An Enhanced Virtualized Control and Key Management Model for QKD Networks

Blanca Lopez [1,2,*], **Ivan Vidal** [2], **Francisco Valera** [2], **Diego R. Lopez** [3]
[1]IMDEA Networks Institute, Avda. del Mar Mediterráneo, 22, 28918 Leganés, Madrid, Spain
[2]Telematic Engineering Department, Universidad Carlos III de Madrid, Avda. Universidad, 30, 28911 Leganés, Madrid, Spain
[3]Telefónica, Ronda de la Comunicación, Fuencarral-El Pardo, 28050 Madrid, Spain
[*]Correspondence: blanca.lopez@imdea.org (B.L.)
https://doi.org/10.1109/MNET.2025.3538752

ABSTRACT

The advent of softwarization and disaggregated architectures has transformed modern communication networks, sparking innovation by separating network functionalities from the underlying hardware. Following this trend, in future quantum networks, the virtualization and softwarization of critical components will be essential to achieve global interoperability and enhance adaptability. Virtualization allows for the abstraction of hardware resources, making it easier to manage and scale networks dynamically, while softwarization promotes flexibility, reconfigurability, and interoperability by enabling the use of standardized, software-defined protocols that can be easily updated and modified to work across diverse and evolving network environments. In this paper, we introduce and examine an operational model for QKD networks that leverages the virtualization of control and key management functionalities. We detail its elements and procedures to optimize QKD services. The design paves the way for the integration of quantum networks with functions already established in modern mobile networks while adhering to established QKD network standards. The proposed operational model has been validated at the 5G Telefonica Open Network Innovation Centre (5TONIC) using an environment that deploys digital twins of QKD networks.

1 INTRODUCTION

As quantum computing technology advances, the need to shift from traditional cryptographic algorithms becomes more urgent. Quantum Key Distribution (QKD) represents the only theoretically inherently secure strategy against any future computational attacks, ensuring security based on the principles of quantum mechanics rather than computational complexity. Nevertheless, constructing a quantum network poses significant challenges. While the landscape has dramatically transformed in recent years, marked by medium-sized deployments of functional QKD networks and the development of standards that guide stakeholders across this novel terrain, substantial progress is still required. Integrating QKD technology into current telecommunication network infrastructures remains a distant yet critical goal, that needs continued innovation, commitment, and collaborative efforts across the global tech community.

One of the challenges in integrating QKD mechanisms into network infrastructures is the incompatibility between the traditionally rigid structures of QKD networks and the dynamic demands of modern communication systems. Currently, QKD component manufacturers typically offer solutions that are tightly coupled to proprietary hardware and have small room for tailored mechanisms or on-demand parameterization. These “black boxes" are primarily hardware-dependent, which is an essential starting point given the nature of quantum communications. However, extending this approach across all network functionalities significantly hinders flexibility and scalability. By decoupling the management and control functionalities from hardware devices that provide the QKD protocol basic mechanisms, a software-driven approach allows these tasks to be implemented using versatile software components. Then, virtualization can be employed to execute these softwarized functions in various locations, such as off-the-shelf computers colocated with the QKD modules, enabling the flexible addition of new mechanisms and updates to the existing infrastructure without replacing the entire hardware setup. This shift enhances adaptability to evolving network requirements and aligns with broader trends toward virtualization and software-defined networking (SDN), paving the way for more robust, scalable, and future-proof quantum networks.

SDN, the primary implementation of the softwarization concept, guarantees higher degrees of flexibility and reconfigurability by allowing network functions to be easily modified and updated through software changes rather than requiring physical hardware adjustments [1]. Meanwhile, virtualization allows the abstraction of hardware devices by creating virtual instances, improving scalability, resource efficiency, and enabling the dynamic allocation of network softwarized functions across different locations [1].

Since QKD networks are still an emerging technology, these two paradigms are particularly beneficial. As technology and standards evolve, a virtualized layer for softwarized network functions significantly aids adaptation to these changes, ensuring pliability and responsiveness for seamless updates, and incorporating new mechanisms without extensive hardware modifications.

Additionally, it is essential to integrate functions related to user authentication and access control in future quantum networks. These functions are already standard in the control planes of advanced telecommunications networks such as 5G, specifically the authentication server function (AUSF) and the access and mobility management function (AMF), among others. Integrating these mechanisms into QKD networks will enhance security measures and ensure that quantum networks can seamlessly interface with existing and future telecommunications infrastructure.

