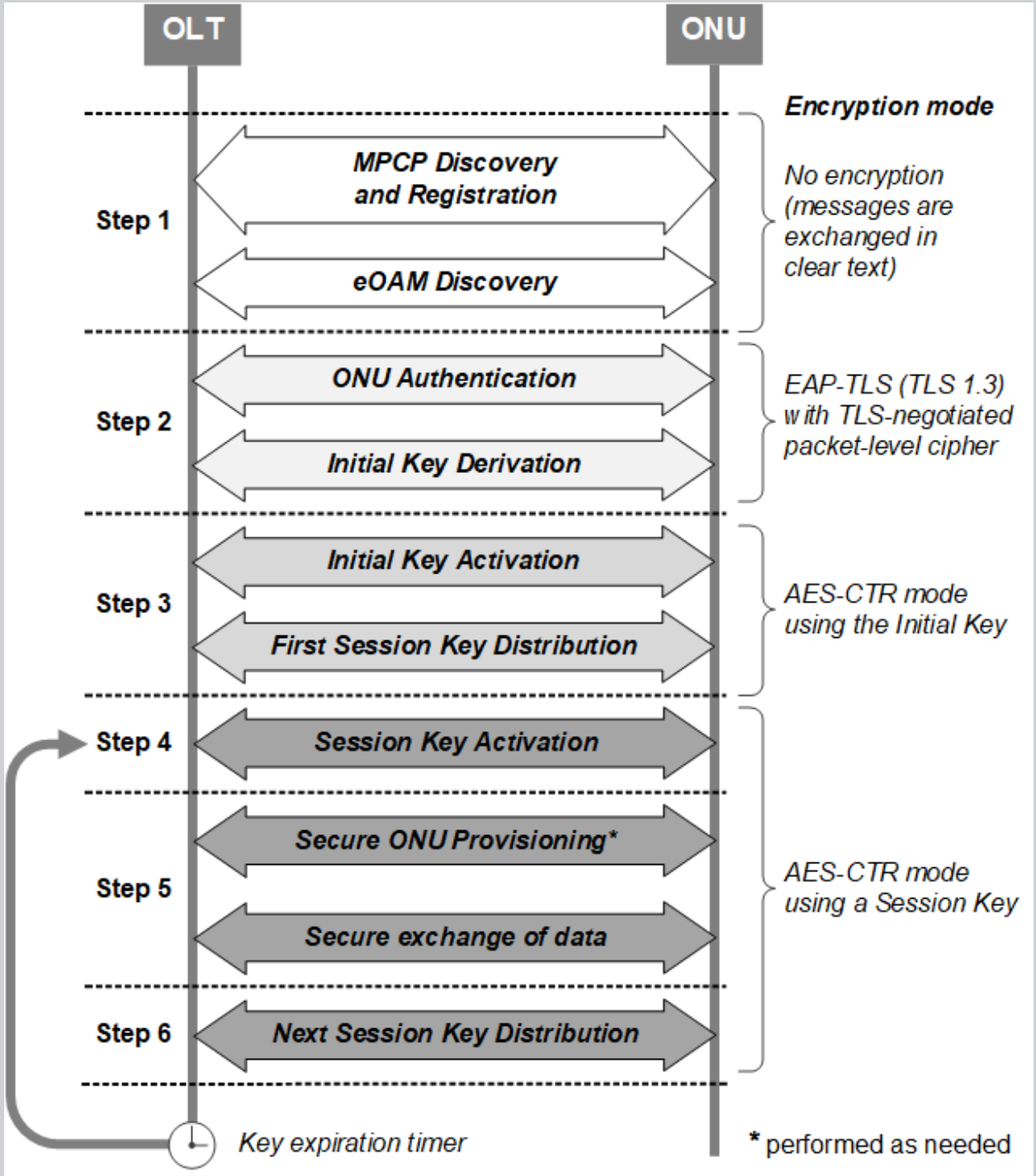
- OpenSSL w/ TLS 1.3 (server)

```
if (SSL_connect(ssl) <= 0) {
        ERR_print_errors_fp(stderr);
    } else {
        printf("TLS handshake successful\n");

        // Export keying material
        const char* label = "EXPORTER_SIEPON4";
        const size_t len = 32;  // Length of keying material
        unsigned char keying_material[len];

        SSL_export_keying_material(ssl, keying_material, len, label, strlen(label), NULL, 0, 0);
        printf("Exported Keying Material: ");
        for (size_t i = 0; i < len; i++) {
            printf("%02x", keying_material[i]);
        }
```

```
$ ./tls-client-exporter-1
TLS handshake successful
Exported Keying Material: 81a764a880d3505772ee6907cdd8161be9e6024fdcfd3e96a972c52376191c7a
```

Updated Initialization Sequence

- Auth first
- Single (EC)DHE
- Extraction from premater secret (DHE)



Q1b Discussion: How should authentication be performed?

- EAP can be a can of worms if underspecified
  - Open-ended options will reduce interoperability
  - Need to specify enough to achieve consistent interoperability
- TLS 1.3 allows for great simplification and reduced specification
  - **Ephemeral Diffie-Hellman method enumeration, selection and process doesn't need to be in SIEPON.4 – TLS has it covered**
  - **For security reasons, implementations don't like to expose the premaster secret via API or other means. TLS Export eliminates this issue and reduces specification in SIEPON.4**
    - We don't need to specify use of HMAC and HKDF in SIEPON.4 – it's all in TLS 1.3/RFC-5705