

Software-Defined Quantum Network Using a QKD-Secured SDN Controller and Encrypted Messages

R. S. Tessinari, R. I. Woodward, A. J. Shields

*Toshiba Europe Ltd, Cambridge, UK
rodrigo.tessinari@crl.toshiba.co.uk*

Abstract: We propose and implement a software-defined network architecture that integrates the QKD SDN Controller within the QKD node, enabling it to use quantum keys to secure its communication with SDN agents while optimizing QKD-keys consumption. © 2023 The Author(s)

1. Introduction

In recent years, quantum key distribution (QKD) has emerged as a robust technology for enabling quantum-safe data communications with information-theoretic security [1]. It is becoming increasingly important to consider such quantum-safe communication solutions as advances in quantum computing edge closer to the prospect of eavesdroppers having sufficient computational power to break the computational hardness assumptions of classical public key cryptography. Various demonstrations have already shown how QKD can be integrated into existing optical networks through multiplexing. However, the problem of how QKD integrates with the management layer of existing networks remains understudied, yet, it is vitally important to ensure the acceptance of QKD into practical telecom networks as well as augment quantum networks through advanced management functionality.

There is currently a trend in telecom networks towards more dynamic, reconfigurable connectivity, controlled by software, rather than relying on fixed connections. The software-defined networking (SDN) approach is an important part of this, introducing clean abstraction of the control and data planes. This brings significant benefits, including more efficient resource utilization and simplified centralized management.

While a number of works have demonstrated proof-of-concept SDN QKD networks [2–6], including in-field deployments [5, 6], the optimal architecture for leveraging and interfacing SDN functionality with off-the-shelf QKD systems remains an open question. Initial SDN QKD studies focused on bespoke implementations of interfaces, such as custom RESTful APIs or extending the OpenFlow SDN protocol [3]. Since then, various standards have been developed for QKD, including a RESTful API for key delivery (ETSI GS QKD 014 [7]) and YANG models for QKD nodes in SDN networks (ETSI GS QKD 015 [8]).

Building upon recent standards, we architect a flexible and secure SDN-enabled QKD node design which is compatible with typical QKD technology using standardized interfaces and discuss how SDN can include QKD modules, key management system and transport-layer control and monitoring. Using this design, we demonstrate

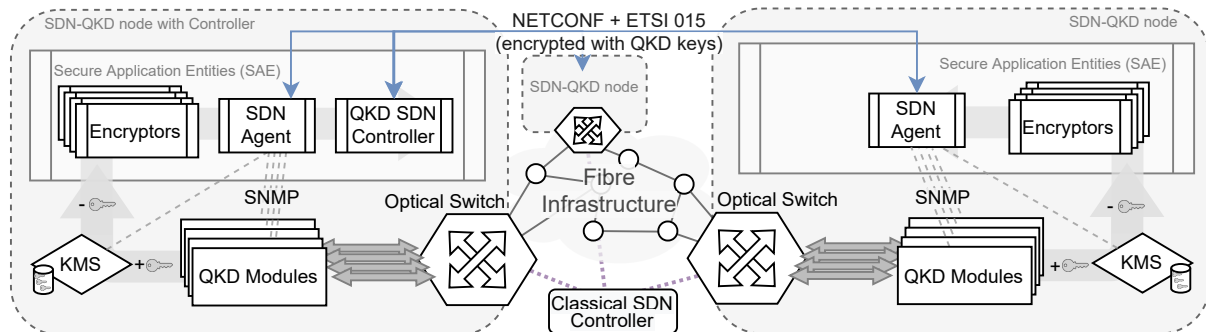


Fig. 1. QKD network architecture. All nodes can host multiple QKD devices and generate and serve keys to applications (e.g., encryptors). Additionally, all SDN-enabled nodes host SDN agents and potentially QKD SDN controllers (left). A classical SDN Controller establishes and manages the optical paths between the various nodes.

a 4-node SDN quantum network including multiple QKD links and reconfigurable optical switching, enabling automated failover protection and a scalable, dynamic design for future quantum-secured telecom networks.

2. Architecture Overview

Although many published works point out that QKD keys may (or should) be used to secure the communication between SDN controllers and SDN agents, how to do it remains an open question. Our solution is two-fold: first, we describe how to implement the QKD SDN controllers and agents within the QKD node, and then we describe how to optimize QKD keys utilization to avoid key material starvation.

Fig. 1 shows a many-nodes QKD network, highlighting two types of QKD nodes. On the left is a QKD node with a controller, and on the right is a node without it. The main idea is to implement both the controllers and agents as Secure Application Entities (SAE). According to ETSI GS QKD 014 [7], an SAE is an entity that requests one or more keys from a node's Key Management Entity (i.e., Key Management System - KMS, in our architecture). In practice, any security rules and procedures enforced to encryptors must also be followed by the controllers and agents. For instance, all secure applications are given X.509 Certificates to make themselves identifiable by the KMS. All communication between controllers and agents and the node's KMS is made over HTTP/2.0 using a TLS 1.2/1.3 channel.