Currently, technology does not allow direct end-to-end quantum key exchange between distant nodes. Instead, key distribution must rely on trusted intermediary nodes to relay the key hop-by-hop. A key management layer where the actions of all nodes are coordinated is needed to ensure secure and efficient key distribution. In [2] we presented a model that decouples the key management plane from QKD module hardware, moving beyond the conventional monolithic QKD node paradigm. This separation enables

more flexible, scalable QKD networks, enhancing adaptability to network changes and function updates. The model supports the integration of hybrid cryptographic strategies, combining QKD and post-quantum algorithms to ensure security and performance even when QKD rates fall short of meeting network demands. Such a hybrid model can run transparently across the network, dynamically selecting the most suitable cryptographic method. Additionally, it allows real-time performance monitoring, data-driven control and management, and tailored access and admission mechanisms [2].

Building on these foundations, in this paper we present a proposal for a new architectural model of QKD networks, featuring a distinct key management plane and introducing new control functionalities. We propose a disaggregated virtualized key management plane that increases network resilience while complying with standards. It offers a high degree of flexibility: since it is built on softwarized elements, these can be adapted according to the needs of the network and the evolution of technologies. We analyze the interactions within the key management plane and with the control plane, where we add several innovative virtualized components that enable mechanisms similar to those in mobile networks. The alignment with existing advanced network structures lays the groundwork for integrating QKD networks into mainstream telecommunication infrastructures. This integration will enable a dynamic and secure interplay between quantum and classical network elements, fostering a cohesive ecosystem where QKD technology seamlessly merges with conventional network architectures. Additionally, the virtualization approach supports dynamic scaling by automatically replacing failed elements and adjusting resource allocation based on real-time demand, ensuring consistent performance and efficient resource use even under fluctuating conditions.

This paper is structured as follows: Section 2 analyzes the relevant standards related to our work. Section 3 presents the proposed operational model, detailing its components and operational procedures. In Section 4, we demonstrate the functional validation of the model. Finally, Section 5 concludes the paper and outlines potential directions for future research.

2 STANDARIZATION LANDSCAPE

In recent years, major global organizations such as ETSI, ITU, and IETF have been developing a standards framework to advance quantum technologies and facilitate their early integration into existing communication infrastructures. We will briefly outline the most relevant standards related to our work, focusing on those discussing the architecture of a QKD network [3, 4], and those referring to the use of SDN technologies in the control functions of said networks [5–7].

The models proposed by ITU and ETSI are fairly aligned, except for the nomenclature and the mapping of certain ancillary functionalities within the architecture. ITU defines an architecture divided into six different layers: service, control, management, user management, key management, and quantum. It proposes two architectural choices: a distributed model, including a controller instance at each of the QKD

nodes in the network; and a centralized model, with a general controller in charge of all the nodes [3]. Consistent with this architecture, Y-3805 [5] specifies using SDN technologies in the control plane to improve network control and management functions. This document details different operational procedures of the SDN control, such as key generation or management monitoring.

ETSI documents focus on interfaces to ensure interoperability at the levels where it can be achieved. In its standards, the QKD network nodes are composed of quantum modules, capable of generating keys, with an associated key manager, an SDN agent capable of communicating with the network controller, and applications that request cryptographic material. Additionally, the QKD network also has an SDN controller, and there may be an orchestrator that serves as a link between several QKD networks or between a QKD network and a classical network [6, 7]. ETSI has developed two documents defining different Application Programming Interfaces (APIs) designed to enable communication between key managers and applications on a node [4, 8]. These APIs allow applications to obtain secure cryptographic material. The ETSI 014 document describes a simpler API intended to enable interoperability, that can be built from the more general API described in ETSI 004.

The research arm of the IETF, the IRTF, has created a Quantum Internet research group, that is working on documents intended to go beyond the QKD networks and apply the experience accumulated with them, discussing possible architectures and applications for the future quantum Internet [9, 10].

These documents aim at enhancing the flexibility of quantum networks, particularly through the use of SDN technologies. However, while providing a solid starting point, they do not fully exploit the potential of advanced softwarization and function disaggregation. These types of technology, already embedded in our telecommunications networks, could make the inclusion of quantum cryptography services in existing infrastructures easier. Another trend that is well-established in modern telecommunications networks and for which we advocate is virtualization. It enables rapid and efficient deployment of new services and network functions, as well as simplifying maintenance tasks and recovery strategies, since it is much easier to scale, migrate, adapt, and restore virtualized functions than those directly provided by hardware.

