GMPLS Network Control Plane Enabling Quantum Encryption in End-to-End Services

Alejandro Aguado, **Víctor López**, Jesús Martínez-Mateo,
Momtchil Peev, Diego López and Vicente Martin

CCS Center for Computational Simulation
POLITÉCNICA
Telefonica
HUAWEI

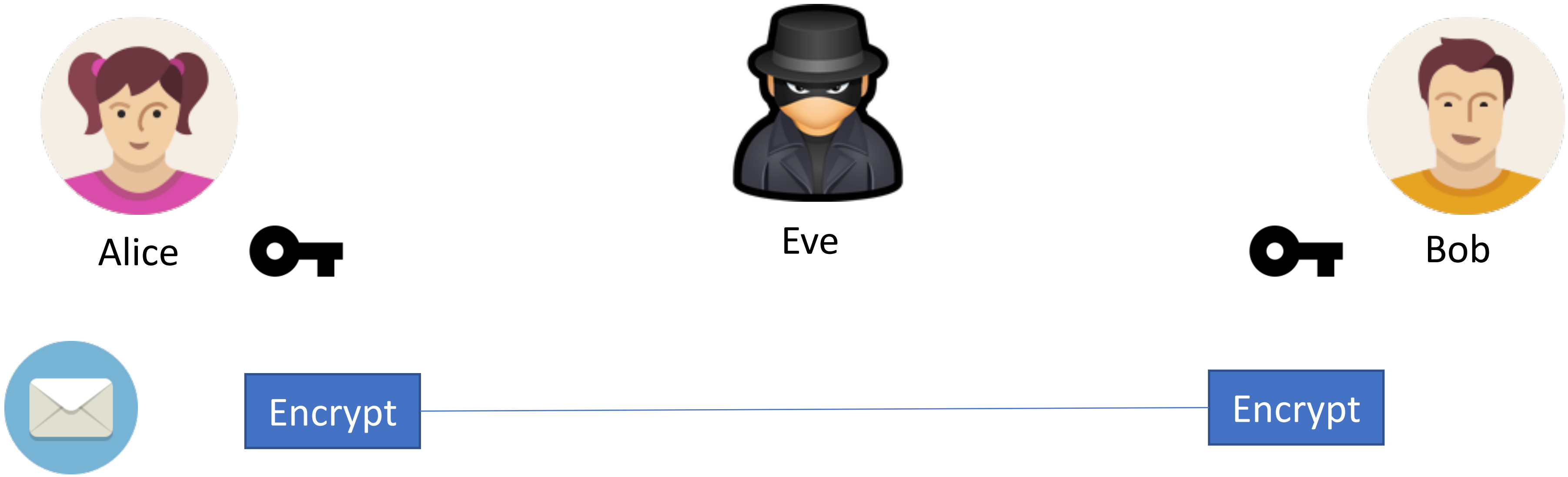
Outline

- Introduction
- Secure Channel Creation
- QKD node architecture
- PCE/GMPLS extensions to enable automatic provisioning
- Experimental validation
- Conclusions

Introduction

- **Quantum key distribution** (QKD) is a novel technology that can be seen as a **synchronized source of symmetric keys** in two separated domains that is immune to any algorithmic cryptanalysis.
- On the other hand, network services are **increasingly requesting more flexibility and network resources**.
- One of the biggest demands is to **increase the level of security** for the transmission between remote premises.
- In this work, we propose a **node architecture** and define **protocol requirements** in a GMPLS environment to provide **QKD-enhanced security in end-to-end services**.

