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

A. Aguado[1], V. Lopez[2], J. Martinez-Mateo[1], M. Peev[3], D. Lopez[2] and V. Martin[1]
[1]Center for Computational Simulation, Universidad Politécnica de Madrid 28660 Madrid, Spain
[2]Telefonica Investigacion y Desarrollo, Ronda de la Comunicacion s/n 28050 Madrid. Spain
[3]Huawei Technologies Duesseldorf GmbH, Riesstrasse 25, 80992 Munchen. Germany
(email: a.aguadom@fi.upm.es, victor.lopezalvarez@telefonica.com, jmartinez@fi.upm.es, momtchil.peev@huawei.com, diego.r.lopez@telefonica.com, vicente@fi.upm.es)

***Abstract*— 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. This technology makes impossible to copy the quantum states exchanged between two endpoints. Therefore, if implemented properly, QKD generates keys of the highest security based on the fundamental laws of quantum physics. No algorithmic advance would force a change of technology, as opposed to current public key cryptographic protocols, that rely on the complexity of certain mathematical problems. These protocols are at risk due to the advances in quantum computing and should be changed. On the other hand, network services are increasingly requesting more flexibility and network resources. One of the most desired capabilities is having higher level of security for the transmission between remote premises. In this work, we propose a node architecture to provide QKD-enhanced security in end-to-end (E2E) services and analyze the control plane requirements in order to provide such services in transport networks. This work defines and demonstrates for the first time extensions for generalized multi-protocol label switching (GMPLS) networks. Results show how these new services could be integrated in existing operators' control plane architectures.**

Keywords*— *Network Management, Quantum Key Distribution, Software Defined Networking, Generalized Multi-Protocol Label Switching, Path Computation Element.

I. INTRODUCTION

NETWORK services are nowadays evolving on a daily basis. They are moving from a traditional static approach, to a more flexible and dynamic one. Traditional services usually require several days (or even weeks) to be established, while new applications and services change their requirements much faster. This evolution, aiming to cope with this dynamicity, is possible thanks to a software-based process. This novel network paradigm, called software defined networking (SDN), permits decoupling the control plane (traditionally running in the core of a network device) from the data plane, managing network services from centralized entities (network controllers).

Security is an increasing concern in communications networks, as critical information travels across an entire infrastructure. However, network security has not been the result of a systematic effort but more as a series of ad-hoc solutions. During some time, even arguments that have been demonstrated to be false —like the intrinsic security of optical fibers— were used as a reason to delay or rule out the implementation of security mechanisms. Today, networks are more complex and, especially with SDN, much more configurable. The security risks are correspondingly larger, therefore, security in network infrastructures must be enhanced.

QKD technology can be regarded as two sources of synchronized random numbers that are separated in space, with the added property that there is a security demonstration that allows to upper bound the maximum information that is leaked out of these two sources. The security of the symmetric keys produced by systems built around this technology is, by principle, immune to any algorithmic cryptanalysis. QKD can be seen as a new opportunity for operators and infrastructure providers as it can enable the provision of new, high security encryption in end-to-end (E2E) services. This work proposes for the first time a new QKD enhanced network device providing encryption for E2E services, analyzes its requirements from the control plane perspective and shows how these nodes can be easily integrated in a GMPLS controlled infrastructure. We work under the assumption of having an underlying network of connected QKD devices in place. Our goal is to develop higher level protocols that allow for E2E QKD-key based encryption services. To the best of the author's knowledge, the only related work with SDN and security for optical networks is [1], where work on the North Bound Interface (NBI) exposing security parameters to the applications is presented. This work is complementary since it allows a GMPLS control plane below the network controller, used to synchronize the QKD-generated keys.

The paper is organized as follows: A brief introduction to QKD is presented in Section II, the basic structure of a "quantum encryption" node and its integration in a QKD network are discussed in Section III. Section IV describes the control plane architecture and network protocols considered for this work. In Section V the workflow in an E2E GMPLS architecture is presented. Section VI elaborates about the emulation platform and the implementation of the tests. Finally, in Section VII, results of our emulation are presented and conclusions are summarized in Section VIII.