For QKD to become a common service in our networks, a natural approach is to align with these trends and design the architecture of QKD networks and their functions accordingly.

3 MODEL ANALYSIS

Our model primarily focuses on virtualizing and softwarizing the key management layer of QKD networks, detaching it from the underlying hardware. By separating key management from most of their physical constraints while maintaining standard mechanisms for communication with QKD modules, QKD networks gain agility and flexibility to adapt to future developments and standards, without replacing the hardware that supports them. Additionally, this separation enables new functionalities, such as real-time

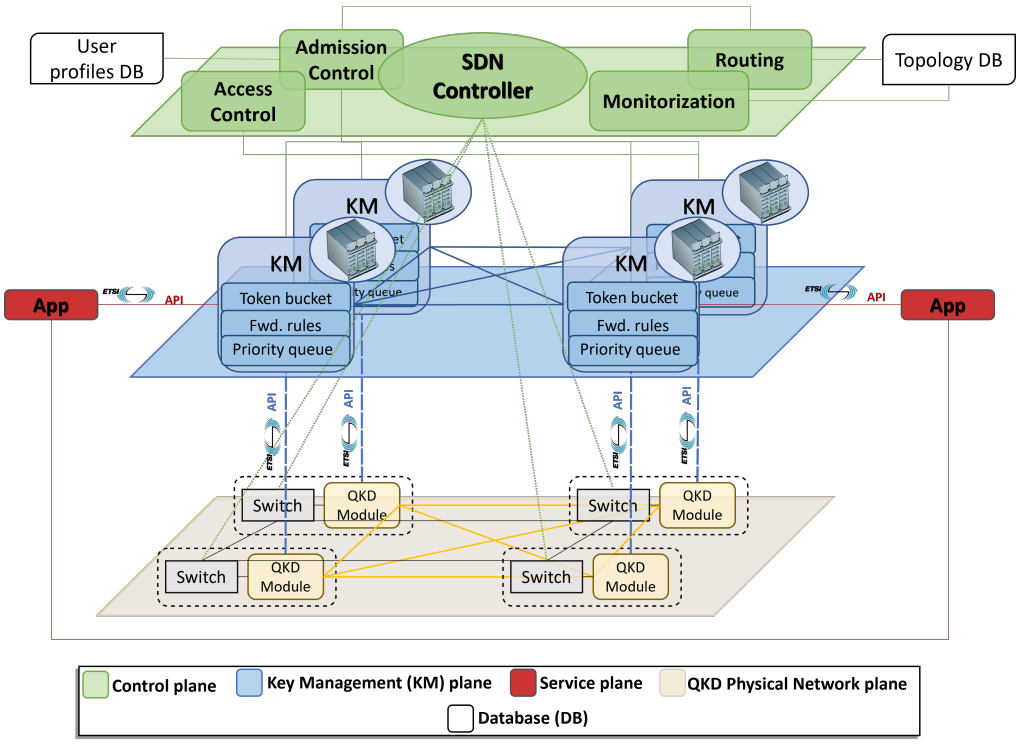


Figure 1: Architecture and elements of the proposed model. Planes can be differentiated by color coding.

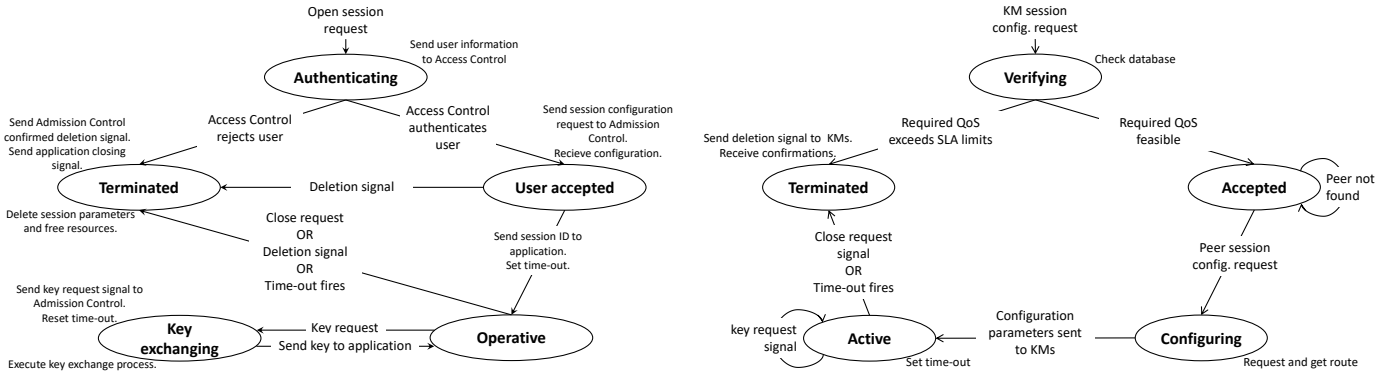
performance monitoring, tailored access and admission control mechanisms, or continuous key provisioning even in the event of a quantum infrastructure outage [2].