Introduction



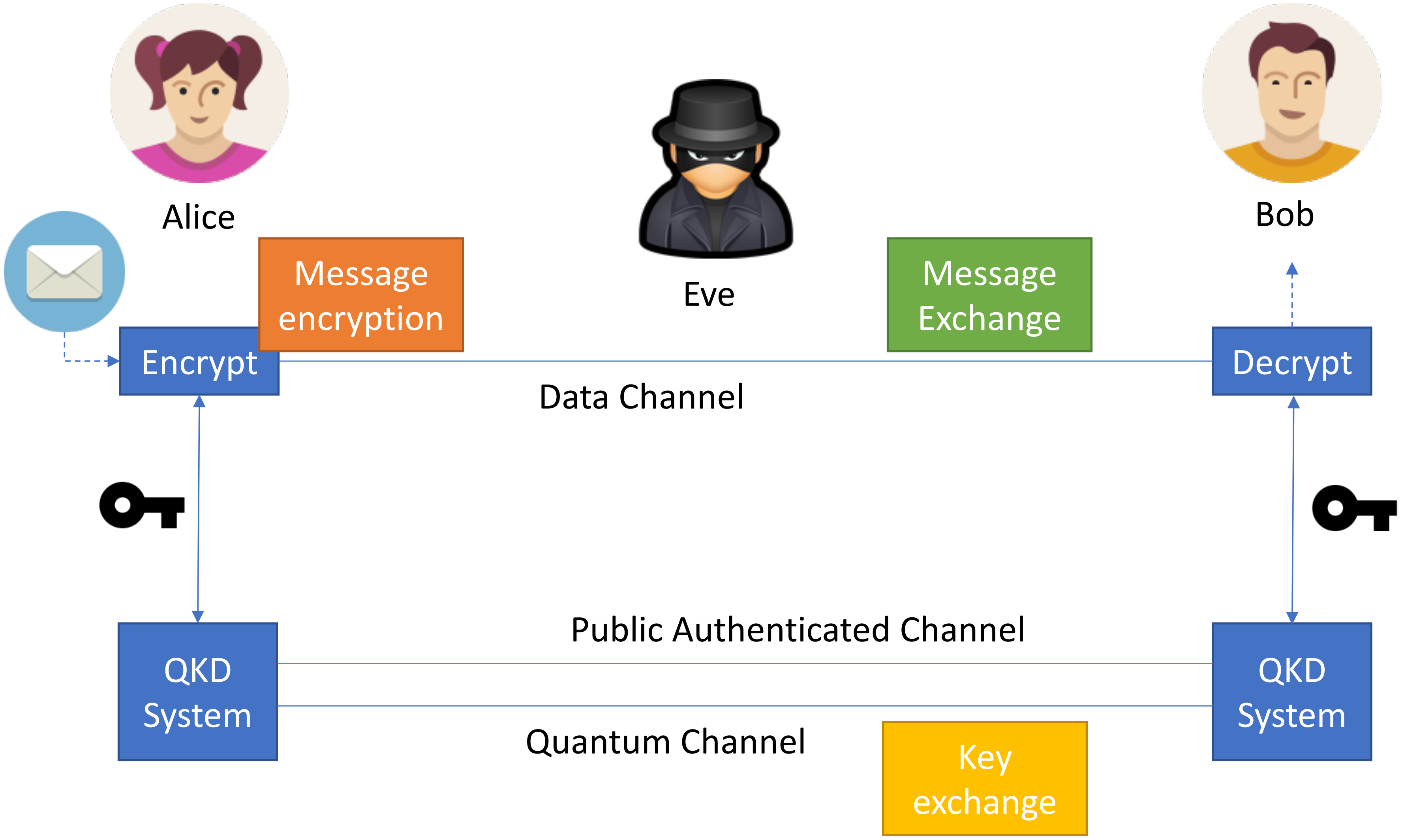
Key exchange

Channel Creation

Message encryption

Message Exchange

Introduction: Quantum Key Distribution



Ingredients:

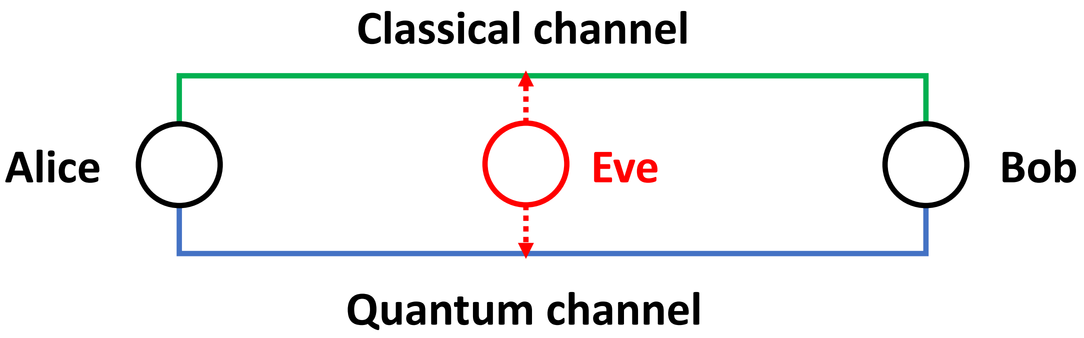
- Qubit transmitter (typically photons), Alice.
- Single qubit receivers, Bob.
- Quantum channel (capable of transmitting qubits from Alice to Bob, in our case fibre).
- Classical channel (public, but authenticated).

Main steps:

- Raw key exchange:
    - Qubit transmission
    - Sifting (basis reconciliation)
- Key post-processing:
    - Information reconciliation
    - Error verification
    - Privacy amplification

Introduction: Quantum Key Distribution

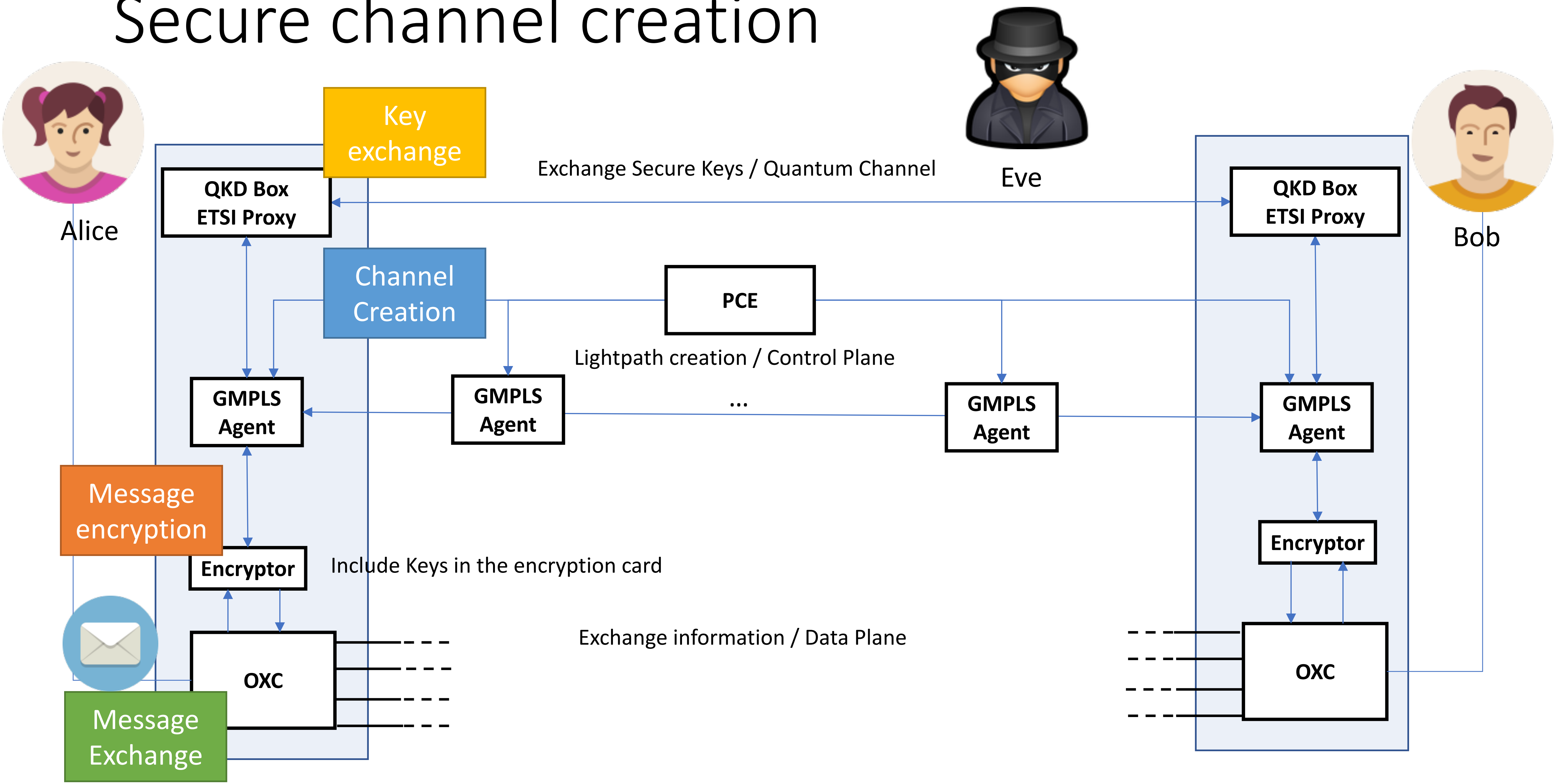
- **QKD** technology can be regarded as **two sources of synchronized random numbers** that are **separated physically.**
- A correct implementation will deliver keys of the **highest security**
- It can be **mathematically proven to be secure** (in principle, an information theoretic secure (ITS) primitive)