II. QUANTUM KEY DISTRIBUTION

Quantum key distribution (QKD) [2] is a means to add an unbreakable physical layer to an optical network. Unbreakable is used here in the sense that it can be mathematically proven

This work was partially supported by the Spanish Ministry of Economy and Competitiveness, MINECO under grant CVQuCo, TEC2015-70406-R and by ACINO European H2020 project, Grant Number 645127, http://www.acino.eu.

to be secure, i.e. it is, in principle, an information theoretic secure (ITS) primitive. Thus, a correct implementation can deliver keys of the highest security. However, QKD has some limitations that do not affect the conventional cryptosystems, whose security is based on algorithmic complexity assumptions.

The same no-cloning property that the laws of physics imposes on quantum signals and that confers the ITS security to QKD protocols, means that the signals cannot be amplified. Hence, any kind of amplifiers or active components that can modify the state of these signals must be bypassed, thus setting a limit to the maximum distance (or absorptions) that a QKD protocol can tolerate.

When transmitting a signal through a not completely transparent medium, it suffers an exponential attenuation. In the case of an optical fiber and using the most transparent window (1550 nm) this typically means a loss of about 0.2 dB/km. If the signal must pass through some other passive optical components, these will add additional losses, another 0.2 dB per connector, 3 dB in a 1:2 splitter, etc. Modern QKD devices can tolerate up to approximately 30 dB of losses, which means that they are well suited to be used within a metropolitan area or with links of up to 150 km. There have been demonstrations going well beyond these limits, but this is unrealistic either because of extremely low key rates or high insecurity, since from a small amount of signals a high security key cannot be extracted. In principle, so-called quantum repeaters can be built, but the main component to be developed, quantum memory, is still not available. These devices will eventually avoid any absorptions/distance related problems, but is a technology still years in the future. Given the delicate nature of the quantum signals, the most reliable way of transmission is to use a dedicated dark fiber as the quantum channel. This is not a must, however, and previous test-beds [3] have shown how to use a fiber shared by quantum and classical channels and the tradeoffs involved.

Nonetheless, when considering real networks, the distance limit has a relative importance as long as it is considered that the task here is to connect the different security perimeters between the different security domains in the network. Operators define a security perimeter and assume that their nodes are secured places. Distances between secure nodes [4] are typically well within the QKD distance limits. Also, in absence of quantum repeaters, this limitation can be ameliorated by forwarding the keys using the trusted nodes approach [5, 6].

Beyond the quantum channel, QKD requires a classical channel. It is also assumed public, but it must be authentic. The origin and integrity of the data must be preserved. On installation, a first token is used to initiate the authentication of the classical channel. After this phase, the same keys that are created at both ends can be used to insure the continued ITS authentication of the channel. Then, the continuous supply of symmetric keys that QKD systems provide, can be used to create authentication chains for the different services in the network. Since they are symmetric keys, fast encryption algorithms can also provide very high throughput capabilities down to the packet level.

III. Quantum Encryption Node

In a QKD enhanced network context, with E2E encryption services, a network device with quantum encryption (QE) capabilities should be able to communicate with the networks' control plane, beyond a standard interface and through an open one, certain additional characteristics. Note that, with QE we mean an encryption scheme that uses the QKD generated keys. More specifically, these devices should generate (or have access to) a pool of keys to perform symmetric encryption, expose unique IDs that identify quantum keys or sessions to the network controllers and/or applications (northbound interface) and perform switching and inline encryption.