With access to keys, those keys can be used to encrypt the controller-related messages exchanged between agents and controllers. We suggest that this communication be performed using either NETCONF or RESTCONF protocols, relying on standards for increased compatibility. This work uses a combination of NETCONF and ETSI GS QKD 015 [8] YANG models. By default, NETCONF messages are encoded using XML. Due to the notorious verbosity of XML, if all the contents of the messages are encrypted, precious key material may be unnecessarily wasted. With this in mind, we designed our agents and controllers to encrypt specific fields of the XML data instead of the whole message. There are different encryption levels, and users can choose between them. As cipher options, it is possible to use one or both One-Time-Pad (OTP) and Advanced Encryption Standard (AES). Fig. 2 shows examples of how the partial encryption schemes work on a short XML message sent during a NETCONF operation to delete a QKD link. For a UTF-8 encoded string with 349 characters (i.e., one Byte per character), it would be necessary to use 3048 bits of key material to one-time-pad the whole message and then apply AES-256, Fig. 2a). Fig. 2b) shows the partially encrypted message when encrypting only the data, disregarding the XML tags. Finally, c) shows an intermediary approach in which the tags (and data) qkdn_id and qkd_links are encrypted.

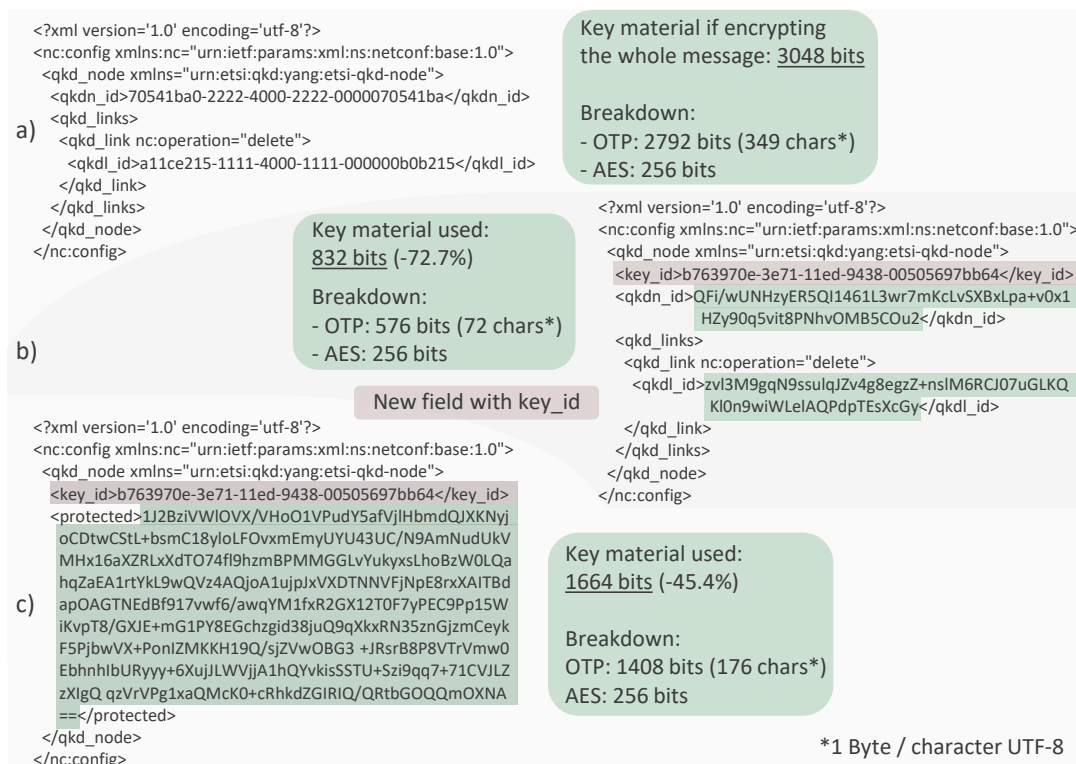


Fig. 2. a) XML containing a delete QKD link operation sent from the controller to a node via NETCONF. b) Same XML encrypting only the data inside the tags. c) Encryption of tags and data.