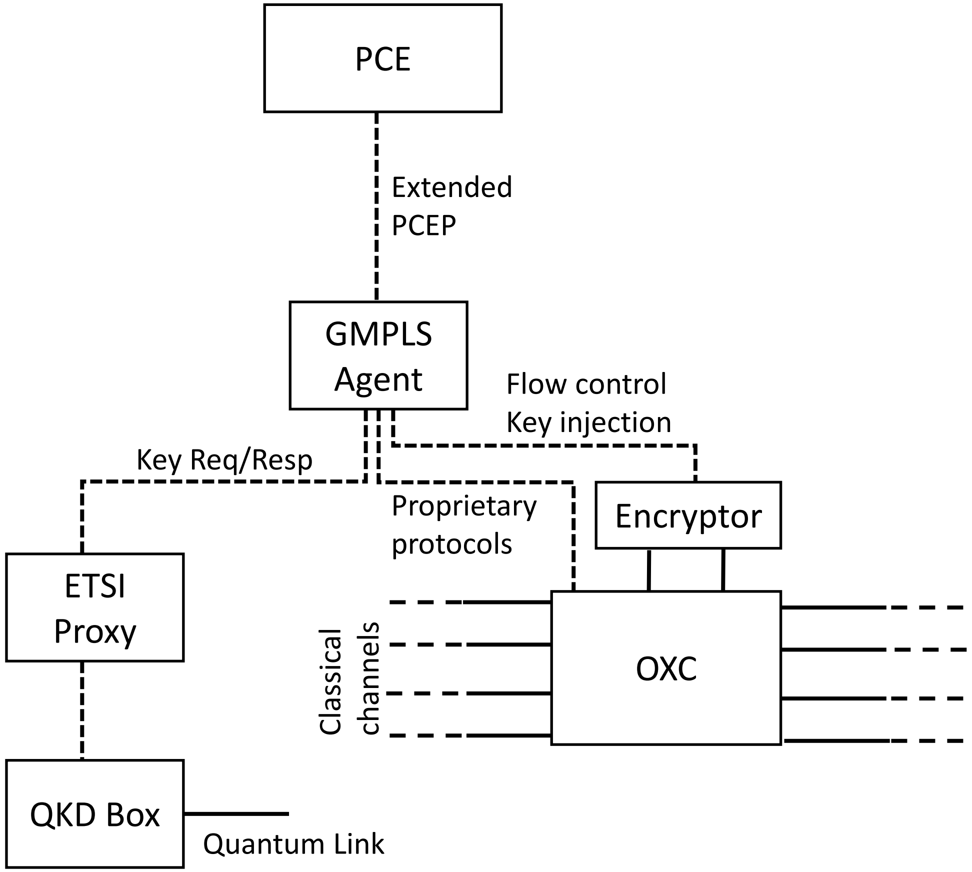
LIMITATIONS

- QKD has some limitations that do not affect the **conventional cryptosystems, usually based on computational complexity.**
- Any kind of **amplifiers or active components** that can modify the state of these signals **must be bypassed**.
- This sets a **limit to the maximum distance** (or absorptions) that a QKD protocol can tolerate, well suited to be used within a metropolitan area or with links of **up to 150 km**

Secure channel creation



Example of QKD-enabled network node architecture



Desired capabilities:

- Access to QKD-generated keys.
- Encryption in upstream services (Data encryptor, security module, etc.).
- Switching/Routing.
- Control plane interface enabling automation

