CableLabs®

SIEPON Security Initialization Proposal v1.4 – 2023-08-10

CableLabs

Craig Pratt | Lead Software Architect

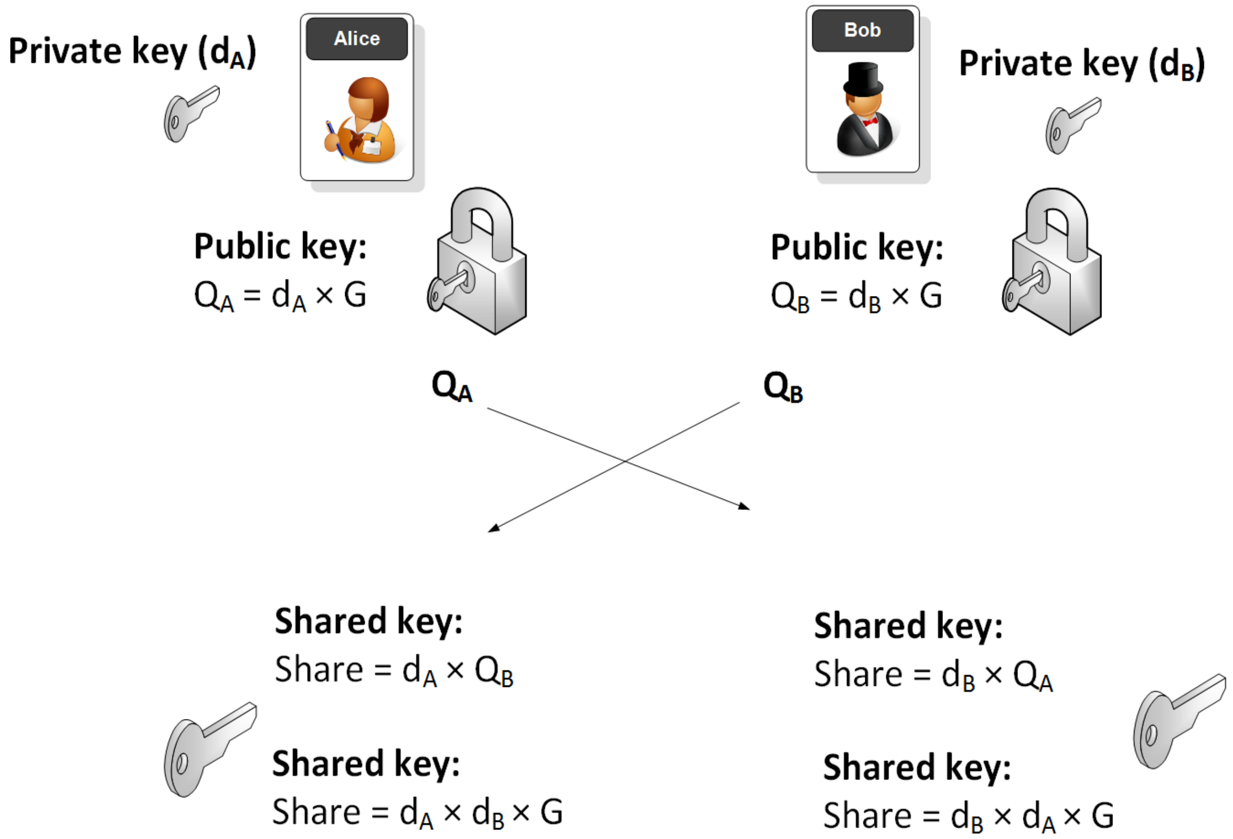
c.pratt@cablelabs.com

SIEPON Security Initialization

- **Strategy: Address authentication and initial key establishment separately**
  - Define initial encryption key establishment first, since it's critical for initialization of the key distribution protocol.
  - Define authentication second – determining forms of authentication and which parties can be authenticated
  - The establishment of the initial ONU encryption key will only be initiated by the OLT if/when it successfully authenticates the ONU
    - i.e. In ONU initialization, authentication should come first, followed by initial key establishment, followed by encrypted data exchange. We're completing the logic in the reverse order.

