Simulations of Denial of Service Attacks in Quantum Key Distribution Networks

Emir Dervisevic[†], Filip Lauterbach[*], Patrik Burdiak[*], Jan Rozhon[*], Martina Slívová[*], Matej Plakalovic[†], Mirza Hamza[†], Peppino Fazio[‡*], Miroslav Voznak[*], Miralem Mehic[†]

[†] Department of Telecommunications, Faculty of Electrical Engineering, University of Sarajevo, Zmaja od Bosne bb, 71000, Sarajevo, Bosnia and Herzegovina
[‡] DSMN, Ca' Foscari University of Venice, Via Torino 155, 30172 Mestre (VE), Italy
[*] Faculty of Electrical Engineering and Computer Science, VSB – Technical University of Ostrava, 17. listopadu 2172/15, 708 33 Ostrava, Czechia

***Abstract*—A QKD network can be considered an add-on technology to a standard communication network that provides IT-secure cryptographic keys as a service. As a result, security challenges resulting in the suspension of functional work must be addressed. This study analyzes a Denial of Service (DoS) attack on the Key Management System (KMS), one of the critical components of the QKD network in charge of key management and key provisioning to authorized consumers. Through simulation methods performed in the QKDNetSim, we show that legitimate customers experience significantly worse service during an excessive DoS attack on KMS.**

***Index Terms*—Quantum Key Distribution Networks, Quality of Service, Simulations, Networking**

I. Introduction

A Quantum Key Distribution (QKD) network is an add-on technology to a standard communication network that provides Information-Theoretically Secure (ITS) cryptographic keys (donated simply as keys in the rest of the text) as a service. Its sole purpose is to generate, manage, and distribute keys between remote locations in an ITS manner, as well as to provide keys as a service to end consumers (e.g., IPsec encryptors) [1].

The QKD network differs from traditional telecommunication networks not only in purpose, but also in the manner in which connections are established. The adjacent nodes of the QKD network, i.e., QKD nodes, are connected through logical QKD links, consisting of two channels: a quantum channel for transmitting cryptographic values encoded in specific photon characteristics and a public channel for verifying and processing the exchanged data. A quantum channel creates a physical path for photons to travel directly and unobstructedly between two adjacent nodes. A public channel, on the other hand, can be implemented as any ordinary connection, with an arbitrary number of intermediary devices [2].

The logical QKD link allows adjacent QKD nodes to run a QKD process, i.e., QKD protocol, that enables symmetric key growth in an IT-secure manner [3]. Given that current generation quantum systems produce keys at rates of up to several hundred kbps [4], this is usually insufficient to support the protection of high-demand communication flows, emphasizing the necessity for effective key management. As a result, both endpoints of the corresponding QKD link implement limited-capacity key buffers (storages) that are gradually filled with the produced key material. The term "key material" refers to the symmetric cryptographic keys created during the QKD process. Stored key material is then utilized for cryptographic operations over user traffic [5], or for a key relay procedure that allows arbitrary QKD nodes to share keys across the QKD network in a hop-by-hop manner.

Key management is the most significant challenge in effectively and securely running a QKD network. Key management tasks include storing keys, relaying keys across the QKD network, and supplying keys on-demand to a broad range of applications with varying Quality of Service (QoS) requirements. These tasks are entrusted to a Key Management System (KMS) - an essential component housed within each QKD node [6]. Since KMS is the first point of contact for communication with user applications, it is reasonable to investigate its security capabilities against threats that could cause network failure, i.e., unavailability of QKD network services. This is a significant threat because QKD network services are most likely to be used to protect very sensitive traffic flows in critical infrastructures where key delivery interruption is unacceptable.

This paper analyzes Denial of Service (DoS) attacks against KMS entities. We use Quantum Key Distribution Network Simulation (QKDNetSim) module developed in network simulator 3 (ns-3) to simulate DoS attacks against KMSs and collect measurement results. We discuss how these attacks affect the level of service provided to legitimate consumers, i.e., user applications. [7], [8] This research raises awareness of security threats in QKD networks that are considered in a classical domain rather than those strictly related to the QKD process itself.

This paper is organized as follows: Section II describes the structure of the QKD nodes that comprise the QKD network. Section III describes the most common interface to receive services from the QKD network. Section IV present the idea behind DoS attacks against KMS entities, while section V

describes the simulation experiment setup. The discussion of obtained results is given in section VI while section VII concludes this study and outlines the future work.

II. QKD Architecture

Although there are several different approaches for defining the structure of the QKD node and, thus, functional requirements for the KMS entity, it is important to highlight the approach being developed within the EU H2020 OPENQKD[1] project [9].

Consider a Secure Application Entity (SAE) that wants to establish secure communication with a remote SAE, such as an IPsec VPN tunnel, as shown in Fig. 1. The data transmitted via tunnel is to be protected using symmetric cryptographic algorithms that use QKD-derived keys provided by the QKD network. As a result, before establishing the VPN tunnel, SAEs must obtain previously established symmetric keys from their respective KMS entities [5]. The KMS is the initial point of contact for processing requests from key-seeking applications. It is linked to a QKD control entity, which operates and monitors the QKD devices within the same QKD node by performing routine tasks such as power on/off, reboot, restart, QBER/temperature monitoring, and so on. The KMS is aware of the status of QKD systems that establish QKD links with nearby nodes and can communicate with other KMS entities to share management information.

