Quantum Abstraction Interface: facilitating integration of QKD devices in SDN networks

Ruben B. Mendez[1], Juan P. Brito[1], Rafael J. Vicente[1], A. Aguado[1], Antonio Pastor[2], Diego Lopez[2], V. Martin[1], Victor Lopez[2]

[1]Center for Computational Simulation, Universidad Politécnica de Madrid, Madrid, Spain
[2]Telefonica gCTIO/I+D, Madrid, Spain

ABSTRACT
This work presents and implements Quantum Abstraction Interface (QuAI). QuAI is integrated in an SDN agent to enable the vendor agnostic configuration of QKD devices. The work demonstrates the framework to provide a flexible procedure to integrate QKD devices in production networks.

Keywords: Quantum Key Distribution, Software-Defined Networking, Application Program Interface

1. INTRODUCTION

Quantum communications is the most mature area of the new generation of quantum technologies. In particular, Quantum Key Distribution (QKD) [1,2] is at the point in which devices are already available in the market. This technology allows the creation of very high security symmetric keys at the ends of a quantum channel -a channel that allows the transmission of quantum signals-. The security of these keys is granted by the properties of the nature and do not rely on any computational assumptions, as is common in the public key cryptosystems. The required quantum channel is usually implemented using an optical fiber. The fact that quantum communications require the transmission and detection of single quantum signals implies that the co-propagation with classical signal, with about 108 photons per pulse, is very difficult. Consequently, the usual choice has been to use dark fiber for the quantum channel. This is very expensive and, from a market perspective, is a no-go, since it requires the building of a special, separated network, for the quantum communications alone.

However, mixing quantum and classical signals is possible under controlled conditions. The typical optical networks, designed with other parameters in mind, rarely allowed to fulfill the requirements to share the optical network between classical and quantum signals. This has started to change with the advent of the Software Defined Networking (SDN) paradigm. SDN has programmability at its base and provide a level of flexibility that allows to dynamically control the network such that the requirements needed to transmit quantum-level signals can be fulfilled. To do this, it is not only necessary to have a flexible SDN, but also to provide the QKD systems with some capability such that they can be controlled from the network.

The SDN scheme permits for the first time to produce a real quantum-classical integration as opposed to an ad-hoc one, in which every link has to be specifically adapted for quantum use. This approach has been recently demonstrated [3] and several, industry-driven, use cases have been implemented. However, the devices have been designed for a simple point-to-point, directly connected, link. This, and the increasing number of companies building QKD devices, make difficult to fulfill in a general way the need to drive the QKD systems from the network.

In this paper we present a way to address this problem in the form of a hardware abstraction layer that we call Quantum Abstraction Interface (QuAI). QuAI allows that any manufacturer can easily export the network capabilities of their QKD devices to an SDN-like network. It uses standard tools and mechanisms to accomplish its objectives. The availability of QuAI will facilitate to QKD manufacturers to integrate their devices in the communications network, a must for this technology to find widespread use.

2. QUANTUM ABSTRACTION INTERFACE

The Quantum Abstraction Interface describes a minimal interface to setup and communicate with a QKD device. Note that this Abstraction Interface is running inside the Agent, which could be executed inside the QKD device or in a separated process in another device, following a disaggregated approximation, leaving how this interface interact with the QKD device implementation dependent.

Fig. 1. QuAI Application Interface

```
QuAI Interface {
        REGISTER(deviceId, URI) return boolean;
        SET_ATTRIBUTE(deviceId, attribute, value) return boolean;
        GET_ATTRIBUTE(deviceId, attribute) return value;
        REMOVE(deviceId) return boolean;
}
```

The QuAI interface is designed to be simple yet powerful and open interface oriented, isolating low level details of the Quantum device. Following this approximation, we decide to use YANG models, as it is "de facto standard" to define open interfaces device oriented.
The QuAI API is formed by four functions with a minimal set of parameters on each call as you can see in Fig. 1. Here we present the details of the Application Interface:

- **REGISTER:** This function registers a new QKD device with a specific *deviceId,* that represents the QKD device using URI schemes inside the network. This function returns a boolean value depending on the success or failure setting up the device. If the function is successful, then an initial set of parameters needs to be sent to the device to make it work correctly.
- **SET_ATTRIBUTE:** Set a value to an individual attribute on certain QKD device. The attribute is specified as a string, that is defined in the YANG model that supports the QKD System. That approximation, makes the QuAI easy to implement on the QKD device, due to only a simple string needs to be detected on the QKD device side, and after that, the appropriate system call is executed with the associated value.
- **GET_ATTRIBUTE:** Return the current value of a specific attribute of certain QKD device. The attribute is specified again as a simple string extracted directly from the YANG model.
- **REMOVE:** This function removes a QKD device from the node.