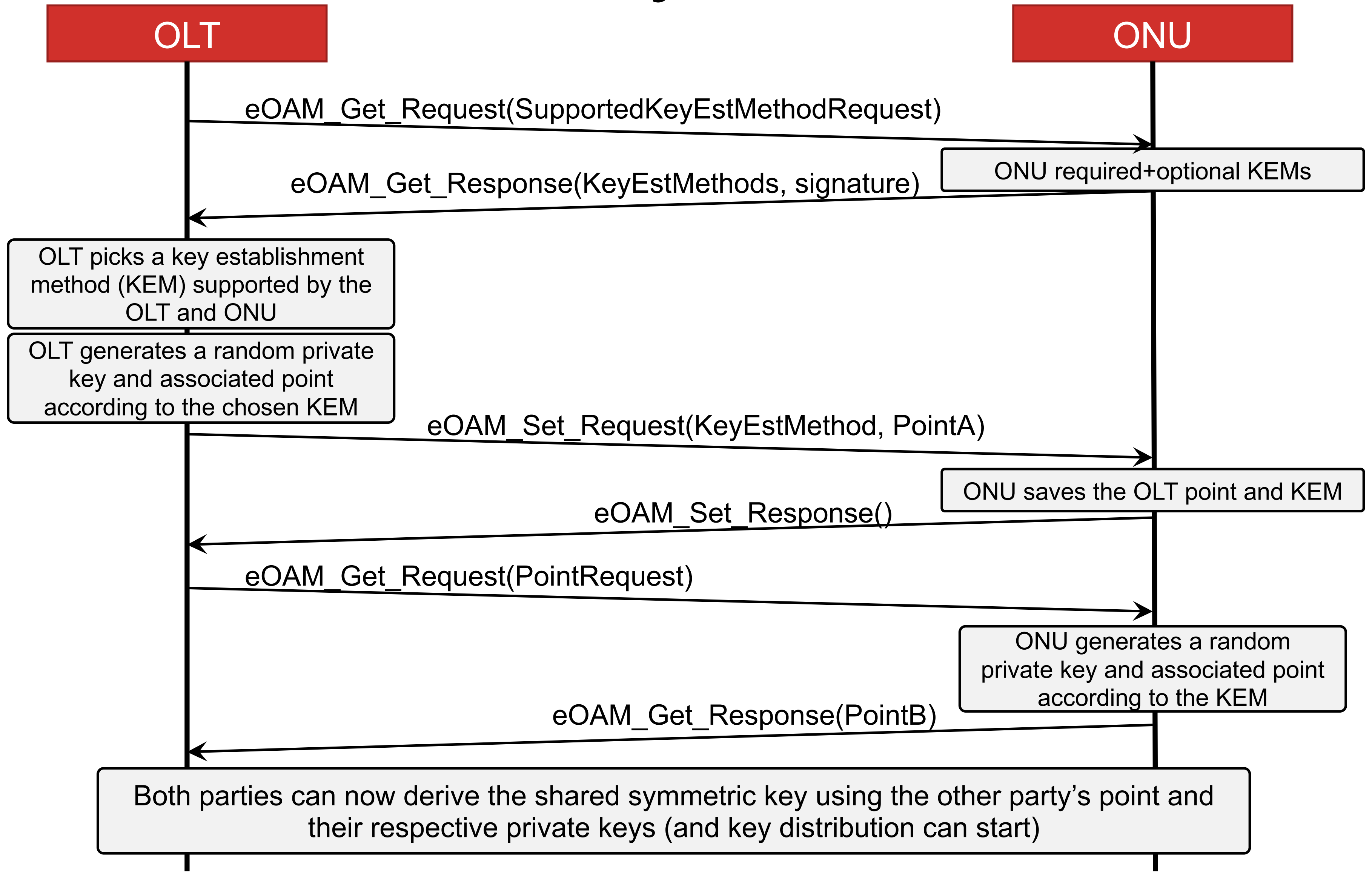
SIEPON Initial Key Establishment

- Approach: Ephemeral Diffie-Hellman (EDH) key exchange
  - Each party generates a random private key (*dA*) to determine a point in the same generator function *G*
  - The domain *G* is special. Knowing a point in the domain can't (practically) tell you how the party providing the point got there (and the key itself)
  - Parties exchange public key points (Q_A and Q_B) and, combined with their respective private key, each ends up at the same point (*Share*)
  - Any party observing the exchanged points can't determine the final point



From https://asecuritysite.com/encryption/

SIEPON Initial Key Establishment


OLT
ONU
eOAM_Get_Request(SupportedKeyEstMethodRequest)
ONU required+optional KEMs
eOAM_Get_Response(KeyEstMethods, signature)
OLT picks a key establishment method (KEM) supported by the OLT and ONU
OLT generates a random private key and associated point according to the chosen KEM
eOAM_Set_Request(KeyEstMethod, PointA)
ONU saves the OLT point and KEM
eOAM_Set_Response()
eOAM_Get_Request(PointRequest)
ONU generates a random private key and associated point according to the KEM
eOAM_Get_Response(PointB)
Both parties can now derive the shared symmetric key using the other party's point and their respective private keys (and key distribution can start)

SIEPON Initial Key Establishment

- Benefits of EDH initial key establishment:
  - Doesn’t require a shared secret or pre-existing trust relationship
  - Observer of OLT-to-ONU and/or ONU-to-OLT traffic cannot derive the initial session key
  - Authentication can be performed independently from initial key establishment
  - Provides Perfect Forward Secrecy (PFS)
    - Decoding a key in the future doesn’t provide access to data recorded in the past
  - Method is widely used and supported
    - Most TLS 1.2 (e.g. HTTPS) sessions utilize ECDH key establishment (elliptic curve Diffie-Hellman) – and all TLS 1.3 connections use ECDH
    - Supported by common crypto libraries