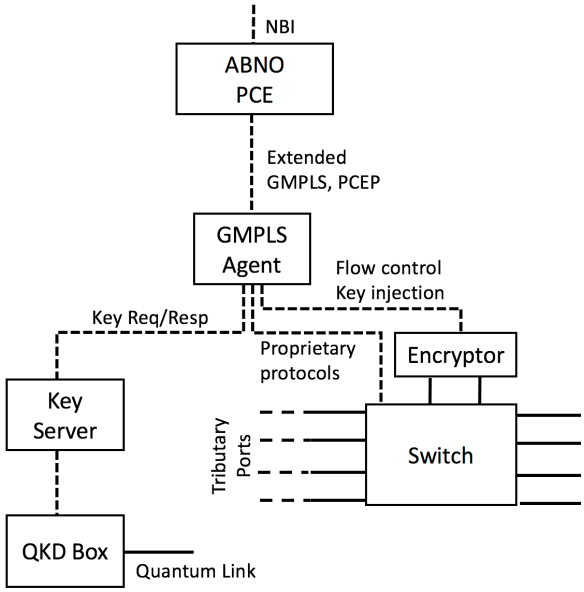

Fig 1 Example of a GMPLS-enabled node with QE capabilities.

Fig. 1 shows an example of how a quantum enabled network device could look like. A QE-capable device should have access to a source of quantum keys. This can be implemented in many ways, as there is still not a standard protocol for QKD devices to expose their keys. We have used the ETSI GS QKD 004 V1.1.1. API [7] to implement such protocol. An optical device will allow the agent to choose whether certain traffic should be encrypted or just forwarded to the network. It will also allow to easily send the packets to a traffic encryption device (e.g. a hardware security module). These are commercial products ready in the market, authors in [1] uses this kind of equipment. This set of hardware devices and interfaces must be orchestrated from a central entity within the domain, acting as a controller for the configuration. This entity (agent) exposes information of a single node, abstracting the multiple devices as one, and allows to be configured by centralized systems using standard protocols (e.g. GMPLS, OpenFlow, NETCONF). However, bringing quantum encryption awareness and the capability of providing inline encryption into a logically centralized control plane will require to extend these protocols, since this will require new information to be transmitted to perform routing and capabilities dissemination.

IV. Control Plane Architecture

The software defined networking (SDN) paradigm, based on network programmability, decouples the network's data

plane from the control plane. This allows operators to centrally manage and dynamically setup, tear down and optimize customer services in their infrastructures, all done via standard protocols and interfaces. Different architectures and protocols have been created to allow this process of decoupling the forwarding decisions from the devices into a centralized architecture. The path computation element (PCE) architecture was presented in [8]. This architecture gathers a collection of algorithms that are computationally difficult and that may require special computational components and synchronization with other domains to calculate E2E paths over multi-layer and multi-domain networks. It can act in passive (receiving path requests and providing responses) or active mode (communicating with network devices to configure the services). It can work as a standalone module, cooperate with other PCE (east-west or parent-child mode [9]), or be a part of a more complex architecture, such as the application-based network operations (ABNO) [10,11].

In order to interact with the network devices, the PCE uses multiple protocols. GMPLS is a family of protocols that extends MPLS to cover time, space and wavelength division multiplexing (TDM, SDM and WDM respectively) switching technologies. It allows devices to share their traffic engineering (TE) information (like reachability, link state, router information, etc.) using protocols such as open shortest path first protocol with TE extensions (OSPF-TE); signal specific configurations across the network via resource reservation protocol with TE extensions (RSVP-TE); or communicate and receive responses or instructions from the PCE via PCE communication protocol (PCEP). In order to interact with the network devices, the PCE uses two protocols PCEP (for LSP management) and OSPF-TE (for Traffic Engineering information). This set of protocols allows the PCE to orchestrate an entire network centrally, handling network services and optimizing resources.

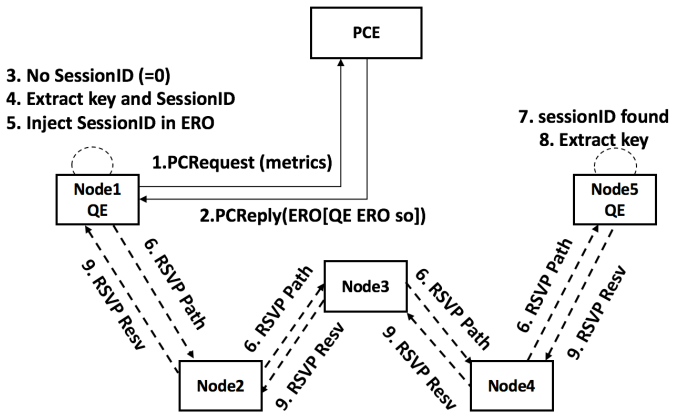

Fig 2 Workflow of the end-to-end quantum encryption in a GMPLS network scenario.