Definition of requirements in terms of parameters

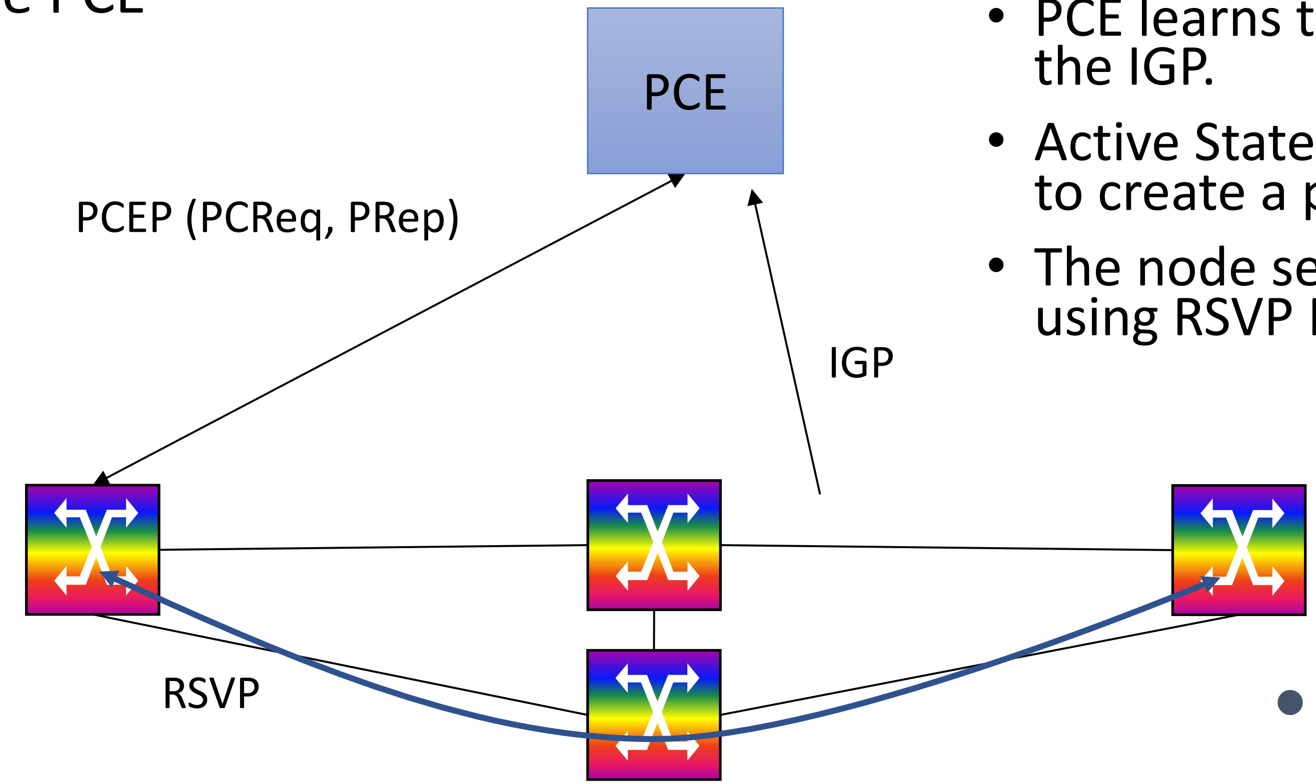
- Parameters required to be exchanged (point-to-point encryption):
  - **Session ID (key_handle)**: Initially set as 0, session ID gets the value of the first Key handle extracted by the source agent in the initial setup. The source agent will be in charge of updates (future work).
  - **Key length**: Length of the key to be used for the encryption.
  - **Destination**: It defines the other peer (encryptor/decryptor) to synchronise with. Currently defined by an IP address.
  - **Encryption Layer**: Layer where encryption is performed.
  - **Refresh type and value**: Type of refresh to be done for a key (time/traffic/etc) and the value to be considered as a threshold.
  - **Algorithm**: Encryption algorithm to be used.

Distributed GMPLS Control

- Majority of the commercial deployments of optical core and transport networks are based on GMPLS.
- GMPLS was standardized by IETF in CCAMP WG
- Fundamental protocols:
  - RSVP-TE : responsible of setting up end-to-end quality-enabled connections
  - OSPF-TE: dissemination of the topology and traffic engineering (TE) information, enabling routing
  - LMP (Link Management Protocol): is responsible of links management

Path Computation Element

- GMPLS is complemented with a logically centralized element, the PCE

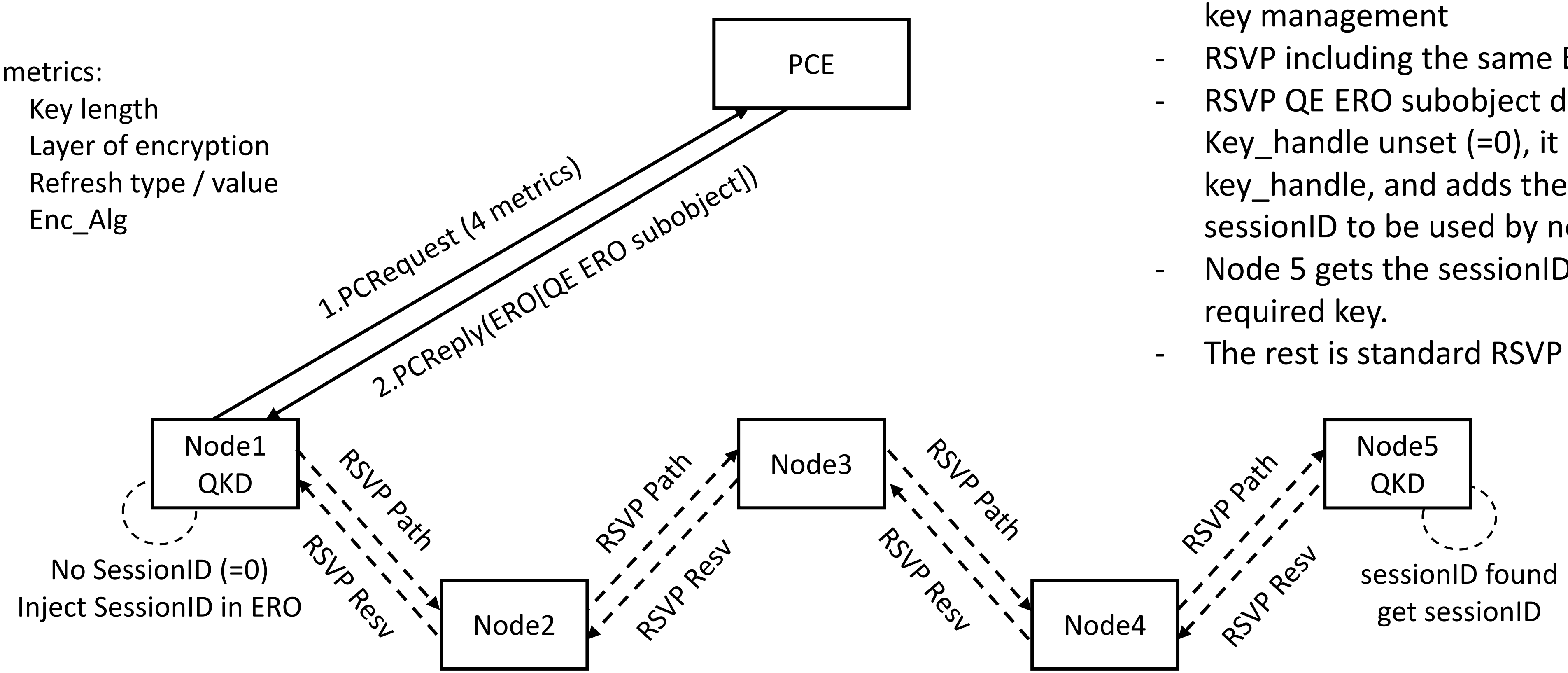


- PCE learns the TE DB listening the IGP.
- Active Stateful PCE can request to create a path using PCInitiate.
- The node set-up the connection using RSVP Path, Resv.