In this section, we analyze the model operational aspects, reviewing the new elements introduced both in the control and key management plane to ensure functionality and to benefit from the flexibility and programmability that the model offers. Furthermore, we discuss its alignment with existing standards, specifically, how the proposed model conforms to ETSI 004and therefore it is extendable to use ETSI 014, and ETSI 015, which describes the interface to the QKD network SDN controller.

3.1 Model elements

An outline of the proposed model with its most relevant components can be found in Fig. 1. In the figure, four distinct planes can be identified: the service plane, which refers to the applications that request cryptographic material, the control plane, which includes those elements with network management and control functions, the key management plane, a virtual plane responsible for the key management functions that were initially embedded in the quantum hardware but are being increasingly addressed through softwarized approaches [11, 12], and a physical network plane that comprises all quantum and optical devices. Additionally, the model relies on two databases. It can be seen how the model strongly follows the architecture defined by standards, almost mapping the layers defined in [3, 4].

Virtualizing the key management functions allows the inclusion of different customized mechanisms. Our model initially considers three basic elements in each key manager, providing fundamental functionality suitable to be extended by other software modules. The priority parameter defined in the ETSI API



(**a**) Key manager cycle.

(**b**) Admission control element cycle.

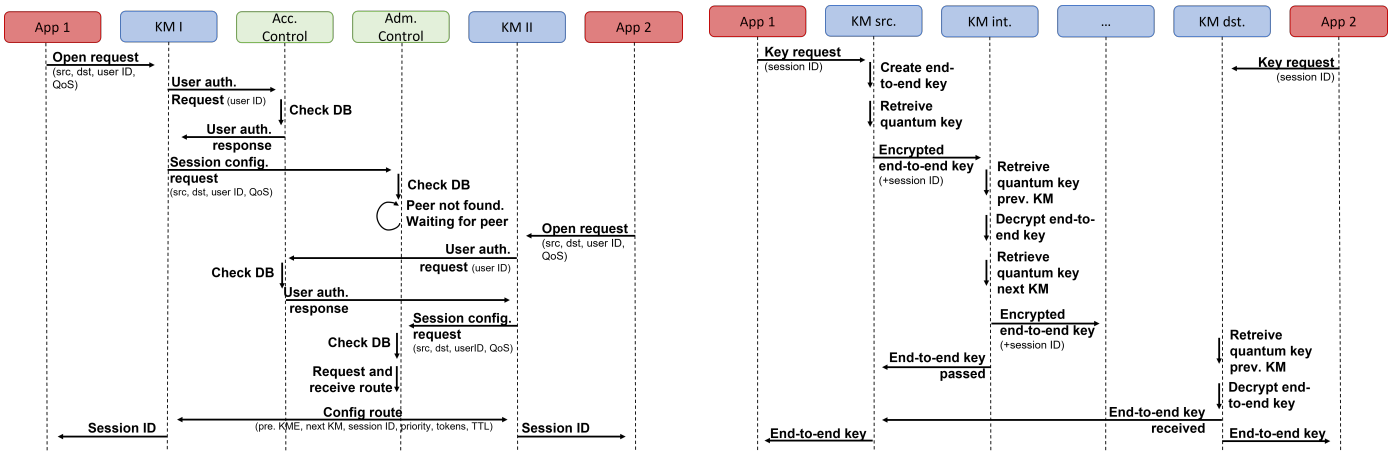
Figure 2: Session cycles. The states are represented in ellipses, with their actions attached, and the transitions between states are located on the arrows.

004 [4] is implemented through priority queues present in each key manager and configured by the admission control element. They sort application requests according to user and/or session critical levels for that particular applicationnote that a user may have multiple applications, more or less critical. The applied policy is entirely controllable by the network service provider and by any Service Level Agreement (SLA) established with network users.