V. Proposed Workflow and Extensions

Calculating and establishing E2E services in a GMPLS-enabled network usually require several operations within the network's control plane. These operations can be divided into three subgroups: information dissemination, path calculation-configuration and signaling. For this, different components communicate among each other via standard protocols, such as OSPF-TE, BGP-LS, PCEP, RSVP, etc. Nodes within a network expose their capabilities and reachability among themselves and externally. A central management entity (in our case, a PCE) calculates paths (and configure them if in active mode), and the network nodes forward the configuration (explicit route object, ERO) from the first to the last node in the path.

Following the same scheme, our new QKD-based encrypted services will require a subset of these operations and protocols to be extended (or partially redefined). We have adjusted the abovementioned subgroups to cover the service workflow (see Fig. 2) in a GMPLS-enabled scenario, exposed below as quantum encryption capabilities dissemination (no reachability information has been extended), path computation and signaling (key synchronization).

A. Quantum Encryption Capabilities Dissemination

Network nodes connecting to remote control plane entities must expose information to be controlled and optimized in the best way. This information includes, for example, number of ports, switching/routing capabilities, supported bandwidth, statistics, traffic engineering capabilities, etc. In this way, the control plane central entity (controller/PCE), can build a graph of the existing network and compute and optimize network services to better use the available resources. Encryption services (if centrally managed by a controller/PCE) require at least a minimum information to be exposed from the network devices that can perform such process. This information should be gathered by the PCE, stored on its TE database (TED), and then used when these services are computed.

The proposed extension to disseminate QE capabilities is based on the RFC7770 [12], which defines OSPF extensions for optional router capabilities. This extension, implemented for OSPFv2, uses the router information (RI) opaque link state advertisement (LSA) within an OSPF update message. The opaque LSA contains a 4-bytes-long informational capabilities TLV to expose, for example, whether a router is graceful restart capable/helper, has TE support, stub router support, etc. Following the same order than the RFC, we choose the first unused bit (number 6) to expose whether a router is quantum encryption (QE) capable or not. The PCE, under reception of this bit, stores the capability on its TED.

B. Path Computation

The main purpose of the PCEP is to communicate a PCE with a path computation client (PCC) or multiple PCEs. It is composed by multiple messages defined to request path calculation and its reply (PCRequest, PCReply) and path instantiation in active mode (PCInitiate, PCReport), among others to open, close and keep the PCEP session. In addition, the initiation of the service deployment process in the network differs depending on the mode of the PCE (active/passive, stateful/stateless):

- A passive PCE receives PCRequests to calculate a path, mainly from the PCC within a network device, and responds PCReply messages including the ERO, which is handled by the device.

- An active PCE can receive both PCRequest or PCInitiate messages to calculate and/or instantiate the connectivity. It can directly communicate with the network devices via PCInitiate message to configure the path.

Both procedures are suitable for our use case. Therefore, since most of the extensions directly affect the PCRequest message and the ERO inside a PCEP message, we have chosen the first one (passive PCE, with a source node asking for a E2E QE service) to expose our work in a clearer way.

When a QE E2E service is required, a set of parameters must be transmitted to the PCE in order to identify the service and to perform the required computation. The set of parameters we have utilized to identify the service includes:

- Session ID: ID to synchronize the keys in both ends of the service. Its length is 64 bytes, following the standard defined in [7] (key_handle).
- Destination: identifier of the destination endpoint of the QE service. In our implementation, an IP(v4) address.
- Key length: Length of the key to be used to encrypt the communication.
- Encryption layer: Layer to be encrypted between the nodes (IP, Ethernet, Optical).
- Refresh type and value: type (e.g. time, amount of information) and value to update keys used in a service.
- Algorithm: The encryption algorithm used for the communication.

