Virtual Network Function Deployment and Service Automation to Provide End-to-End Quantum Encryption

Alejandro Aguado, Victor Lopez, Jesus Martinez-Mateo, Momtchil Peev, Diego Lopez, and Vicente Martin

***Abstract*—The nature of network services has drastically changed in recent years. New demands require new capabilities, forcing the infrastructure to dynamically adapt to new scenarios. Novel network paradigms, such as software-defined networking (SDN) and network functions virtualization, have appeared to provide flexibility for network management and services. The reliance on software and commoditized hardware of these new paradigms introduces new security threats and, consequently, one of the most desired capabilities is a strengthened security layer when connecting remote premises. On the other hand, traditional cryptographic protocols are based on computational complexity assumptions. They rely on certain mathematical problems (e.g., integer factorization, discrete logarithms, or elliptic curves) that cannot be efficiently solved using conventional computing. This general assumption is being revisited because of quantum computing. The creation of a quantum computer would put these protocols at risk and force a general overhaul of network security. Quantum key distribution (QKD) is a novel technique for providing synchronized sources of symmetric keys between two separated domains. Its security is based on the fundamental laws of quantum physics, which makes it impossible to copy the quantum states exchanged between both endpoints. Therefore, if implemented properly, QKD generates highly secure keys, immune to any algorithmic cryptanalysis. This work proposes a node design to provide QKD-enhanced security in end-to-end services and analyze the control plane requirements for service provisioning in transport networks. We define and demonstrate the necessary workflows and protocol extensions in different SDN scenarios, integrating the proposed solution into a virtual router providing QKD-enhanced IPsec sessions.**

***Index Terms*—MPLS; NETCONF; OpenFlow; Quantum key distribution; Service automation; Software-defined networking.**

I. Introduction

The network infrastructure is evolving from static, black-boxed, and monolithic approaches towards dynamic and open solutions. 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 based on the development of different software paradigms, where multiple functions, actions, and operations that usually run internally in a network device are abstracted and executed remotely as software processes. One of these novel network paradigms, called software-defined networking (SDN) [1], permits decoupling the control plane (control/management protocols and actions) from the data (forwarding) plane. This separation allows dynamic management of network services and the infrastructure from a logically centralized management entity, called a SDN controller. SDN copes with the changing behavior of new services, increasingly demanding more capacity together with new capabilities, as they are developed and made available in the market. One of the key demands is to have an enhanced security layer at the network level, while keeping the current infrastructure intact. Despite the behavior of future network services being unpredictable, it is certain that communication networks will remain the core to support the forthcoming traffic. Therefore, securing this infrastructure is an increasing concern, as critical information travels across an entire infrastructure. Up until now, network security has been achieved with as a series of *ad hoc* solutions. Today's networks are more complex and, especially with SDN, much more configurable. The security risks are correspondingly larger and the security in the network infrastructure must be enhanced.

Quantum key distribution (QKD) technologies [2,3] allow generation synchronized random bits in two sources that are separated in space, with the additional property that the maximum information leaked out of these sources can be upper-bounded, i.e., the random bits can be used as a secret key. The security of the QKD-produced keys is rooted on the physical layer and, therefore, immune to any algorithmic cryptanalysis. QKD is a new opportunity for operators and infrastructure providers as it can bring an additional physical layer for securing control and data communications. Authors in Ref. [4] demonstrated how QKD is a suitable technology for securing control plane channels by combining existing key exchange algorithms together with quantum-generated keys in a hybrid cryptosystem. In this work, we propose and demonstrate a scenario for

Manuscript received December 4, 2017; revised February 26, 2018; accepted February 28, 2018; published March 30, 2018 (Doc. ID 314936).

A. Aguado, J. Martinez-Mateo, and V. Martin (vicente@fi.upm.es) are with the Center for Computational Simulation, Universidad Politecnica de Madrid, Campus Montegancedo, 28660 Boadilla del Monte, Madrid, Spain.

V. Lopez and D. Lopez are with Telefonica GCTO, Ronda de la Comunicacion s/n 28050 Madrid, Spain.

M. Peev is with Huawei Technologies Duesseldorf GmbH, Riesstrasse 25, 80992 Munchen, Germany.

https://doi.org/10.1364/JOCN.10.000421

combining QKD systems with secure end-to-end (E2E) services between remote premises. This paper extends the work done in Ref. [5], where authors presented a proof of concept for key synchronization using control plane emulation [Open Shortest Path First (OSPF), Resource Reservation Protocol (RSVP), and Path Computation Element Protocol (PCEP)]. The creation of connections using this approach is called quantum encryption (QE) service. Here, we define, implement, and experimentally validate the QE service to operate with multi-protocol label switching (MPLS) [6], OpenFlow [1], and NETCONF [7]. The dynamic creation of the QKD key synchronization operations together with the E2E-encrypted services is a key requirement for operators to deploy these services in production networks. The current service creation process (manually operated) does not meet the operators' needs for next-generation networks. To the best of the authors' knowledge, this is the first work that demonstrates the control plane configuration as well as the data plane setup by instantiating two virtual network functions (VNFs) that connect two remote virtual networks via QE service.

This work is organized as follows: Section II provides an overview about current QKD technologies and networks, outlining the main benefits, challenges, and restrictions. Section III defines the virtual router structure, explaining which software components and interfaces are used to provide the desired encryption capabilities. Section IV describes the proposed workflows for each protocol. Section V explains the required extensions to make every workflow operation possible. Section VI exposes the platform and the design of the testing scenario. Section VII presents the results and captures of the control plane traffic for each protocol and the traffic of the final service. Finally, Section VIII finally concludes the work.