This simple interface collects a minimal set of calls to communicate with QKD devices in a generic way. A typical approximation followed by vendors to interact with their QKD device consists on the use of specific adaptors to interact with specific devices, as you can see on Fig. 2, but this approximation makes difficult and hardly device dependent the integration of the devices into network infrastructures. Instead of that, the architecture presented on this paper is based on open interfaces, that means that if the Agent and the QKD device are model compliant, they both basically talks the same language, easing the integration of the device in the network.

Nevertheless, the extraction of parameters defined on the model to send to QuAI are not direct. In that sense, we propose the Quantum Abstraction Manager (QuAM), a module that acts as a bridge between the model and the QuAI interface. This module implements a recursive traverse of the model, collecting the attributes and their corresponding values to transform this information into QuAI calls, for example, applying the successive SET_ATTRIBUTE calls over the QKD device, making QuAM an adaptor model driven, as you can see on Fig. 3.

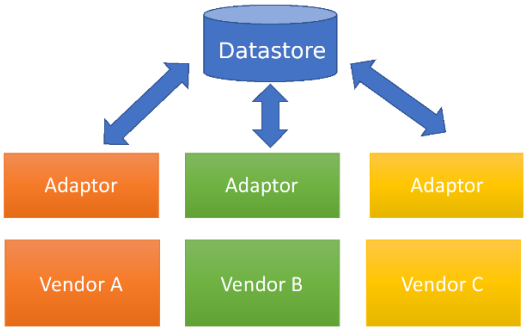

Fig. 2 Integration model based on adaptors per vendor

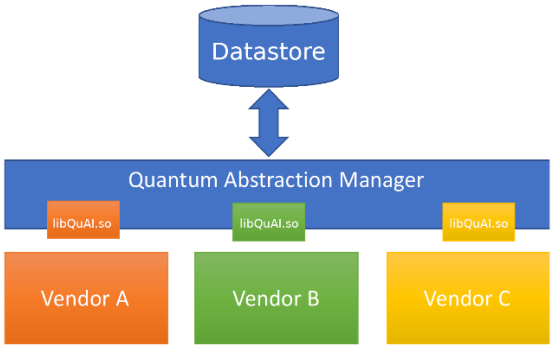

Fig. 3 Integration model based on Quantum Abstraction Interface

Note that the QuAM module is model independent and at the same time, this module could receive the data from the model from different locations, for example from a simple RPC, a database or directly from a specific location inside the infrastructure.

3. SOFTWARE ARCHITECTURE FOR QUAI INTEGRATION

Nowadays, the integration of platforms is aligned with the deployment of a compatible SDN solution. The QuAI framework for this work is deployed in a modular environment that considers the support for NETCONF/Yang models. Fig. 4 represents the software architecture for the prototype. QuAI resides in the core of the developed agent and provides the interface to enable the adaptation of the configuration to the vendor dependent commands. On the other hand, the interface to the SDN controller is a NETCONF server [4], which provides the commands to retrieve the QKD status and the configure the device. NETCONF defines three datastores: *startup*, *candidate* and *running* [4]. The *startup* configuration is the initial status for the device after its installation or reboot. The *candidate* configurations are the propose configuration from an external entity like the SDN

controller. Finally, the *running* configuration is the real configuration of the device. Every time that the SDN controller requests to modify the status of the device, a *candidate* configuration is created in the datastore. Each *startup* or *candidate* configurations are taken by the Quantum Abstraction Manager (QuAM). QuAM process translate the configuration into QuAI calls to create a *running* configuration. Therefore, any *candidate* or *startup* configuration is moved to *running* by QuAM, once the device has this configuration.

So far, all description of the agent is model agnostic. On the startup process, the agent takes as a parameter which is the information model to be used. These YANG models enable the agent to validate the configuration against the model and to facilitate the utilization of other YANG models. Similarly, the software architecture is designed to be protocol agnostic. As an example, an agent can be started with ETSI QKD 004 model [5] or ETSI QKD 015 model [6]. QuAI will be limited to the parameters in YANG model but provides a solution to operate with both interfaces. Similarly, this work considers NETCONF, but a development could be done with a gNMI or RESTCONF interface, providing a server that will create datastores that the QuAM will consume

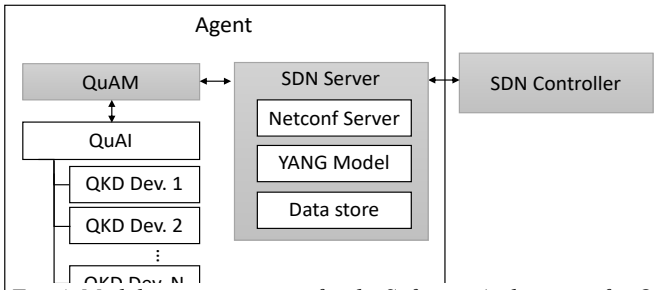

Fig. 4. Modules representation for the Software Architecture for QuAI

4. EXPERIMENTAL SETUP AND RESULTS

Our experimental setup follows a disaggregated approximation, where the Controller, the Agent and the simulated QKD Systems are separated process. The Controller acts as a client of a NETCONF server that runs inside the Agent, both of them are ETSI QKD 015 compliant and QuAM and the QuAI runs as a logic part inside the Agent.