In current QKD networks, if two applications request cryptographic material from their respective QKD modules that are not directly connected, the network must rely on a trusted-node scheme. This involves transmitting an end-to-end key across multiple nodes along the path. Key managers rely on a table of forwarding rules, according to which, given the ID of a session, i.e. a period during which the application and its peer can request symmetric cryptographic material secured by quantum mechanisms, the key manager knows what the next hop in the path of that particular session is. These tables are set by the control plane elements and can be dynamically updated whenever needed. For example, the table can be adjusted whenever an application starts or ends a session or when a physical or virtual route needs to be changed [6]. This approach helps to address the limitations of the traditional trusted-node scheme by enhancing flexibility and responsiveness.

Lastly, key managers implement a token bucket for each application session. A token bucket is a well-known mechanism to control the request rate by limiting it to an agreed-upon value at the Quality of Service (QoS) level. This mechanism involves two main parameters: the bucket size, which determines the maximum burst size allowed, and the token rate, which controls the average allowed data rate. These parameters can be dynamically configured based on the QoS requested by the application and the SLA.

We consider four different elements interacting through the control plane apart from the SDN controller. On the one hand, the Access Control component is responsible for enforcing network access policies, ensuring that no unauthorized user or process enters the network. In the ETSI API 014 specification [8], TLS is recommended as the authentication mechanism, which can guarantee quantum-proof security by in-



(**a**) First part. Configuration of the sessions.

(**b**) Second part. Key exchange.

Figure 3: Flow diagram of a symmetric key exchange between two non-adjacent applications.

tegrating post-quantum algorithms. On the other hand, the Admission Control module manages configuration parameters for the new sessions opened by applications, verifying that they are granted adequate QoS. To do this, it has access to a database with the profiles of the different network users, allowing it to ascertain the priority to be given to the requests and the QoS allowed, and communicates with the Routing Element to obtain a suitable path for each session and QoS.

The Routing Element has access to the topology database which includes the available physical links and their performance metrics, such as key rate, quantum bit error rate (QBER), number of sessions using a certain link, and any other magnitudes that may help to compute a suitable path for a given session request. The Routing Element keeps a graph using the information provided by the SDN controller and adjusts the weights of the links according to the information gathered in the database. This information is collected by the Monitorization Element, responsible for making performance tests over the physical links and collecting statistics on the QKD modules to keep data updated, rapidly warning about any failure or possible attack on the quantum physical network.

Finally, the SDN controller acts as the central piece of the control plane, distributing the initial information about the topology of the network and any changes that may occur, registering new physical or virtual links as described in [6].

3.2 Operational procedures

The design of the flow of actions in (and among) the different elements that make up the network is a delicate and crucial task for applying the operational model.

We will first review the actions of the session cycle both at the key manager receiving an open request (Fig. 2a) and at the admission control element (Fig. 2b). These cycles represent the flow of actions for one session, i.e., for each session an application starts on the network, an instance of this cycle begins. The network elements can manage several instances in different states simultaneously.

Next, we will examine the complete flow diagram of a symmetric key exchange. We will consider the general case where a pair of applications make requests to two non-adjacent key managers. In this scenario, there is a path of intermediate key managers connecting them (Fig. 3). The flow diagram for the case where the QKD modules of the endpoint key managers are physically connected is simpler and can be derived from this.

As shown in Fig. 2a, once the key manager receives an open request from an application, which includes information identifying the user who owns the application, it passes this information to the access control element and waits in the authenticating state. If the access control component verifies and authenticates the user, the session instance switches to the user accepted state.

In this state, the key manager sends a request to the admission control module for session configuration, sharing information about the user and the QoS required by the application. The key manager can either receive a deletion signal moving on to the terminated state, or receive a route configuration. In the latter case, the key manager returns a session ID to the application and configures a timer setting how long the session can remain open without receiving any key requests. The session is now in the operative state.

In the operative state, the key manager is ready to handle any key request from the application using the specific session ID; whenever the application sends a cryptographic material request it enters the key exchanging state. The key manager sends a signal to the admission control element, resets its timer, and executes the key exchange process. Once the key is sent to the application, it returns to the operative state.

Three events can end a session in the operative state: a close request from the application, a deletion signal from the admission control module, or the expiration of the key manager instance timer. If any of these triggers occur, the session instance passes to the terminated state. In this state, the key manager proceeds to delete the session configuration parameters and frees the resources that were reserved to comply with the session-required QoS. Finally, the key manager confirms the session deletion to the admission control element and sends a closing signal to the application.

The admission control element cycle begins whenever a configuration request arrives from a key manager. The admission control element checks its database to ensure that the QoS requested by the application is appropriate. If it exceeds the SLA limits, the instance passes to the terminated state and the admission control module sends a deletion signal to the key manager. On the contrary, if the QoS is suitable, the session instance advances to the accepted state.