II. Quantum Key Distribution Networks

QKD [3] is a technology that allows the growing of a symmetric key shared among the two endpoints of a quantum channel. A quantum channel is the physical media used to transmit the quantum signals (qubits). In our case it is the optical fiber. QKD also requires the existence of a public but authenticated classical channel that is used to distill the secret key out of the raw detections of quantum signals. An initially small secret is assumed to be shared among the two legitimate users in order to authenticate themselves during the first communication. This is a short-lived secret since afterwards the new communication rounds can be authenticated using the QKD-produced keys. QKD keys have the distinct advantage that they are not algorithmically correlated in any sense. Thus, keys obtained in different rounds have no relation among them, and forward and backward secrecy is guaranteed; an attacker that by some means obtains one of the keys will be unable to derive any other. Under reasonable assumptions, e.g., that no eavesdropper sits inside the QKD device, that the protocols are executed correctly and that the physical implementation is also correct, it can be demonstrated to be absolutely secure. This means that the amount of information on the secret key leaked outside the systems is bounded and that the bound can be made arbitrarily small. In other words, QKD is an information theoretical secure (ITS) primitive. It is to be noted that the ITS character is not necessarily inherited by the rest of the cryptographic chain and that it depends on the methods used after the QKD primitive have been used. For example, if Advanced Encryption Standard is used to cypher messages instead of a one-time pad, the result will not be ITS, no matter if the keys used came from a QKD process or not.

This is in contrast with conventional cryptography, where the secrecy is based on algorithmic complexity assumptions. These assumptions are not demonstrated, and its security is just based on the belief that nobody has been or will be able to solve the mathematical problem that guards the security of the secret within its desired lifetime. Forward and backward security are also not guaranteed, since secrets are algorithmically related, and their security is built upon the assumption that the mathematical problems cannot be broken using the computational resources available.

Although this is the de facto and well-established way of working that has served us well, the existence of algorithms breaking the mathematical problems underlying the security of Rivest, Shamir y Adleman (public-key cryptosystem), Diffie–Hellman, or Elliptic Curve cryptography cannot be ruled out. In fact, it is the advent of quantum computers and Shor's algorithm that has triggered the obsolescence process of these algorithms recommended by the US National Security Agency [8] and the flurry of research activity in new algorithms.

On the other hand, algorithmic-complexity-based cryptographic protocols have the advantage of not being limited by a physical implementation. QKD is intrinsically distance-limited because qubits have a non-zero probability to interact with the transport medium. Any signal propagating in a medium suffers an exponential attenuation, being critical when the signals are composed of a single quantum. Also, the interactions with the environment are indistinguishable from the action of a spy, and errors must be treated as if they were the action of an eavesdropper, thus heavily penalizing the secret key throughput. As a typical example, in the case of qubits and optical fiber, losses are about 0.2 dB per km when the qubits are encoded in photons at 1550 nm. QKD systems working with maximum losses of about 30 dB have been demonstrated, and the technology is reaching maturity quickly. This means that today's practical limit in distance—not taking into account one-of-a-kind laboratory efforts—is about 150 km. Going to this distance also means a strongly reduced secret key rate output. Top performance figures are about 1 Mbit of final secret key at 40 km distance in direct links, without crossing any passive network equipment that would increase losses. Other options, better suited for network usage, are also being demonstrated. Also, quantum repeaters [9] can be potentially built, but this requires technology that may be years in the future.

QKD technologies have been demonstrated in metropolitan areas in networks of exclusive use for the quantum parts that run in parallel with the telecommunications network [10,11], and the capability to run in an integrated way with conventional telecommunications networks is also being explored [12,13]. The former ones emphasized the usage of different QKD implementations working in a trusted node regime to allow key distribution and forwarding among any node within the QKD network (and also to logically connect physically incompatible devices). In a trusted node regime, the system is assumed to be protected under a security perimeter. This system can be composed of multiple subsystems or devices (e.g., a host computer or a QKD endpoint). Within a trusted node, ITS mechanisms (e.g., a simple Exclusive OR among the QKD keys generated pairwise between directly connected nodes) can be used to forward the keys and traverse the network to securely connect two remote locations. The later ones emphasized direct switched connections using as many off-the-shelf optical components as possible. It is the optical infrastructure of this later type [13] that we have used in the present paper. In the last years, a surge of interest in the technology has drawn the attention of several companies, and the availability of QKD devices from different sources—an important issue in the security market—seems larger and more promising than ever.

While the distance limit might currently seem an insurmountable one, in practice this is not such a problem in metropolitan areas working with modern network paradigms [14]. From a security point of view, SDN and network functions virtualization (NFV) designs mandate that network controllers and virtualized network functions run in a protected environment. Thus, the important figure here is the distance that separates the security perimeters protecting the points of presence—trusted nodes—where the controllers and VNFs are running. In a metropolitan area, these are within present-day QKD limitations and, thus, QKD is a realistic technology on which to base a physical security layer to protect infrastructure and network services. From a logical point of view, the QKD links can be seen as devices that extend the security perimeter of the points of presence to the optical fiber connecting them, thus making it possible to securely share keys among any connected nodes.

III. Virtual Router Design

The NFV [15,16] paradigm is also at its peak in terms of development and technological advances for next-generation networks. It uses concepts from traditional computing virtualization to encapsulate functionalities from network devices into software instances. It allows dynamic allocation of functions (such as firewalls, switches, routers, deep packet inspectors, etc.) in distributed environments, reducing time and costs in deploying new infrastructures. In this work, we propose the integration of basic routing functions, together with IPsec point-to-point sessions using QKD-generated keys, all automated via standard protocols. Figure 1 shows an example of the high-level structure of the VNF design. It is divided into northbound interfaces (NBI), southbound interfaces (SBI), and core (forwarding):

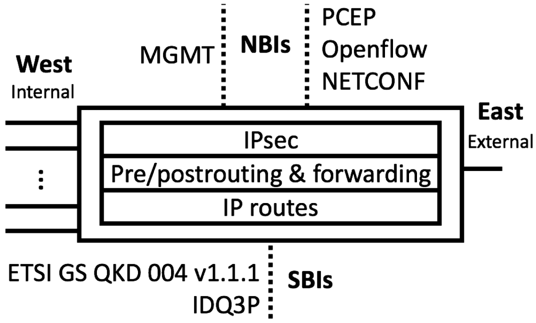

Fig. 1. Schematic view of the virtual router capabilities and interfaces.