- Telefonica Netphony release open source PCE implementation and GMPLS control plane.

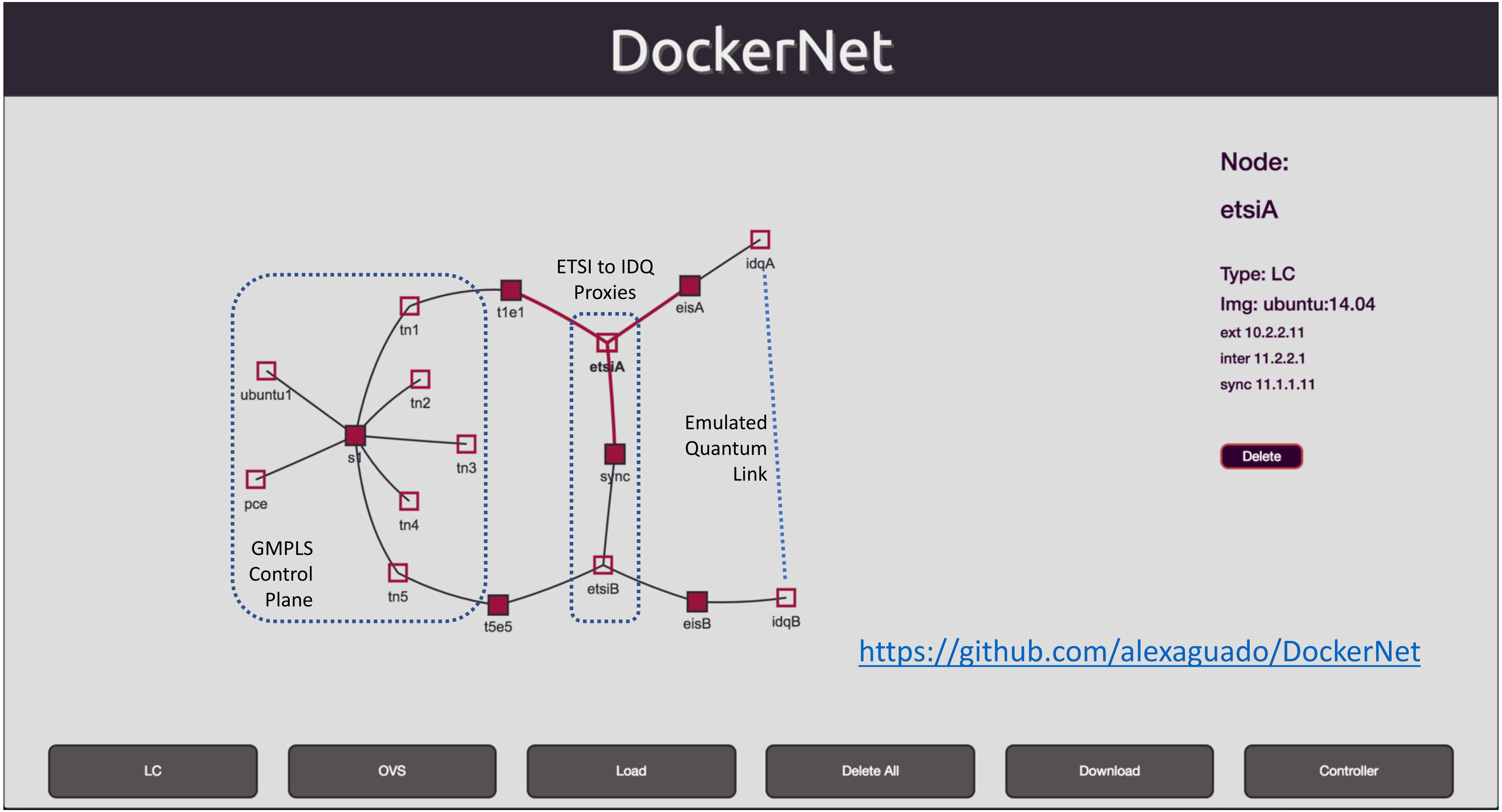
GMPLS+PCE Architecture
Proposed workflow: Case “Node starts”

4 metrics:
- Key length
- Layer of encryption
- Refresh type / value
- Enc_Alg



GMPLS case:
- PCRequest including metric for inline encryption.
- PCReply including new ERO subobjects for key management
- RSVP including the same ERO
- RSVP QE ERO subobject detected by node 1. Key_handle unset (=0), it gets a new key and key_handle, and adds the key_handle as sessionID to be used by node5
- Node 5 gets the sessionID and extracts the required key.
- The rest is standard RSVP

Experimental validation



https://github.com/alexaguado/DockerNet

Experimental validation
OSPF for Quantum encryption capabilities

```
▼ Open Shortest Path First
  ▶ OSPF Header
  ▼ LS Update Packet
      Number of LSAs: 1
    ▼ Opaque LSA, Area-local scope
        .000 0000 0000 0000 = LS Age (seconds): 0
        0... .... .... .... = Do Not Age Flag: 0
      ▶ Options: 0x00
        LS Type: Opaque LSA, Area-local scope (10)
        Link State ID Opaque Type: Optional Router Capabilities Opaque RI LSA (4)
        MBZ: 0x0000
        Advertising Router: 10.1.1.5 (10.1.1.5)
        Sequence Number: 0x00000000
        Checksum: 0x0000
        Length: 28
      ▼ Opaque Router Information LSA
        ▼ RI TLV
            TLV Type: Router Informational Capabilities TLV (1)
            TLV Length: 4
          ▼ RI Options: 0x12 ((TES) Traffic Engineering)
              0... .... = (GRC) Graceful Restart: Not capable
              .0.. .... = (GRH) Graceful Restart Helper: Disabled
              ..0. .... = Stub Router Support: No
              ...1 .... = (TES) Traffic Engineering: Supported
              .... 0... = (P2PLAN) Point-to-point over LAN: Not capable
              .... .0.. = (ETE) Experimental TE: Not capable
```