The experiment shows how to register and configure a new QKD device. This process starts with a standard NETCONF EDIT_CONFIG RPC sent by the Controller to the Agent. The content of this RPC (Fig. 5a) is then stored in the candidate datastore of the NETCONF Server, where QuAM collects the new QKD interface configuration to adapt it into QuAI calls (Fig. 5b). The QuAI functions are implemented as raw socket communication for illustrative purposes. Each QuAI call generates a response on the QKD simulated device (Fig. 5c), if all responses are succeeded, the new configuration is stored in the running datastore and standard OK NETCONF RPC is sent to the Controller. Fig. 5d is an example of the QuAI call SET_ATTRIBUTE package travelling through the network.

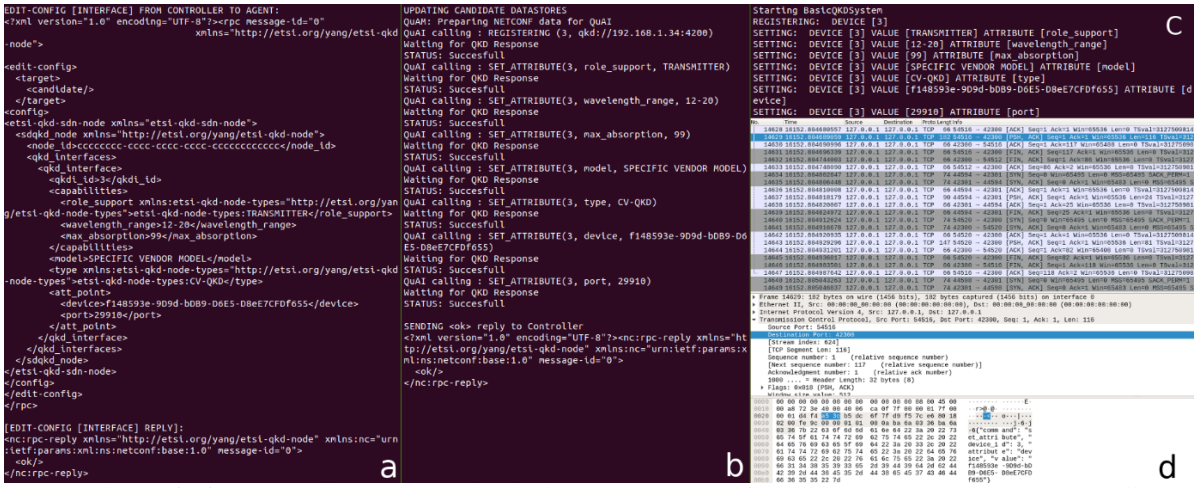

Fig. 5 a) Edit_config from controller to agent and reply. b) QuAM extraction from candidate datastores and QuAI calls. c) QKD system applying QuAI calls and answer states, d) Example of QuAI SET_ATTRIBUTE call through the network.

5. CONCLUSIONS

The main contribution of this paper is the definition of the Quantum Abstraction Interface (QuAI). QuAI is not just defined, but also integrated in an SDN agent to create a flexible procedure to integrate QKD devices in production networks. The demonstrations validate QuAI concept and its capability to configure QKD devices in a vendor agnostic fashion.

ACKNOWLEDGEMENTS

Authors would like to thank the Madrid's regional government, Comunidad Autonoma de Madrid, for the project Quantum Information Technologies Madrid, QUITEMAD+ S2013/ICE-2801, the FET Flagship on Quantum Technologies, European Union Horizon 2020 research and innovation programme under grant agreement No 820466: Continuous Variable Quantum Communications (CiViQ) and ICT grant agreement No 857156: Open European Quantum Key Distribution Testbed (OpenQKD) and the team of Transport and IP Connectivity in Telefónica Spain for their support to this activity.

REFERENCES

[1] N. Gisin, G. Ribordy, W. Tittel, H. Zbinden, "Quantum cryptography", Rev. Mod. Phys. 2002, v. 74, pp. 145-195

[2] [2] V. Martin, et al., "Quantum Key Distribution" Wiley Encyclopedia of Electrical and Electronics Engineering, Wiley pp 1-17, 2017.

[3] [3] A. Aguado, et al. "The Engineering of a SDN Quantum Key Distribution Network", IEEE Comms. Mag. July 2019

[4] [4] RFC 6241, Network Configuration Protocol (NETCONF)

[5] [5] ETSI GS QKD 004 V1.1.1 (2010-12). Group Specification. Quantum Key Distribution (QKD), Application Interface.

[6] [6] ETSI GS QKD 015 Quantum Key Distribution (QKD); Quantum Key Distribution Control Interface for Software Defined Networks (draft)