- As a NBI, our solution provides two different points of access: a management interface to the NFV control framework and a second one connected to the network controller (path computation element, PCE [17]; SDN controller; or NETCONF manager). The first interface receives commands in order to control the lifecycle of the VNF as well as to provide connectivity to the system itself. The second one allows remote control for network service management and automation via network (SDN) controller.
- In the core of the node, we control the traffic from/to our private network using "ip route" and "iptables" rules. For setting up IPsec sessions, we use an extension of the ipsectools code to inject the keys extracted from the QKD systems via PF_KEY key management API [18].
- As a SBI, the virtual node connects to the QKD systems using an interface based on [19]. This interface allows the node to specify a Quality of Service for the key session, and provides a unique session ID for key extraction.

This virtual node is also composed by east–west (internal–external) interfaces for packet forwarding in the data plane. Upon deployment, the node runs the agent in charge of communicating with the controller for further configuration of its core functionalities (forwarding and secure sessions) when required.

IV. Proposed Workflows

As described in Ref. [5], the main operations in any workflow for setting up a QE–E2E service are: capabilities dissemination, key synchronization, and device (service) configuration. Although any protocol must take care of the same set of actions and finally deploy a similar configuration, the different protocols fundamentally differ in the way these actions are performed, as they were defined for different purposes and use various collections of messages for their operations. Figure 2 shows a high-level example of the main control plane instances and interactions among the entities involved using the considered protocols. In this figure, we can see two nodes (QE nodes) that have the capability of encrypting the traffic using QKD-generated keys, an intermediate network (cloud), and a network controller. It is important to note that any

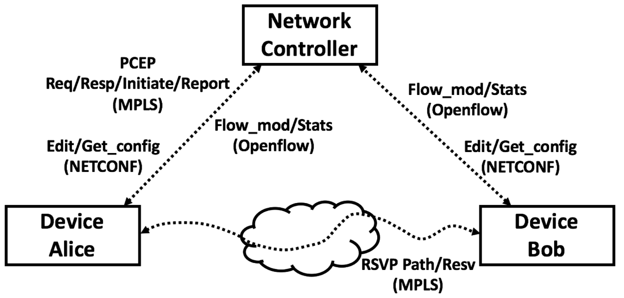

Fig. 2. Generic view of control plane's messages involved in QKD key synchronization process.

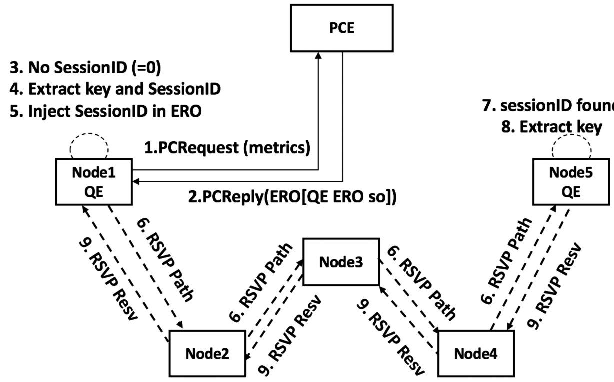

Fig. 3. MPLS workflow for setting up a quantum encryption service.

workflow could be initiated either via the controller's NBI or from the network device itself. This work mainly focuses on the service deployment (key synchronization), while the capabilities dissemination is demonstrated as a proof of concept by sharing basic information (flags) between device and controller. The main information to be transmitted is as shown in Table I, while the workflow description associated to each protocol and the main differences among them are as follows. An important issue in deploying VNFs is the mutual authentication problem. In this work we do not deal explicitly with this problem, but point out that it can be tackled in several ways. One possibility that we have used in several tests is to use the same tool kit that we integrated for the encryption (ipsectools) and create new authentication rules for the IPsec session (using, as an example, AH and hmac-md5) with keys extracted from the QKD link.

A. MPLS Protocol Suite

Figure 3 shows the workflow for synchronizing a QKD key and setting up a QE service between two endpoints. Initially, both nodes must expose their capabilities to the PCE via interior gateway protocol (IGP). Regardless of whether the request is initiated by an active PCE, a northbound application, or from the network node itself, the PCE will transmit the configuration via PCEP message (initiate or response). Upon receipt, the source node in the path detects that QE is required by checking the explicit route object (ERO) and that the keyID is not yet set, extracting a valid key and keyID pair from the QKD system. The keyID is encapsulated and forwarded via RSVP to reach the destination node. The destination node finds the valid keyID (among other parameters) and uses it to extract the key for the secure E2E channel. The workflow finalizes when the confirmation (RSVP Resv) arrives to the source node. The workflow is compatible with the generalized version of the protocol suite (GMPLS).

B. OpenFlow Protocol

OpenFlow was designed as an enabler for SDN, allowing remote management of the forwarding plane using a controller. In that sense, OpenFlow differs from MPLS, as it was not defined for device-to-device communications. The workflow must change accordingly, as the key synchronization cannot happen directly between devices, and it should be orchestrated by the SDN controller. Figure 4 shows the proposed workflow for OpenFlow. Initially and similarly to MPLS, the devices should expose the QE capabilities to the controller. Once this has been done, the controller can create the service, regardless of who initiated the request (packet_in message from the device, static service/intent NBI request, etc.). The controller sends a flow with a new action specifying the parameters for the encryption. Within those parameters, the controller sends an unset keyID to be detected and modified by the device. When the device receives the flow, it extracts the key and keyID pair from the QKD system and saves the ID with the flow information, which can be remotely accessed. The

TABLE I
PARAMETERS TO BE EXCHANGED BETWEEN DEVICE AND CONTROLLER FOR THE QE-SERVICE

Parameter	Description
KeyID (Key_handle)	Identifier used to synchronize QKD key sessions in separated endpoints
Key length	Length of the key used to encrypt the current channel
Destination	Other endpoint (remote peer) of the encrypted channel
Source	Ingress node of the channel (only used in the request to the controller); For node configuration, destination (remote peer) is used
Encryption layer	Layer in which the traffic is encrypted (e.g., Ethernet, IP)
Encryption algorithm	Value to specify the symmetric encryption algorithm
Refresh type	This value identifies the type of refresh considered to update the key (e.g., time, length)
Refresh value	This value specifies the amount considered for a given type used to refresh a key