In this state, and before configuring the session route that would reserve a certain amount of network resources, the admission control module waits for the registration of the session peer, as described in [6]. For a pair of applications that wish to start sharing cryptographic material to be able to make key requests, it is necessary for both to have registered with their respective key managers and obtained an

identical session ID. As long as one of the two applications in the pair has not started the session opening process, the admission control element will not establish the path, to avoid temporarily spending resources on processes that may never complete.

Once both peers have requested session configuration, the admission control module communicates with the routing element to obtain an appropriate path. It sends the configuration parameters to all key managers in said path.

From that moment on, the admission control instance is in the active state and sets a timer for the session, restarted each time the admission control element receives a key request signal for that session, meaning that the session remains active.

If the admission control module receives a close request signal from one of the peer key managersthose who requested the session configuration on behalf of the applicationsor the timer expires, it enters the terminated state. Then, the admission control element sends a deletion signal to all key managers working on the session, i.e. all key managers in the route, and receives their confirmations.

Fig. 3a shows the sequence of actions performed by the different elements of the network for the first part of a key exchange between two applications whose respective key managers are non-adjacent. This flow diagram corresponds to the session state from authenticating to operative.

The first application sends an open request to the correspondent key manager. This request includes the source and destination of the keys, the identity of the user who owns the application, and the QoS, as described in [4]. Once the access control element authenticates the user, the key manager shares the request information with the admission control component, which verifies the data and waits for the application peer to make its open request. The second application repeats this process, and when the admission control module receives the request to set up the session from the second key manager, it contacts the routing element to retrieve a suitable path for the application requests. Then, the admission control element distributes the configuration parameters among all the involved key managers. These parameters include the session ID, the priority of the request, the number of tokens to be assigned to the session, the time-out until the session is considered inactive and must be deleted, and the previous and next key manager in the route for each specific key manager. Note that the key managers are unaware of the full path, and only know the information needed to build their forwarding tables. The key managers connected to the applications return them the session ID and the session configuration is finished.

As soon as the applications have the session ID they can start making requests to their key managers. Since we suppose a case where the source and the destination key managers are non-adjacent, the cryptographic material exchange must follow a trusted node relay scheme, outlined in Fig. 3b. When the applications request a key to the endpoint key managers, the source key manager creates an end-to-end key. The end-to-end key is sent to the next key manager in the path using the one-time pad method and

a quantum key shared by the key manager holding the end-to-end key and the next key manager. This process is repeated until the end-to-end key arrives at the destination key manager, which sends a signal to the source, and both endpoint key managers proceed to return the end-to-end key to their applications. The symmetric key exchange has been completed, and the two applications now share some secure cryptographic material to be used for any desired purpose.