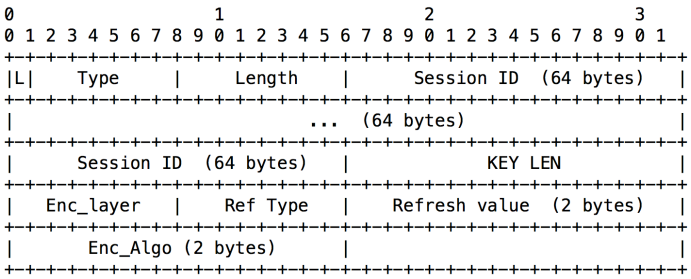

Fig 3 The proposed QE ERO subobject structure.

Handshaking these parameters between PCE and PCCs must happen in two phases:

- The PCC sends the PCRequest with certain parameters to consider in the path computation. These parameters are: key length, destination, encryption layer, refresh type and value and algorithm. This information is sent adding new metrics in the metric list of the PCRequest.
- The PCE should respond with a path inside the PCReply. This path contains new subobjects placed in the ERO just after the subobjects identifying the nodes that have to perform the encryption. They contain a session ID (initialized as zero), key length, encryption layer, refresh type and value and algorithm, as shown in Fig. 3.

Upon response, the source node analyzes the PCReply and starts the signaling process (and the key synchronization, in case of being source node of the QE service).

C. Signaling and Key Synchronization

When a PCReply or a PCInitiate message arrives to a PCC of a network node, this node is in charge to start the signaling process by extracting the ERO from the PCEP message and transmitting it via an RSVP Path message. Other intermediate nodes forward these messages down to the destination node, getting the required configuration parameters from the message. When this message arrives to the destination node, assuming that no error has occurred, it responds back with a RSVP Resv message, informing that all the necessary resources have been reserved.

When performing quantum encryption, considering any layer up to the application layer, one mandatory step is to synchronize the keys in both sides. This process can be easily done by exchanging IDs to identify the session, or the key in both extremes through an open channel. RSVP is the best candidate to automate the key synchronization process. It is capable of forwarding the encryption requirements across the path and return a confirmation (Resv message) if the resources (keys) have been reserved, while the signaling process traverses the network. This mechanism is defined as follows:

- The QE-capable source node finds itself inside the ERO, and detects the new QE ERO subobject as well.
- This node decodes the session ID, sees that it is set to zero (64 bytes) and extracts a new key and session ID (key_handle) from the key server using an interface based on ETSI GS QKD 004 v1.1.1 specification [7].
- The source node stores the key and injects the session ID inside the QE ERO subobject of the destination node.
- The destination node receives the RSVP Path message, finds itself inside the ERO and detects the QE ERO subobject.
- As the session ID is not zero, the destination node extracts the key using the session ID through the same interface as above [7].

Both nodes extract the other parameters (encryption layer, key, length, refresh type and value and algorithm) from the QE ERO subobject as well, to be used afterwards by the device in charge of the encryption. The QE service will finish its deployment process when the source node receives the RSVP Resv message. In this moment the keys would be available for any upstream service, like AES or 3DES encryptors.

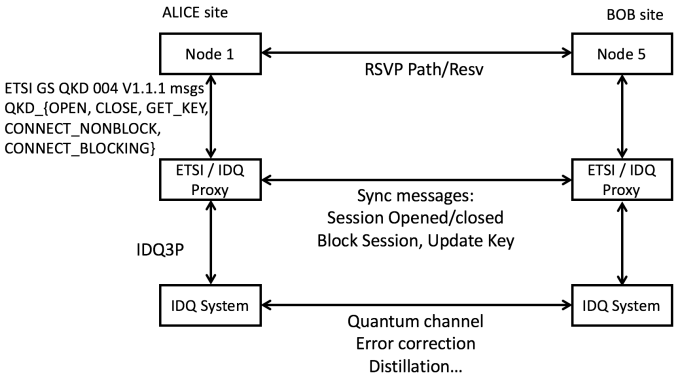

Fig 4 Logical representation of the set of nodes and messages used to extract and synchronize keys.