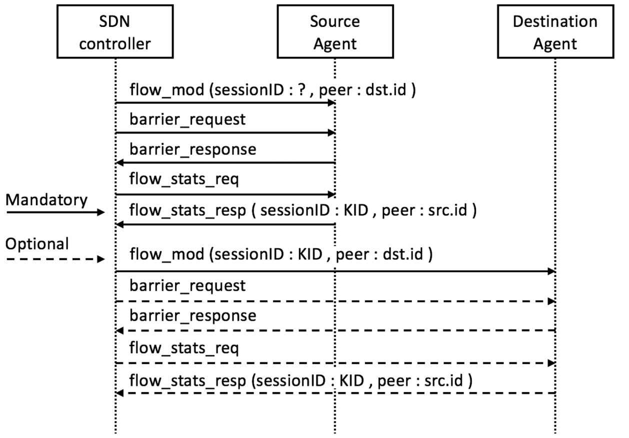

Fig. 4. OpenFlow workflow and messages enabling the QE service.

controller waits until the key is extracted and the flow is installed using a barrier request, retrieves the keyID from the device from the flow statistics, and sends the same flow to the destination node with an updated keyID. This process ends when the second (destination) node installs the flow.

C. NETCONF

NETCONF is a transactions-based protocol standardized by Internet Engineering Task Force that provides access to network device configuration. The data transmitted using the NETCONF protocol is defined using a modeling language called YANG, and usually encapsulated into XML data structures. The workflow in this case is similar to the OpenFlow one (Fig. 5), only differing in that it is not necessary to wait until the configuration has finished, as NETCONF is transaction-based, and any misconfiguration or error setting up the QE session (such as unavailable keys) will reverse the whole set of instructions. In this way, the controller will get the capabilities of the node from the hello message, and will send an edit-config command to the device with an unset keyID. When the node finalizes its configuration, the controller sends a get-config message to

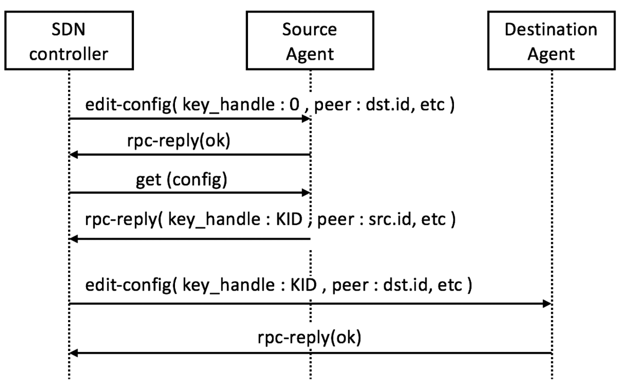

Fig. 5. NETCONF workflow for QE service.

retrieve the keyID set in the source node, sending a final edit-config to the destination node with the valid keyID. As the key extraction from the QKD system is part of the transaction, the source node will not respond to the edit-config until the key is obtained (or return an error if there are no available keys). When the response reaches the controller, it can gather the key ID via get-config.

The key difference between workflows resides mainly between MPLS and OpenFlow/NETCONF, as the actual deployment of the service (and, therefore, the key synchronization process) is forwarded differently: directly between the network nodes across the path or via network controller. OpenFlow and NETCONF only differ in the way the controller and devices communicate for the key synchronization (transaction-based Secure Shell (SSH) channel versus sequential OpenFlow commands synched via barrier request). After the service deployment, dynamic rekeying must be performed by the nodes. Updates over the control plane could cause overheads in the management network and the controller, which may be requested to handle hundreds or thousands of requests per second, depending on the number of services and the rekeying constraints of each of them. In this sense, nodes should synchronize between them for rekeying purposes, leaving service updates over the control plane for some specific situations (routing changes, issue with insufficient keys available, change on the service level agreements, etc.). The work shown in Ref. [20], which describes a protocol and method for synchronizing QKD keys in IPsec sessions, is a solution that could be integrated to support the dynamic rekeying after the service deployment operations.

V. Protocol Requirements

The workflows described in Section III require various protocol extensions and information models to be defined. The proposed solutions will mainly focus on the mechanism for key synchronization and the transmission of the basic data used for the encrypted channel. Additionally, we will propose some extensions to include the new device capabilities into these protocols, to make the controller aware of which nodes do support these services. The main purpose of the dissemination is to expose the QE capabilities (for simplicity, as a single bit). Once this is done, adding further capabilities (e.g., supported algorithms, encryption layers, etc.) is easy, although they are out of the scope of this paper and left aside for simplicity.

A. MPLS Protocol Suite

1) Device Capabilities: The MPLS protocol suite provides various solutions for exchanging topological information, depending on whether the information is shared internally (interior gateway protocol, IGP) or externally (border gateway protocol, BGP). Within the IGP, we have chosen the OSPF protocol to include the QE capabilities, utilizing the extensions to advertise optional router capabilities defined in the RFC7770 [21]. The router informational capabilities' type, length, and value (structure) contains 4 bytes exposing the optional capabilities, where the bits 6–31 remain unassigned. The QE-enabled nodes will use the seventh bit to expose to other nodes and the PCE its new capability. When the OSPF message arrives, the PCE will store this capability in its traffic engineering database.

2) Key Synchronization and Service Configuration: For initiating the deployment, the PCE has to receive a request that might come from different entities [the network device, a northbound user, network management system (NMS) or application, etc.]. Independently of who initiated the request, a PCEP request containing the encryption requirements must be transmitted, as these requirements are encapsulated in new metric objects (used later by the PCE). The transmitted metrics in our case are: encryption layer, encryption algorithm, key length, and refresh type and value. Upon receipt, the PCE will compute the required path, including the main parameters extracted from the metrics and an unset keyID inside a new QE explicit route object (ERO) subobject (as explained in Ref. [5]). These new subobjects must be located after the nodes that are required to encrypt the traffic. The ERO is either returned as a response to whom requested the path, or as an initiate message to the source device in the path. The remaining part of the workflow does not require any additional extensions, apart from the manipulation done by the source node in order to inject the valid keyID. This insertion is done inside the QE ERO subobject placed after the destination device of the encrypted path.