4 PRACTICAL PROOF OF CONCEPT

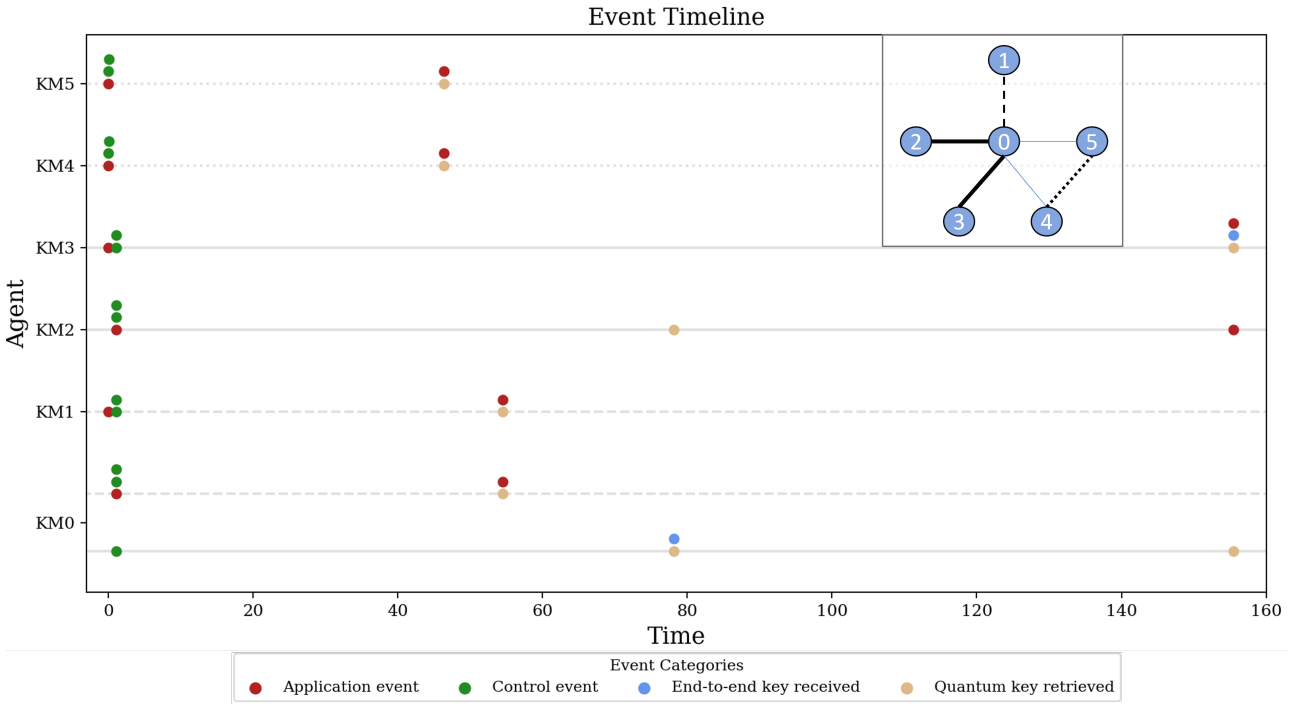


Figure 4: Event stream of key exchange processes. The solid line corresponds to the trusted-node scheme followed in the key exchange between 2 and 3, the dashed line corresponds to the key exchange between 0 and 1, and the dotted line corresponds to the key exchange between 4 and 5.

Using the Quditto platform [13], a proof of concept was designed to test the model and its feasibility. Quditto is a software that allows the generation of QKD network digital twins, and to build, using virtual machines and containers, all kinds of architectures, processes, and mechanisms upon them.

We deployed a QKD network of six nodes connected in a star topology, with an extra link connecting two of the vertices. To each of them, we associated a key manager, building the network depicted in Fig. 4. Additionally, we built the control plane with an access control component, an admission control module, a routing element, and a centralized controller. Each element was deployed in separate virtualization containers using Docker images, connected virtually following the proposed model. The experiment was carried out using a computer with 12 cores, 16 logical processors, and 16GB of RAM.

In the experiment, there was no monitoring element, so in the graph generated by the routing element, every link had the same weight.

The admission control element used a publish/subscribe protocol to communicate with key managers,

using a specific topic for each key manager in the network. This presents certain advantages of scalability, dynamism, and flexibility, as topics can be created and discarded depending on the functionalities of the network and the necessary communication patterns.

Besides the network topology, Fig. 4 shows all the events that occur when applications make key requests. In the test, parallel key requests were made to all nodes, represented in the figure by the first red dot appearing on the lines of the different key managers. Symmetric key requests were made between nodes 0 and 1, 2 and 3, and 4 and 5. The first and last pair, being directly connected, exchanged a purely quantum-emulated key. In contrast, for nodes 2 and 3, separated by an intermediate node, a trusted node scheme was followed.

The same order of events is followed in all processes. First, the arrival of the request, marked in red. Then, two events come from the control plane, one authenticating the application, and the other configuring the key managers involved in the path. The only exception is the trusted-node configuration of KM0 (at the bottom of Fig. 4), which implies it only receives a configuration event, given its role. The QKD protocol takes some time to converge, in our case about one minute, until the key managers get enough cryptographic material. This event corresponds to the yellow marker. The reader is warned that the time required to complete the QKD protocol is linked to the computational resources that perform the emulated key exchange, and therefore differs from the timing of real quantum devices.

In the case of the two keys exchanged by adjacent nodes, 0-1 and 4-5, once the key is obtained, it is returned to the applications that made the requests, which is represented by the red dot marking the end of the processes. Meanwhile, the process to obtain a key between nodes 2 and 3 continues, and KM0, the intermediate node, receives the end-to-end key encrypted via a one-time pad scheme with the quantum cryptographic material it shares with KM2. This event is represented in the figure with a blue dot. Once nodes 0 and 3 generate a quantum key, the latter receives the end-to-end keyblue marker and both applications get a responsered markers.

The proof of concept we carried out demonstrates the feasibility of the proposed model, showing its dynamism in configuring different routes within a given topology. Moreover, the flexibility of the model allows the integration of several virtualized modules with different functionalities, highlighting its adaptability to meet different needs.