VI. SCENARIO, IMPLEMENTATION AND EMULATION PLATFORM

The emulated control plane scenario for this demonstration is equivalent to the one shown in Fig. 2. It comprises five

GMPLS-enabled nodes (logically connected as shown in Fig. 2) and a PCE. We run an additional client that connects via telnet to the first device to start the workflow. In addition to this, nodes 1 and 5 are connected to "Alice and Bob" QKD domains, providing keys in two different layers. We emulate the ID Quantique Clavis2 3100 system by having a set of predefined keys in two different nodes that provide IDQ3P interfaces, which is the ID Quantique proprietary interface of the Q3P protocol [13].

These two nodes communicate with two proxies that provide, as a northbound interface, a protocol based on the API defined in [7]. They synchronize the sessions maintained between themselves and provide the session IDs (key_handle) that are kept to extract keys until the service is finished. Keeping the same session ID to extract keys in both sides is convenient, as IDs can be mapped one to one to services. This can be used to update keys dynamically in both extremes when a service requires to do so. Fig. 4 shows the logical connectivity between nodes 1, 5, and Alice and Bob domains.

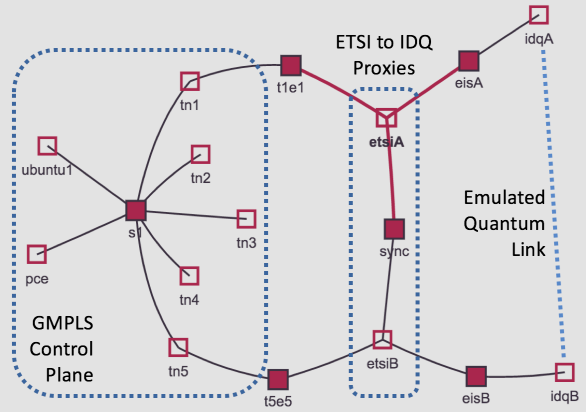

Fig 5 Capture of the Dockernet GUI, showing the set of GMPLS nodes (tn1-5), PCE, client (ubuntu1), proxies (etsiA-B), ID Quantique devices and OVSs (dark nodes).

To perform this test, we have implemented a platform [14] that allows the creation of virtual networks using Docker [15] for the deployment of containers and Open vSwitches to provide the required connectivity. Six IP domains are created to have the traffic isolated (GMPLS control plain domain, two domains to connect the nodes 1 and 5 to the proxies, two domains to connect the proxies to the IDQ3P interfaces and an additional one for the synchronization channel). Our control plane topology is the one shown in Fig. 5, where the dark nodes are OVS and the rest of them are containers running either java or python processes.

```
22.910357  10.1.1.5    224.0.0.5   OSPF  LS Update
30.691575  10.1.1.1    10.1.1.200  PCEP  Path Computation Req
30.733564  10.1.1.200  10.1.1.1    PCEP  Path Computation Rep
30.799866  10.1.1.1    10.1.1.2    RSVP  PATH Message. SESSIO
30.852555  10.1.1.2    10.1.1.3    RSVP  PATH Message. SESSIO
30.892129  10.1.1.3    10.1.1.4    RSVP  PATH Message. SESSIO
30.941130  10.1.1.4    10.1.1.5    RSVP  PATH Message. SESSIO
30.964791  10.1.1.5    10.1.1.4    RSVP  RESV Message. SESSIO
30.974283  10.1.1.4    10.1.1.3    RSVP  RESV Message. SESSIO
30.988739  10.1.1.3    10.1.1.2    RSVP  RESV Message. SESSIO
31.017170  10.1.1.2    10.1.1.1    RSVP  RESV Message. SESSIO
```

Fig 6 Set of messages exchanged among nodes, and with the PCE.

The GMPLS agents and the PCE have been implemented in Java, using Netphony [16] open source project. The IDQ3P interface and the proxies have been implemented in Python.

```
▼ 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
```

Fig 7 Opaque RI LSA and Informational Capabilities TLV.