B. OpenFlow Protocol

1) Device Capabilities: Similarly as in the MPLS scenario, it is intended to expose the QE capabilities in a basic manner. For this purpose, we have included a single bit in the features reply message, the QE-capable bit (17th bit), within the capabilities field. As it will be explained below, the extension for the flow configuration will consist of a new action. Therefore, a new action could be added as well inside the feature reply if desired. Further capabilities (layer of encryption, algorithms, etc.) are left out of the scope, and could be added later as featured actions/capabilities or as new experimental messages. The SDN controller will identify, when this message is processed, which nodes are capable of performing a QE flow and proceed with configuring the QE service/intent.

2) Key Synchronization and Service Configuration: An OpenFlow rule is formed by a match/action pair (flow), used to identify and modify the incoming packets in accordance with the device's internal flow table. When considering the possibility of encrypting specific incoming packets, the same duple must be considered to identify (match) the traffic and apply the desired encryption to it (action). There is no need to modify or extend the matching process inside the device, as OpenFlow does support traffic matching for different layers. The main addition required consists of a new action specifying what to do (how to encrypt) with the incoming packets matched by the QE flow. Specifically, we have defined a new action (action type OFPAT13_QKD=0xFFFC), which contains the keyID, key length, the destination (IPv4), encryption layer and algorithms, and refresh type and value (if required) to be used by the device. When the device receives the FLOW_MOD OpenFlow message from the controller, it extracts a new key and keyID pair from the QKD system (if the keyID is unset), saving the valid keyID in the flow's action field. The synchronization process is handled by the SDN controller and uses OpenFlow stats to retrieve the flow information (in this particular case, to retrieve the keyID), but it does not require additional extensions, as it includes the new action described for the FLOW_MOD operation.

C. NETCONF

1) Device Capabilities: Two devices interacting via NETCONF protocol initiate their communications with a HELLO message. This message includes all the YANG models (also known as capabilities) supported by both ends. To enable the deployment of the QE service, we have created a new YANG model to configure the different parameters previously mentioned. When the hello message is transmitted, it includes a URL pointing to the new YANG model to be used to encrypt an end-to-end communication using QKD keys. The NETCONF manager (controller), upon receipt, stores this capability on its database to be used when required by administrators or applications.

2) Key Synchronization and Service Configuration: Due to the flexibility that YANG and NETCONF provide to create models for new capabilities and configurations, the required work is much simpler than in the other protocols. To achieve this goal, a new YANG model has been defined to include all the parameters required for the QE service. These parameters are, as previously mentioned: a keyID, key length, the destination (IPv4), encryption layer, algorithm, and a refresh type and value (if necessary). The NETCONF server (device) is the one in charge of retrieving and keeping the new QKD keys and their IDs, being (the IDs) gathered by the manager afterwards. No other extension is required, as the same model is used to configure (edit-config) and gather (get-config) the necessary information for the key synchronization.

VI. Platform Implementation

To showcase the proposed scenario and extensions, we have created a tool (DockerNet) on top of the Docker container platform. This tool allows setting up virtual networks in different servers based on containers and virtual switches. The containers deployed for this demonstration encapsulate different functionalities that are used to create the service and are connected in a distributed network, as shown in Fig. 6. The left part of the network includes a remote data center (DC) network providing database and web services, which are accessible through a virtual router. This virtual router forwards internal and external traffic

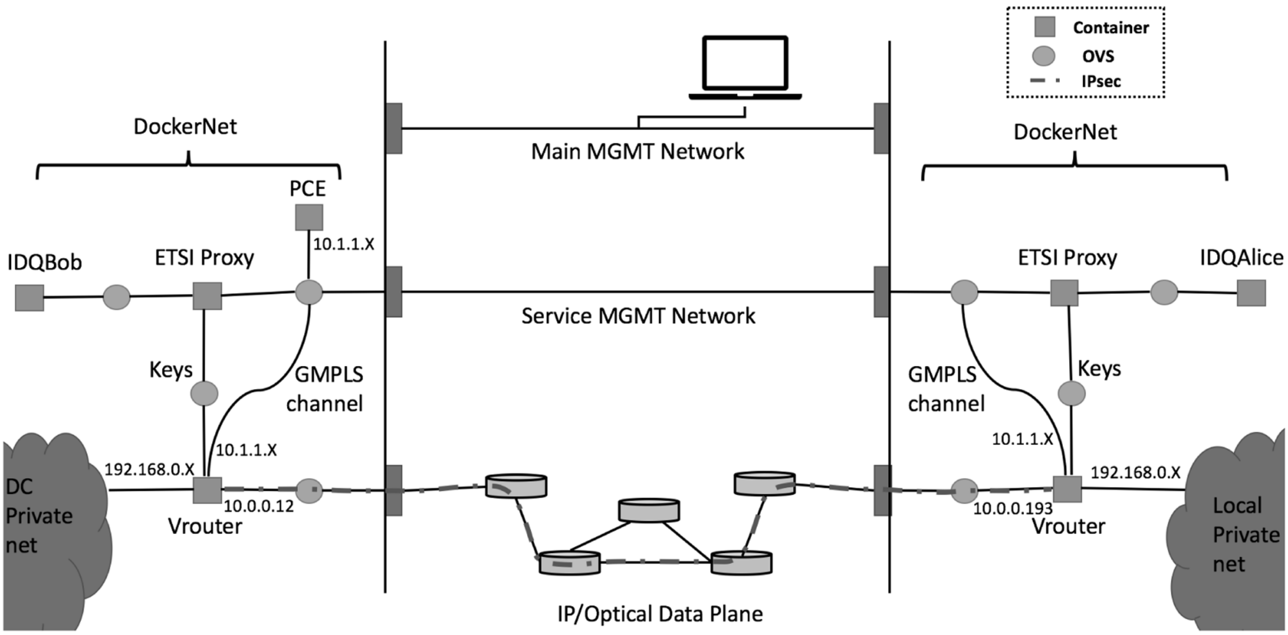

Fig. 6. Logical scheme describing the network used for the end-to-end quantum encryption service via IPsec. The left part shows the DC management and data networks, with a QKD domain (Bob) and a virtual router connected to a PCE. The right part exposes the local network, connecting the other virtual router to the remote PCE and the other QKD domain (Alice). The intermediate area exposes the packet/optical network.