5 CONCLUSION AND FUTURE WORK

Taking inspiration from the evolution of modern telecommunication networks, this paper analyzes a QKD network design that virtualizes the key management layer. This operational model not only offers benefits like continuous key provisioning and real-time monitoring [2], but also facilitates the integration of established functions from next-generation mobile networks, such as AUSF or AMF, into future quantum networks.

In line with this paper, we plan to employ physical QKD modules for field testing, building a more complex network that allows us to test our model thoroughly. This involves virtualizing the key management functions and building in software all the elements of the control plane to demonstrate that the model is fully adaptable and can be deployed using commercially available QKD components. Additionally, we plan to use our virtualized key manager model to achieve interoperability both between physical QKD devices from different manufacturers and between physical and emulated QKD modules, enabling hybrid deployments using Quditto.

ACKNOWLEDGMENT

This work has been partially supported by the project MadQuantum-CM, funded by the Regional Government of Madrid, the Spanish Government through the Recovery, Transformation and Resilience Plan, and the European Union through the NextGeneration EU funds; the EU Horizon Europe projects Quantum-oriented Update to Browsers and Infrastructures for the PQ transition (QUBIP), under grant 101119746, and Quantum Security Networks Partnership (QSNP), under grant 101114043.

REFERENCES

[1] M. Condoluci, T. Mahmoodi, Softwarization and virtualization in 5G mobile networks: Benefits, trends and challenges, Computer Networks, Volume 146, 2018, Pages 65-84, ISSN 1389-1286

[2] B.Lopez, I. Vidal, F.Valera, D. Lopez, A. Pastor, Unleashing Flexibility and Interoperability in QKD Networks: The Power of Softwarized Architectures. In Proceedings of the 2024 International Conference on Quantum Communications, Networking, and Computing (QCNC) pp. 216–220.

[3] ITU-T, *Quantum key distribution networks Functional architecture*; ITU-T Y.3802 (2020) Amd. 1 (11/2023); ITU: Geneva, Switzerland, 2023.

[4] ETSI, *Quantum Key Distribution (QKD); Application Interface*; ETSI GS QKD 004 V2.1.1 (2020-08); ETSI: Sophia Antipolis, France, 2020.

[5] ITU-T, *Quantum key distribution networks Software-defined networking control*; ITU-T Y.3805 (2021) Amd. 1 (11/2023); ITU: Geneva, Switzerland, 2023.

[6] ETSI, *Quantum Key Distribution (QKD); Control Interface for Software Defined Networks*; ETSI GS QKD 015 V2.1.1 (2022-04); ETSI: Sophia Antipolis, France, 2022.

[7] ETSI, *Quantum Key Distribution (QKD); Orchestration Interface for Software Defined Networks*; ETSI GS QKD 018 V1.1.1 (2022-04); ETSI: Sophia Antipolis, France, 2022.

[8] ETSI, *Quantum Key Distribution (QKD); Protocol and Data Format of REST-Based Key Delivery API*; ETSI: Sophia Antipolis, France, 2019.

[9] D. Lopez, V. Martin, B. Lopez, and L.M. Contreras, *A Multiplane Architecture Proposal for the Quantum Internet*. Internet Engineering Task Force, 2023. Work in Progress. Available online: https://datatracker.ietf.org/doc/draft-lopez-qirg-qi-multiplane-arch/ (Accessed on 14 March 2024).

[10] C. Wang, A. Rahman, R. Li, M. Aelmans, and K. Chakraborty, *Application Scenarios for the Quantum Internet*. Internet Engineering Task Force, 2024. Available online: https://datatracker.ietf.org/doc/rfc9583/ (Accessed on 31 July 2024).

[11] M. Wang *et al.*, *A Segment-Based Multipath Distribution Method in Partially-Trusted Relay Quantum Networks*. IEEE Communications Magazine, vol. 61, no. 12, pp. 184-190, December 2023, doi: 10.1109/MCOM.010.2200672.

[12] A. Tajima *et al.*, *Quantum key distribution network for multiple applications*. Quantum Science and Technology, vol. 2, no 3, 2017, doi: 10.1088/2058-9565/aa7154.

[13] Raul Martin, Blanca Lopez, Ivan Vidal, Francisco Valera, and Borja Nogales *Service for Deploying Digital Twins of QKD Networks*. Appl. Sci. 2024, 14, 1018. https://doi.org/10.3390/app14031018

www.quditto.io