VII. EXPERIMENT AND RESULTS

The full set of required messages transmitted across the GMPLS control plane is shown in Fig. 6. The first message is an OSPF update from the fifth node (others are omitted), which contains the router information opaque LSA, with the traffic engineering capable and que QE capable bits set to 1 within the informational capabilities TLV (Fig. 7). The second and third messages are the PCRequest and PCReply messages.

```
▼ 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
```

Fig 8 Metrics sent inside the PCRequest message for the QE service.

Fig. 8 shows the 4 metrics that define the service, included as follows: (1) Encryption algorithm, Type: 253, Value: 10, (2) Refresh Type: 252, Value: 1000, (3) Key length Type: 255, Value: 32, (4) Layer of encryption Type: 254, Value: 2.

```
▼ 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)
```

Fig 9 Explicit route object including the new QE ERO subobject.

Upon request, the PCE decodes the multiple metrics and injects them in the QE ERO subobjects, together with an unset session ID. Fig. 9 shows the ERO, which includes the new

defined ERO subobjects. Both (source and destination of the QE service) are placed just after the subobjects that identify the nodes with QE capabilities.

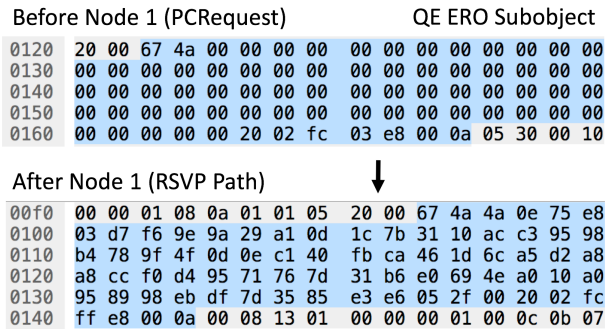

Fig 10 QE ERO subobject change to signal the session ID.

The last set of messages are RSVP Path messages (node 1 to 5) and RSVP Resv messages (on the opposite direction). These messages contain the same ERO as the one that comes inside the PCReply (Fig. 9). This ERO is maintained until the last node of the path, except for the changes introduced by the source node of the QE service. This node, by finding out the requirement of establishing QE, extract a new key and a new session ID (if previously set as zero), and modifies the QE ERO subobject of the destination node, adding the new session ID which is mandatory to synchronize the key in both points. The process of signaling the session ID can be observed in Fig. 10 where the first image shows the QE ERO subobject received by the source node inside the PCReply (third message) and the second shows the same object when transmitted from node 1 to 5 inside the RSVP Path message. The entire workflow from the PCRequest until the last RSVP Resv message takes around 300 ms, bearing in mind that the emulation was run in a single computer, with no other processes running to increase the latency. Other messages have been omitted to improve readability.

Finally, Fig. 11 shows a list of UDP messages within the Alice and Bob domains which are used for key extraction, synchronization between proxies and the IDQ3P messages. The secret key throughput of state of the art QKD devices at typical metropolitan distances is in the range of Mbps. Using the acknowledged 2^40 of data per key limit for DES [17], these secret key rates would allow to encrypt multi Tbps links with a much higher security level than what is standard today.

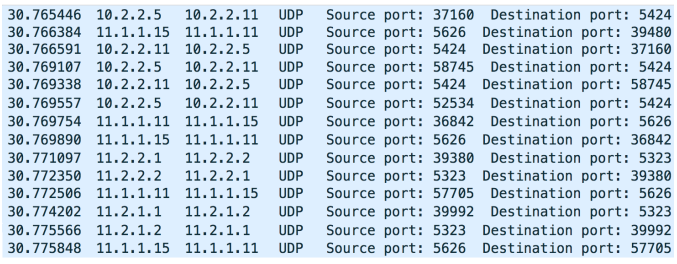

Fig 11 Set of UDP messages between Alice and Bob proxies and IDQ3P interfaces and nodes 1 and 5.