Quantum Encryption support (bit 7): capable

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            LS age             |     Options   |  9, 10, or 11 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       4       |                       0                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Advertising Router                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     LS sequence number                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         LS checksum           |             length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+-                            TLVs                             -+
|                             ...                               |
```

OSPFv2 Router Information Opaque LSA

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              Type             |             Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Informational Capabilities                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Informational Capabilities TLV

Experimental validation PCEP

```
10.1.1.1     10.1.1.200   PCEP   160 Path Computation Request (PCReq)
10.1.1.200   10.1.1.1     PCEP   268 Path Computation Reply (PCRep)
10.1.1.1     10.1.1.200   PCEP   272 Path Computation LSP State Report (PCRpt)
```

```
▼ Path Computation Element communication Protocol
  ▶ Path Computation Request (PCReq) Header
  ▶ RP object
  ▶ END-POINT object
  ▶ BANDWIDTH object
  ▶ METRIC object
  ▶ METRIC object
  ▶ METRIC object
  ▶ OBJECTIVE FUNCTION object (OF)
```

```
▼ METRIC object
    Object Class: METRIC OBJECT (6)
    0001 .... = Object Type: 1
  ▶ Flags
    Object Length: 12
    Reserved: 0
  ▶ Flags: 0x80
    Type: Unknown (255)
    Metric Value: 32
```

```
▼ METRIC object
    Object Class: METRIC OBJECT (6)
    0001 .... = Object Type: 1
  ▶ Flags
    Object Length: 12
    Reserved: 0
  ▶ Flags: 0x80
    Type: Unknown (252)
    Metric Value: 1000
```

```
▼ METRIC object
    Object Class: METRIC OBJECT (6)
    0001 .... = Object Type: 1
  ▶ Flags
    Object Length: 12
    Reserved: 0
  ▶ Flags: 0x80
    Type: Unknown (254)
    Metric Value: 2
```

```
▼ METRIC object
    Object Class: METRIC OBJECT (6)
    0001 .... = Object Type: 1
  ▶ Flags
    Object Length: 12
    Reserved: 0
  ▶ Flags: 0x80
    Type: Unknown (253)
    Metric Value: 10
```

```
▼ Path Computation Element communication Protocol
  ▶ Path Computation Reply (PCRep) Header
  ▶ RP object
  ▶ BANDWIDTH object
  ▶ Unknown object
  ▼ EXPLICIT ROUTE object (ERO)
      Object Class: EXPLICIT ROUTE OBJECT (ERO) (7)
      0001 .... = Object Type: 1
    ▶ Flags
      Object Length: 256
    ▶ SUBOBJECT: Unnumbered Interface ID: 10.1.1.1:1
    ▶ Non defined subobject (103)
    ▶ SUBOBJECT: Label Control
    ▶ SUBOBJECT: Unnumbered Interface ID: 10.1.1.2:1
    ▶ SUBOBJECT: Label Control
    ▶ SUBOBJECT: Unnumbered Interface ID: 10.1.1.3:3
    ▶ SUBOBJECT: Label Control
    ▶ SUBOBJECT: Unnumbered Interface ID: 10.1.1.4:3
    ▶ SUBOBJECT: Label Control
    ▶ SUBOBJECT: IPv4 Prefix: 10.1.1.5/32
    ▶ Non defined subobject (103)
  ▶ BANDWIDTH object
```

New QE ERO subobject

Experimental validation RSVP (signalling)

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|L|     Type      |     Length    |  IDQ Session ID  (64 bytes)  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              ...  (64 bytes)                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  IDQ Session ID  (64 bytes)   |    KEY LEN     |    Ref Type   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Refresh value  (2 bytes)   |      Enc_Algo (2 bytes)        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

10.1.1.1	10.1.1.2	RSVP	PATH Message.
10.1.1.2	10.1.1.3	RSVP	PATH Message.
10.1.1.3	10.1.1.4	RSVP	PATH Message.
10.1.1.4	10.1.1.5	RSVP	PATH Message.
10.1.1.5	10.1.1.4	RSVP	RESV Message.
10.1.1.4	10.1.1.3	RSVP	RESV Message.
10.1.1.3	10.1.1.2	RSVP	RESV Message.
10.1.1.2	10.1.1.1	RSVP	RESV Message.

Node 4 QE ERO subobject.
(before node 2)
Type: 0x67
Value: ”00..00” (64 bytes)
KeyLenght: 32
Enc_layer: 2
RefType: 0xfd
RefValue: 60
Alg: 10 (TBD)

Node 4 QE ERO subobject.
(before node 2)
Type: 0x67
Value: “4a0e…052f” (64 bytes)
KeyLenght: 32
Enc_layer: 2
RefType: 0xfd
RefValue: 60
Alg: 10 (TBD)

```
0120  20 00 67 4a 00 00 00 00  00 00 00 00 00 00 00 00
0130  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0140  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0150  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0160  00 00 00 00 00 20 02 fc  03 e8 00 0a 05 30 00 10
```