(ip routes, tables, and nating), and allows creation of IPsec sessions to external endpoints. It connects to the control plane using the protocols and extensions described in Sections IV and V. It also has access to the QKD domain (Bob), as it is necessary to extract keys from the emulated QKD system used to encrypt the IPsec channel. This domain contains the network controller container (PCE, in the figure) as well, connected to both virtual routers using the service management network. The right part is analogous to the left. It includes another virtual router with the same capabilities as its left-side homologous, a QKD domain to provide keys (Alice) and a local private domain with different hosts that can be accessed by end-users. Both networks are running inside two servers in Telefonica's laboratory. The QKD domains are composed by *IDQ* containers (emulating IDQuantique Clavis2 systems) and *ETSI proxy* containers, providing an intermediate interface (based on Ref. [19]) between applications and the QKD domain.

The intermediate network comprises the physical infrastructure, connecting the data plane via carrier-grade IP and optical devices. The IP routers are two MX240 routers and the optical layer with four FSP3000 from ADVA. Both locations are connected via a L3 VPN service as it is done for the corporate services in Telefonica. Despite the QKD domain that is emulated (with symmetric keys stored in Linux containers), the optical equipment used for this test has been previously demonstrated to be capable of supporting a quantum channel [12,13]. The service management network is used for the control plane communications, while the main management network is used for controlling the virtualization tool (creating the virtual networks). Any control/management channel is not protected during these tests, but they could also support the hybrid security scheme proposed in Ref. [4]. Any intermediate (control or data) network is assumed to be pre-configured before the service deployment.

VII. Implementation and Results

The implementation of the protocol extensions has been carried out in two different ways. Each protocol has been demonstrated in different emulation scenarios, showcasing the capabilities dissemination and the QKD key synchronization workflow. Furthermore, the MPLS solution (based on Telefonica's Netphony project [22]) has been also encapsulated inside a container and deployed via DockerNet, to demonstrate the integration of the proposed solution inside a virtual router. The virtual routers provide connectivity between two remote premises, giving access to remote web services and databases. They also encrypt the traffic between both endpoints, creating IPsec tunnels using QKD keys extracted from an emulated QKD link, all orchestrated from the control plane. It is important to note that any of the QKD networks described in Section II (direct link or dark fiber, trusted-relay network, with or without classical and quantum coexistence [10–13]) is a valid solution compatible with our demonstration, as long as QKD keys are generated at both ends of the QE service and made available for the virtual routers to secure the traffic.

Figure 7 presents the capture for the OpenFlow workflow, describing the necessary and extended messages to support the use case. It includes the features_reply from the device (containing the new capability), flow_mod from the controller (with the new action), barrier request and response, and the final statistics gathered via multipart_reply. Figure 8 (left) shows the QE capability inside the features' reply, advertised as a single bit, while the same figure (Fig. 8 right) exposes the new action and how the keyID (key_handle) is modified by the device. This ID changes from an unset value (0 × 00…00) to a valid ID, which is retrieved by the controller using the flow_stats. The retrieved ID will be consequently sent by the controller to the destination agent/device, which will take care of extracting the key from the emulated QKD system on its secure side.

The implementation for NETCONF has required the definition of a YANG model for the QE configuration. Figure 9 shows the proposed YANG model, including the parameters shown in Table I. Despite the traffic capture for the NETCONF use case not being shown (as the NETCONF messages are encrypted via SSH), Fig. 10 shows the XML transmitted (edit-config) and received (get-config) from and by the NETCONF manager. It also shows the new capability exposed inside the HELLO message (Fig. 10, bottom).

Initially the manager sends the QE information with the unset keyID. The NETCONF server (agent) receives this information, extracts the key and keyID pair from the QKD system, and stores the valid ID in the running data store. When the manager has finished transferring the configuration, it retrieves the keyID again from the devices using the get-config message. To finalize, the manager must transmit the valid keyID, together with the rest of parameters, to the destination device of the QE service.

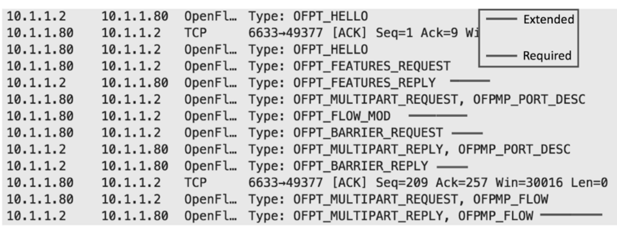

Fig. 7. OpenFlow messages used for setting up a QE service.

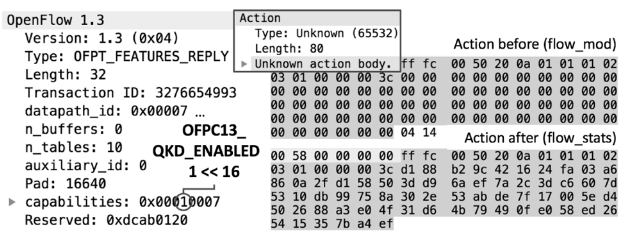

Fig. 8. (a) QE capability inside OpenFlow features_reply message and (b) new action before and after the QKD Key ID extraction.