VIII. CONCLUSION

The integration of QKD systems in existing networks and the coexistence with classical channels, although still a matter of research, has been widely studied. However, their automation and integration in current infrastructures and novel network paradigms is still missing, with vendors manufacturing devices whose integration is not straightforward. With the aim of achieving quantum devices that can integrate as a plug-and-play technology into networks, this work proposes and demonstrates a full architecture for a use case where quantum devices are used to provide E2E services. In this paper, we propose for the first time a collection of node architectural and control plane requirements for Quantum Enabled service automation in a GMPLS environment. Furthermore, we implement a set of extensions to the GMPLS protocol suite to demonstrate how existing protocols can be used to perform the required handshakes for security sessions and the key synchronization for quantum encryption. This work demonstrates a way to integrate this new type of service in SDN networks, such that operators can easily deploy and commercialize it. We have chosen GMPLS as it is a set of protocols widely deployed. Other protocols, like OpenFlow and NETCONF will be the subject of future research.

REFERENCES

[1] T. Szyrkowiec, M. Santuari, M. Chamania, D. Siracusa, A. Autenrieth and V. López, "First Demonstration of an Automatic Multilayer Intent-Based Encryption Assignment by an Open Source Orchestrator," Post-Deadline Paper in *European Conference on Optical Communication (ECOC)*, 2016.

[2] N. Gisin, G. Ribordy, W. Tittel, and H. Zbinden, "Quantum cryptography," Rev. Mod. Phys., vol. 74 pp 145, 2002.

[3] D. Lancho, J. Martinez, D. Elkouss, M. Soto, and V. Martin, "QKD in Standard Optical Telecommunications Networks," in *QuantumComm 2009, International Conference on Quantum Communication and Quantum Networking*, LNICS, vol. 36, pp. 142-149, 2009.

[4] T. Jimenez, V. López, F. Jimenez, O. Gonzalez and J. P. Fernandez, "Techno-economic analysis of transmission technologies in low aggregation rings of metropolitan networks," in *Proc. Optical Fiber Conference (OFC)*, 2017.

[5] M. Peev *et al.*, "The SECOQC quantum key distribution network in Vienna," New J. Phys., vol. 11, pp 075001, 2009.

[6] M. Sasaki *et al.*, "Field test of quantum key distribution in the Tokyo QKD Network," Opt. Express, vol. 19, pp 10387-10409, 2011.

[7] ETSI GS QKD 004 V1.1.1, "Quantum Key Distribution (QKD); Application Interface", 2010-12.

[8] A. Farrel, J.-P. Vasseur, and J. Ash, "A Path Computation Element (PCE)-Based Architecture," RFC4655, 2006.

[9] M. Cuaresma, F. Muñoz, S. Martinez, A. Mayoral, O. Gonzalez de Dios, V. López, and J. P. Fernández-Palacios, "Experimental demonstration of H-PCE with BGP-LS in elastic optical networks," *in European Conf. on Optical Communication (ECOC)*, London, UK, 2013, paper We.4.E.3.

[10] D. King, and A. Farrel, "A PCE-based architecture for application-based network operations," IRFC 7491, 2015.

[11] A. Aguado *et al.*, "ABNO: A feasible SDN approach for multivendor IP and optical networks," IEEE/OSA Journal of Optical Communications and Networking, vol. 7, Issue. 2, pp. A356-A362, 2015.

[12] A. Lindem, N. Shen, J. P. Vasseur, R. Aggarwal, and S. Shaffer, "Extensions to OSPF for Advertising Optional Router Capabilities", RFC7770, 2016.

[13] O. Maurhart, "QKD Networks Based on Q3P," Lect. Notes Phys., vol. 797, pp. 151-171, 2010.

[14] [Online]. Github Dockernet-tool. Available: https://github.com/alexaguado/DockerNet (Accessed January 31, 2017).

[15] [Online]. Docker. Available: https://www.docker.com/ (Accessed January 31, 2017).

[16] [Online]. Java Library of Networking Protocols: PCEP, RSVP-TE, OSPF, BGP-LS. Available: https://github.com/telefonicaid/netphony-network-protocols (Accessed January 31, 2017).

[17] G. Van Assche: "Quantum Cryptography and Secret-Key Distillation." Cambridge University Press (2006)