```
00f0  00 00 01 08 0a 01 01 05  20 00 67 4a 4a 0e 75 e8
0100  03 d7 f6 9e 9a 29 a1 0d  1c 7b 31 10 ac c3 95 98
0110  b4 78 9f 4f 0d 0e c1 40  fb ca 46 1d 6c a5 d2 a8
0120  a8 cc f0 d4 95 71 76 7d  31 b6 e0 69 4e a0 10 a0
0130  95 89 98 eb df 7d 35 85  e3 e6 05 2f 00 20 02 fc
0140  ff e8 00 0a 00 08 13 01  00 00 00 01 00 0c 0b 07
```

Conclusions

- We propose a node architecture and define protocol requirements in a GMPLS environment to provide QKD-enhanced security in end-to-end services.
- This is the first work to propose, implement and validate extensions in a PCE/GMPLS architecture to use this technology.
- The work is done with Open Source tools using Netphony and DockerNet.
- As future work, the authors will explore this approach in OpenFlow or Netconf.

THANK YOU!!!

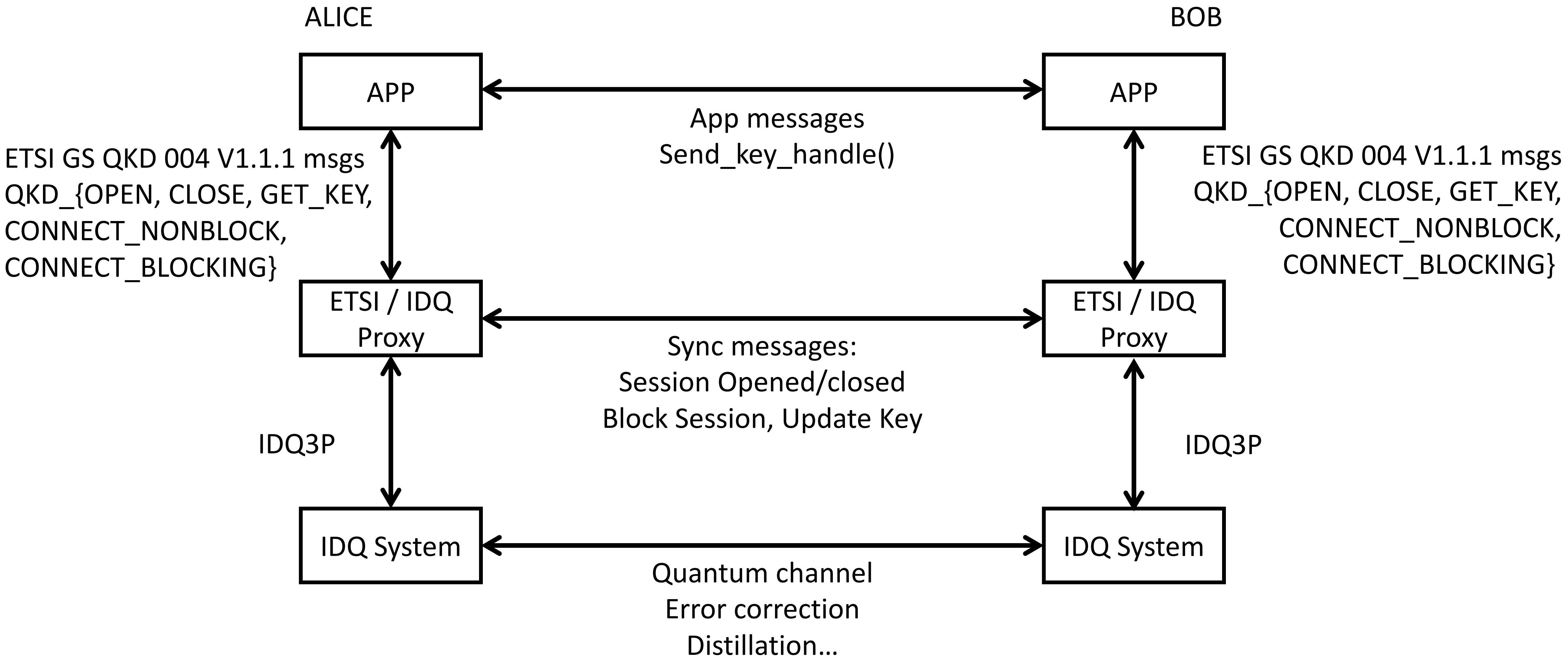