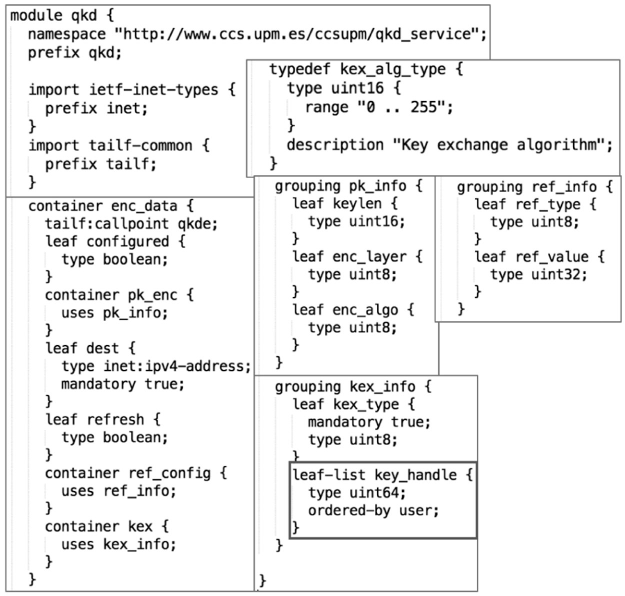

Fig. 9. YANG model for QE configuration.

The last demonstration includes the deployment of the QE service utilizing the QKD keys for the encryption of an IPsec tunnel. Figure 11 shows the messages exchanged between a PCE and both virtual routers across the service management network. The service request is initiated from the virtual router within the data center (point of presence)

Fig. 10. NETCONF edit (left), get (right) config XML information and capabilities within a HELLO message.

10.1.1.1	10.1.1.200	PCEP	172	Path Computation Request (PCReq)
10.1.1.200	10.1.1.1	PCEP	308	Path Computation Reply (PCRep)
11.2.1.1	11.2.1.2	UDP	73	Source port: 43840 Destination
11.2.1.2	11.2.1.1	UDP	106	Source port: 5323 Destination p
10.1.1.1	10.1.1.5	RSVP	308	PATH Message. SESSION: IPv4-LSP,
10.1.1.5	10.1.1.1	RSVP	144	RESV Message. SESSION: IPv4-LSP,

Fig. 11. Set of MPLS and key extraction messages exchanged during the QE service deployment.

```
METRIC object                                METRIC object
  Object Class: METRIC OBJECT (6)              Object Class: METRIC OBJECT (6)
  0001 .... = Object Type: 1                   0001 .... = Object Type: 1
▸ Flags                                      ▸ Flags
  Object Length: 12                            Object Length: 12
  Reserved: 0                                  Reserved: 0
▸ Flags: 0x80                                ▸ Flags: 0x80
  Type: Unknown (255)                          Type: Unknown (252)
  Metric Value: 32                             Metric Value: 1000

METRIC object                                METRIC object
  Object Class: METRIC OBJECT (6)              Object Class: METRIC OBJECT (6)
  0001 .... = Object Type: 1                   0001 .... = Object Type: 1
▸ Flags                                      ▸ Flags
  Object Length: 12                            Object Length: 12
  Reserved: 0                                  Reserved: 0
▸ Flags: 0x80                                ▸ Flags: 0x80
  Type: Unknown (254)                          Type: Unknown (253)
  Metric Value: 3                              Metric Value: 10
```

Fig. 12. Metrics sent inside the PCRequest message for the QE service.

premises, sending a path computation request to the PCE with the required metrics (Fig. 12). The response from the PCE already includes the new QE ERO subobject used to synchronize the QKD keys and create the secure path. When the response reaches the source node, it extracts the key and keyID pair from the emulated QKD system (messages 3 and 4). The keyID is subsequently included in the RSVP path message by changing the unset keyID within the ERO, as shown in Fig. 13. The configuration finalizes when the source node receives the RSVP RESV message back from the destination node.

Finally, Fig. 14 shows a traffic capture of one of the final services accessed using the deployed IPsec tunnel. This service is accessed from the local domain, going across the local virtual router, the intermediate network, and the data center virtual router, reaching a database (Redis server). The capture was taken inside the data center domain. Messages (a) and (b) show the request before and after

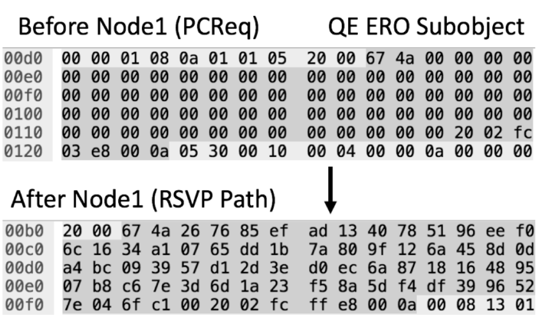

Fig. 13. QE ERO subobject change to signal the session ID.

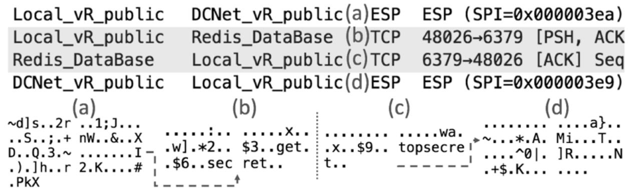

Fig. 14. Traffic capture (DC virtual router) of the encrypted and open data via IPsec.

passing through the virtual router; (c) and (d) show the response, opened before traversing the virtual router and encrypted afterwards.

VIII. Conclusion

QKD is a promising technology that brings an additional physical layer to secure current network infrastructures. This work proposes and demonstrates the required control plane workflows and protocol extensions for the integration of QKD keys into current network services. The key synchronization process (together with other encryption parameters) required for the subsequent encryption has been integrated in the major control plane protocols: MPLS, OpenFlow, and NETCONF/YANG. We also demonstrate the ease with which these new systems can be integrated in novel network paradigms by integrating the QKD-generated keys in IPsec sessions. These sessions are finally encapsulated inside VNFs automated via the aforementioned control plane extensions.

Acknowledgment

This work has been partially supported by project CVQuCo, Ministerio de Economía y Competitividad, Spain, Project No. TEC2015-70406-R, MINECO/FEDER UE, Comunidad Autónoma de Madrid, project Quantum Information Technologies Madrid, QUITEMAD+ S2013/ICE-2801, and by ACINO European H2020 project, http://www.acino.eu, grant number 645127.