  - Does not require hardware support
    - Since this will only be performed at ONU initialization
    - And ECDH is both time- and space-efficient – well suited for embedded application

Details...

- Which EDH domains(s) to support?
  - TLS defines ~25 named elliptic curves and ways to support arbitrary unnamed curves
- What format to use for point exchange?
  - TLS defines 3 formats
- Which domains to require?
  - Want to pick one that's widely supported - an ECC curve
- Recommend:
  1. Allow for the expression of a few high-security ECC curves (subset of RFC3748)
  2. A single encoding for the point exchange (uncompressed)
  3. Pick a couple high-security widely-supported elliptic curves as required by the ONU (e.g. prime256v1, secp256k1)

Point here is to keep things simple while ensuring enough flexibility to enable migration to different EDH methods in the future if/when necessary

(without defining new PDUs/TLVs and semantics)

- Backup material

SIEPON Initial Key Establishment

- DHE semantics in SIEPON.4 PDUs:
  1. OLT issues Get_Request on ONU to with TLV to retrieve list of EDH key establishment methods supported by the ONU
     - One or more methods will be mandatory for the ONU (e.g. “ecc_prime256v1”)
     - Message will be signed by the ONU
  2. OLT picks a EDH key establishment method (e.g. “ecc_prime256v1”) and randomly generates its public/private key components (e.g. a point on the ecc_prime256v1 elliptic curve)
  3. OLT issues a Set_Request on ONU with TLV providing its EDH public key parameters
     - ONU stores the OLT’s public key parameters
     - Message will be signed by the OLT
  4. OLT issues a Get_Request on ONU to retrieve the ONUs EDH public key parameters
     - ONU randomly generates its public/private key components
     - ONU returns its EDH public key parameters
  5. ONU and OLT mutually derive the same shared ephemeral session key which will be used as the initial key for the encryption entity associated with the ONU
  6. Key updating/activation is performed as described in section 11.5