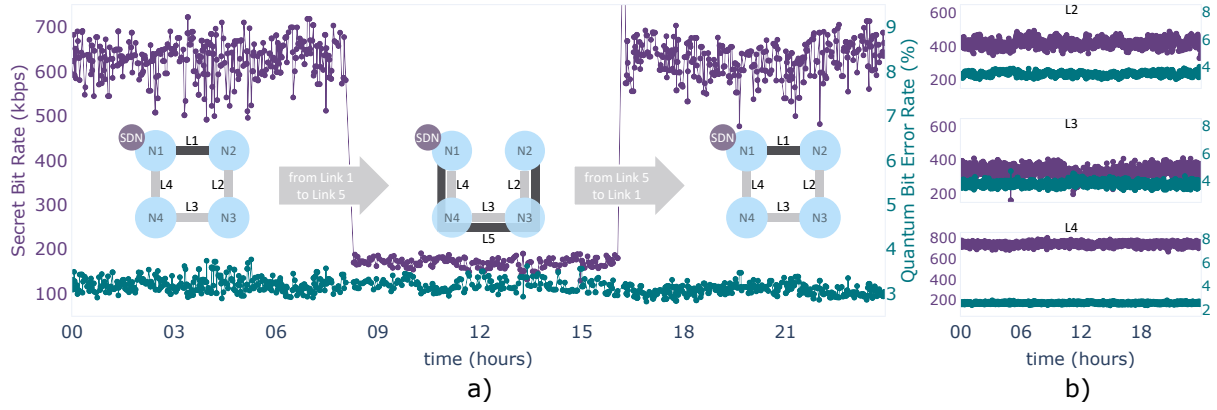


Fig. 3. Monitoring data acquired by the SDN Controller during 24 hours of operation. The secret bit rate (SBR) and quantum bit error rate (QBER) showed in purple and teal colors, respectively. a) shows the two switching operations performed back and forth from links L1 and L5. Due to L5's higher losses, a reduction in SBR is perceived. b) monitored data of links L2, L3 and L4.

This approach to partially encrypt the contents of the XML messages rewards massive savings in key material usage, which can be critical in extreme long-distance links with low key rate generation.

3. Testbed Description and Results

To demonstrate the feasibility of our architecture, we deployed a 4-node SDN quantum network with multiple QKD links. Each node includes a controllable optical switch, a pair of QKD devices, and multiple servers for the QKD software, Key Management System, and SDN agents. The SDN Controller resides in node N1. The topology is shown as an inset in Fig. 3. When the network was operational, we tested automatic recovery from a link outage, which, for example, made link L1 unavailable. In this case, our SDN controller switched the existing link to a new set of fibers (link L5) to enable continuous QKD key generation despite the link L1 outage. Fig. 3 shows 24 hours of operational data, including the eight hours while L1 was unavailable. During the first eight hours of operation, the four pairs of QKD devices ran over links L1 to L4, with losses of ≈ 4.5 dB each. Then, the classical SDN controller issued a command to switch the connection between N1 and N2 from link L1 to a lossier link L5 (≈ 9.5 dB), resulting in an expected decrease in key material generation rate. After another eight hours, the network is reverted to the initial configuration. These results are shown in Fig. 3a), whereas b) shows monitored QBER and SBR of links L2, L3, and L4.

4. Conclusion

We architect a flexible and secure SDN-enabled QKD node design compatible with typical QKD technology using standardized interfaces and discuss how SDN can include QKD modules, key management system, and transport-layer control and monitoring. We also propose partial data encryption and potential QKD key material savings. Finally, we demonstrate a 4-node SDN quantum network using this design, including multiple QKD links and optical switches. Further analysis of the network monitoring clearly shows the SDN controller acting upon the QKD network, paving the way to reconfigurable and scalable, quantum-safe networks.

References

1. M. Mehic, et al. Quantum Key Distribution: A Networking Perspective. *ACM Comput. Surv.*, 53(5):1–41, 2020.
2. A. Aguado, et al. Hybrid Conventional and Quantum Security for Software Defined and Virtualized Networks. *J. Opt. Commun. Netw.*, 9(10):819–825, 2017.
3. A. Aguado, et al. Virtual Network Function Deployment and Service Automation to Provide End-to-End Quantum Encryption. *J. Opt. Commun. Netw.*, 10(4):421, 2018.
4. E. Hugues-Salas, et al. Monitoring and Physical-Layer Attack Mitigation in SDN-Controlled Quantum Key Distribution Networks. *J. Opt. Commun. Netw.*, 11(2):A209–A218, 2019.
5. D. R. Lopez, et al. Demonstration of Software Defined Network Services Utilizing Quantum Key Distribution Fully Integrated with Standard Telecommunication Network. *Quantum Reports*, 2(3):453–458, 2020.
6. O. Alia, et al. Dynamic DV-QKD Networking in Trusted-Node-Free Software-Defined Optical Networks. *J. Lightwave Technol.*, 40(17):5816–5824, 2022.
7. ETSI GS QKD 014-Quantum Key Distribution; Protocol and Data Format of REST-based Key Delivery API, 2019.
8. ETSI GS QKD 015-Quantum Key Distribution; Control Interface for Software Defined Networks, 2022.