References

[1] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008.

[2] N. Gisin, G. Ribordy, W. Tittel, and H. Zbinden, "Quantum cryptography," *Rev. Mod. Phys.*, vol. 74, no. 1, pp. 145–195, Mar. 2002.

[3] V. Martin, J. Martinez-Mateo, and M. Peev, "Quantum key distribution, introduction," in *Wiley Encyclopedia of Electrical and Electronics Engineering*, Wiley, 2017, pp. 1–17.

[4] A. Aguado, V. Lopez, J. Martinez-Mateo, T. Szyrkowiec, A. Autenrieth, M. Peev, D. Lopez, and V. Martin, "Hybrid conventional and quantum security for software defined and virtualized networks," *J. Opt. Commun. Netw.*, vol. 9, no. 10, pp. 819–825, Oct. 2017.

[5] A. Aguado, V. Lopez, J. Martinez-Mateo, M. Peev, D. Lopez, and V. Martin, "GMPLS network control plane enabling quantum encryption in end-to-end services," in *Int. Conf. Optical Network Design and Modelling (Best Paper Award)*, 2017.

[6] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol label switching architecture," IETF RFC 3031, 2001.

[7] R. Enns, M. Bjorklund, J. Schoenwaelder, and A. Bierman, "Network configuration protocol (netconf)," IETF RFC 6241, 2011.

[8] Information Assurance Directorate, National Security Agency/Central Security Service, "Commercial national security algorithm suite and quantum computing FAQ," MFQ U/OO/815099-15, Jan. 2017.

[9] H.-J. Briegel, W. Dur, J. Cirac, and P. Zoller, "Quantum repeaters: the role of imperfect local operations in quantum communication," *Phys. Rev. Lett.*, vol. 81, no. 26, pp. 5932–5935, 1998.

[10] M. Peev, C. Pacher, R. Alléaume, C. Barreiro, J. Bouda, W. Boxleitner, T. Debuisschert, E. Diamanti, M. Dianati, J. F. Dynes, S. Fasel, S. Fossier, M. Fürst, J.-D. Gautier, O. Gay, N. Gisin, P. Grangier, A. Happe, Y. Hasani, M. Hentschel, H. Hübel, G. Humer, T. Länger, M. Legré, R. Lieger, J. Lodewyck, T. Lorünser, N. Lütkenhaus, A. Marhold, T. Matyus, O. Maurhart, L. Monat, S. Nauerth, J.-B. Page, A. Poppe, E. Querasser, G. Ribordy, S. Robyr, L. Salvail, A. W. Sharpe, A. J. Shields, D. Stucki, M. Suda, C. Tamas, T. Themel, R. T. Thew, Y. Thoma, A. Treiber, P. Trinkler, R. Tualle-Brouri, F. Vannel, N. Walenta, H. Weier, H. Weinfurter, I. Wimberger, Z. L. Yuan, H. Zbinden, and A. Zeilinger, "The SECOQC quantum key distribution network in Vienna," *New J. Phys.*, vol. 11, no. 7, p. 075001, 2009.

[11] M. Sasaki, M. Fujiwara, H. Ishizuka, W. Klaus, K. Wakui, M. Takeoka, S. Miki, T. Yamashita, Z. Wang, A. Tanaka, K. Yoshino, Y. Nambu, S. Takahashi, A. Tajima, A. Tomita, T. Domeki, T. Hasegawa, Y. Sakai, H. Kobayashi, T. Asai, K. Shimizu, T. Tokura, T. Tsurumaru, M. Matsui, T. Honjo, K. Tamaki, H. Takesue, Y. Tokura, J. F. Dynes, A. R. Dixon, A. W. Sharpe, Z. L. Yuan, A. J. Shields, S. Uchikoga, M. Legré, S. Robyr, P. Trinkler, L. Monat, J.-B. Page, G. Ribordy, A. Poppe, A. Allacher, O. Maurhart, T. Länger, M. Peev, and A. Zeilinger, "Field test of quantum key distribution in the Tokyo QKD network," *Opt. Express*, vol. 19, no. 11, pp. 10387–10409, 2011.

[12] A. Ciurana, J. Martinez-Mateo, M. Peev, A. Poppe, N. Walenta, H. Zbinden, and V. Martin, "Quantum metropolitan optical network based on wavelength division multiplexing," *Opt. Express*, vol. 22, no. 2, pp. 1576–1593, 2014.

[13] D. Lancho, J. Martinez, D. Elkouss, M. Soto, and V. Martin, "QKD in standard optical telecommunications networks," in *Quantum Communication and Quantum Networking*, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol. 36, Springer Berlin Heidelberg, 2010, pp. 142–149.

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

[15] "Network functions virtualisation (NFV); architectural framework," in ETSI GS NFV 002 V1.2.1, 2014–12, 2013.

[16] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network function virtualization: challenges and opportunities for innovations," *IEEE Commun. Mag.*, vol. 53, no. 2, pp. 90–97, Feb. 2015.

[17] A. Farrel, J.-P. Vasseur, and J. Ash, "A path computation element (PCE)-based architecture," IETF RFC 4655, 2006.

[18] D. McDonald, C. Metz, and B. Phan, "PF_KEY key management API, version 2," IETF RFC 2367, 1998.

[19] "Quantum key distribution (QKD); application interface," ETSI GS QKD 004 V1.1.1, 2010–12.

[20] S. Marksteiner and O. Maurhart, "A protocol for synchronizing quantum-derived keys in IPsec and its implementation," in *9th Int. Conf. Quantum, Nano/Bio, and Micro Technologies*, 2015, pp. 35–40.

[21] A. Lindem, N. Shen, J. Vasseur, R. Aggarwal, and S. Shaffer, "Extensions to OSPF for advertising optional router capabilities," IETF RFC 7770, 2016.

[22] "Java library of networking protocols: PCEP, RSVP-TE, OSPF, BGP-LS," [Online]. Available: https://github.com/telefonicaid/netphony-network-protocols, accessed Jan. 31, 2017.