Alejandro Aguado, **Víctor López**, Jesús Martínez-Mateo,
Momtchil Peev, Diego López and Vicente Martin

Appendix A
ETSI GS QKD 004 V1.1.1
for remote apps and IDQ3P

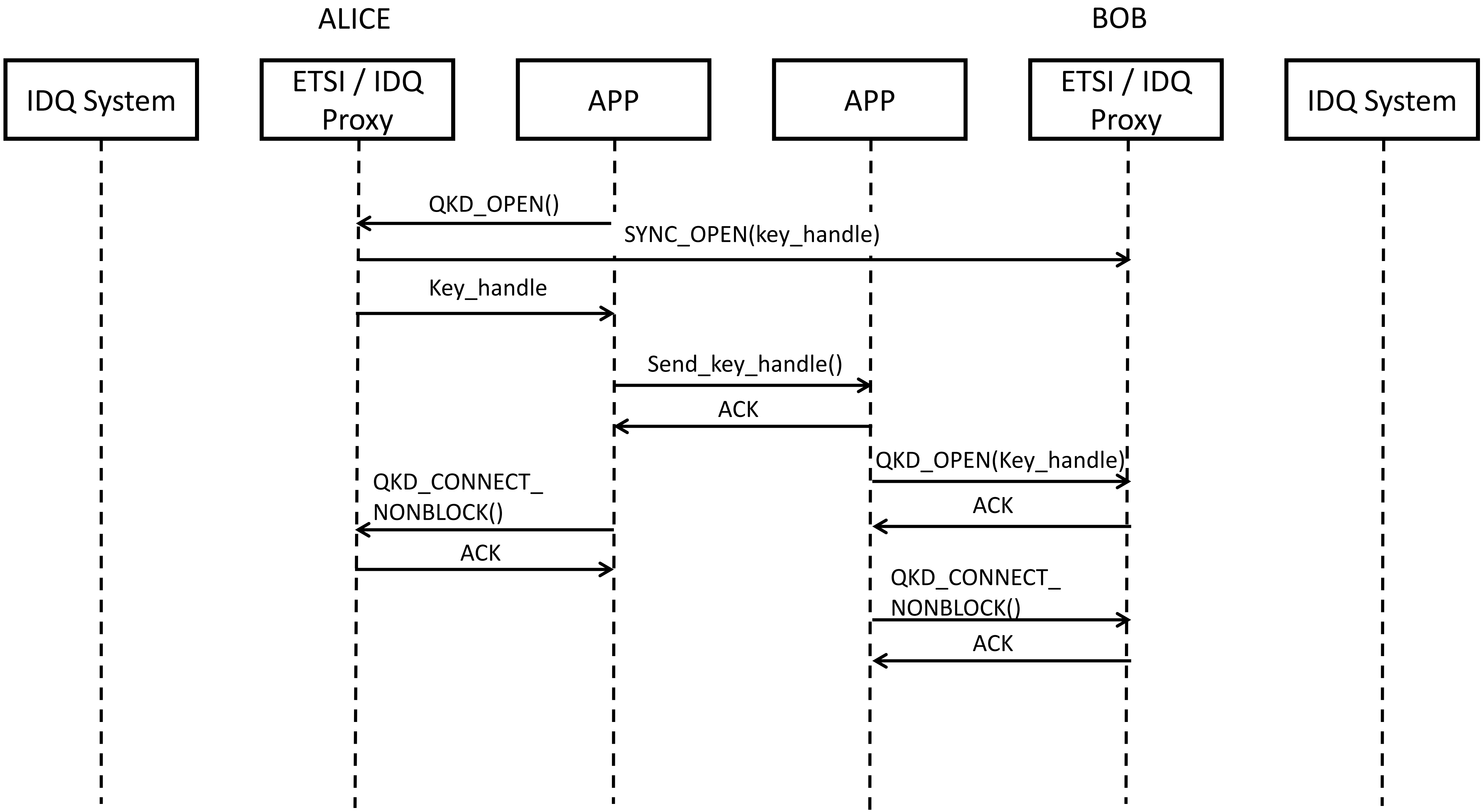
ETSI IDQ Proxy

- ETSI GS QKD 004 V1.1.1 defines an API to be used by applications which are running within the same server as the Key Manager.
- In order to justify the use of this standard, we have developed a proxy that implements ETSI GS QKD 004 V1.1.1-based messages to communicate with external applications
- These messages are mapped to IDQ3P requests.
- Additional Sync messages have been implementes as well.
- This interface allows to use a single identifier (key_handle) that can be used to extract multiple keys.

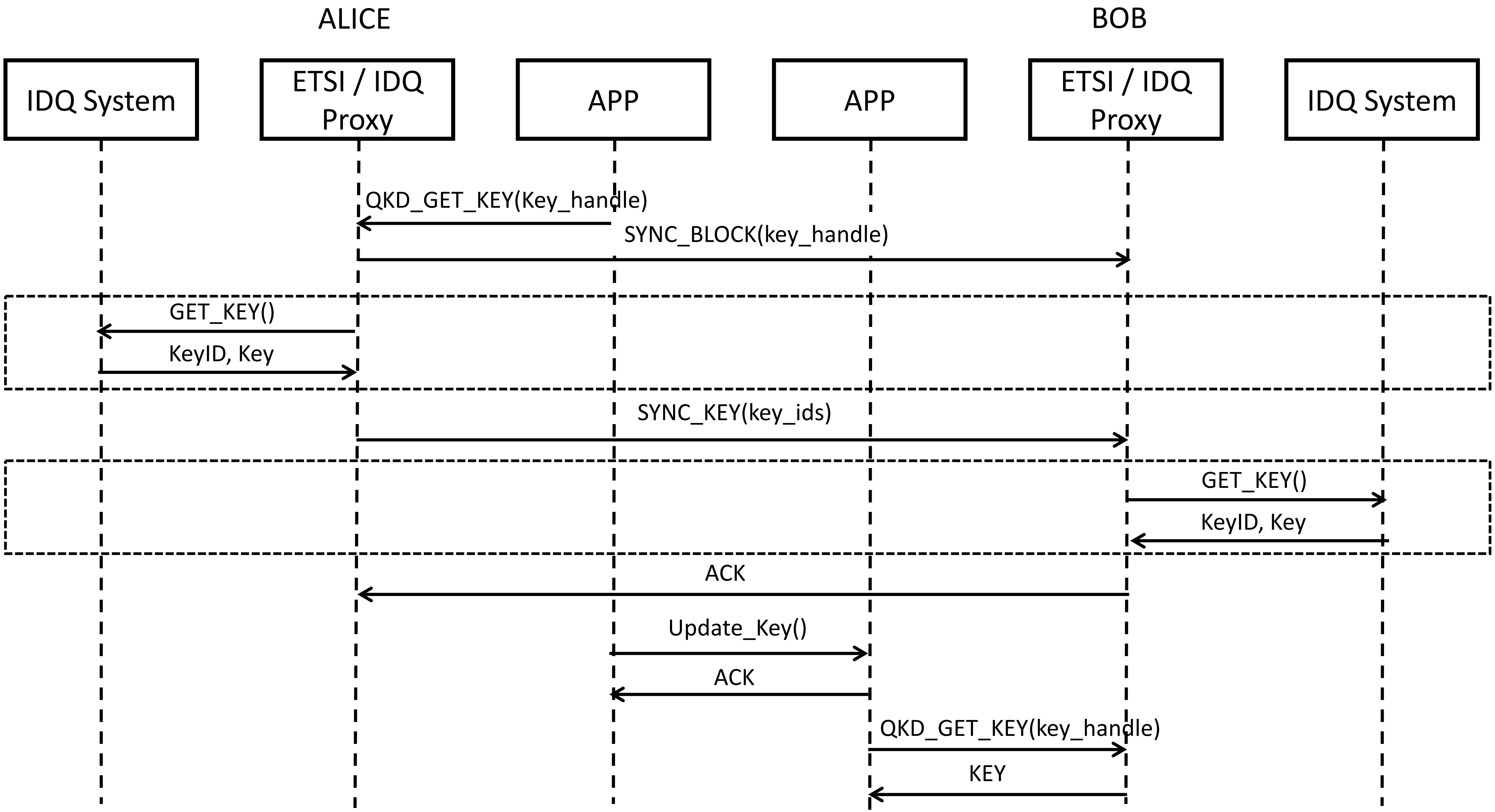
Modules / Messages



Example OPEN & CONNECT



Example GET_KEY



Example CLOSE