Furthermore, the node's KMS communicates with a specific routing entity, which attempts to compute the optimal route for key relay based on information gathered about QKD link statuses and application requirements. It is critical to emphasize that a routing/forwarding entity should be physically placed separately from the KMS entity, as this would give a great degree of flexibility. Thus, routing decisions can also be made by external entities such as Software-Defined Network (SDN) controllers. The separation of the routing component enables the network administrator to change or upgrade the routing logic as needed without interfering with communications between other entities in the network node.

III. QKD End-User Standards

Several available standards discuss QKD integration with IP networks. The ETSI GS QKD 014 standard, which describes an Application Programming Interface (API) between end-user applications and the QKD network's KMS entities, is highlighted in this paper [10]. It is based on the resource access flow approach using REpresentational State Transfer (REST) key acquisition communication. REST is an architectural style for distributed organizations that relies on six guiding principles: client-server paradigm, stateless and cacheable communication, uniform interface, layered system, and option code on demand[2].

The use of the REST approach in the context of the ETSI 014 standard means that there are no dedicated reservations of key resources. QoS service is not guaranteed, and the responsibility of providing QoS communication is transferred from KMS entities to SAE applications. Since the communication is performed on the stateless REST principle, there is no session establishment function, session identifiers (KSID), or session closing functions as in ETSI GS QKD 004 standard, which is based on the reservation of key resources to deliver satisfactory QoS experience. ETSI 014 defines three API functions:

- GET_STATUS: probes the status of the available key material shared with the desired SAE destination. The function returns information about the assigned KMS entity for the destination application, the key size that can be delivered to the SAE (bits), the number of keys stored in the key buffers, and the maximum number of keys that can be delivered in one response.
- GET_KEY: obtains keys to reach the destination SAE application. The input values conveyed by this message are the number and size of keys requested, additional SAE IDs which can be used to specify two or more slave SAEs sharing the same key, and additional extension parameters. The response is the key container JSON structure, including the key identifier(s) (ID) [3], base64-encoded key value(s), and additional key_extension values.
- GET_KEY_WITH_KEY_IDS: obtains keys from KMS for the slave SAE. The input value conveyed by this message is the key identifier(s) ID, while the output should be the same as from the GET_KEY message.

As shown in Figure 2, Alice's SAE application will first request information about the status and possibility of establishing a connection to the destination SAE application. After obtaining a response, the application will decide whether and when to request key(s) from the KMS by sending a GET_KEY message. It is up to the application to consider the obtained status information and calculate the number of keys it will ask for. The maximum number of keys that can be requested from KMS per single query is provided in response to the GET_STATUS message. However, asking for larger quantities of keys per single query can reduce further queries and reduce jitter. The application should store the obtained keys in the local buffer and use them as needed without waiting for subsequent queries and answers from the KMS, leading to delays. However, the request to deliver many keys in a single query can lead to increased delay. Due to the enormous size of the content, many IP packets might be fragmented along the path to the destination.

IV. Attack Scenario

Consider a situation in which an attacker physically comes into possession of user terminal equipment with which he can communicate freely with the KMS. Certificates on the stolen user equipment are valid, the equipment is within a secure perimeter of communication, and there are no physical barriers that could prevent communication with KMS.

[1] www.openqkd.eu

[2] More details about REST can be found at www.restfulapi.net

[3] Note that the keys are organized into blocks, and each block of the cryptographic key is identified using a unique ID. The ID values must be identical on both sides of the QKD link to be managed.

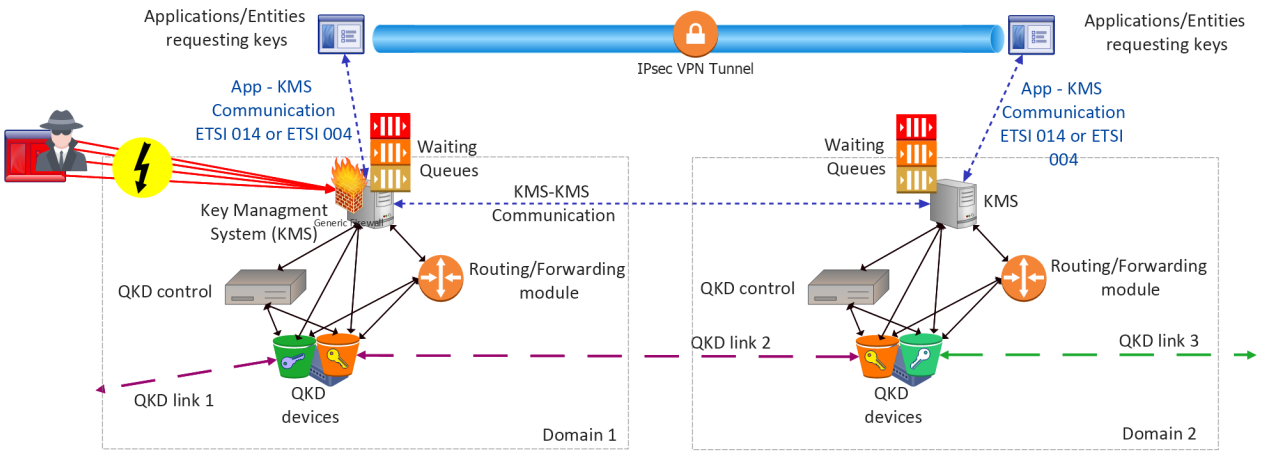

Fig. 1. Schematic illustration of attack on KMS node

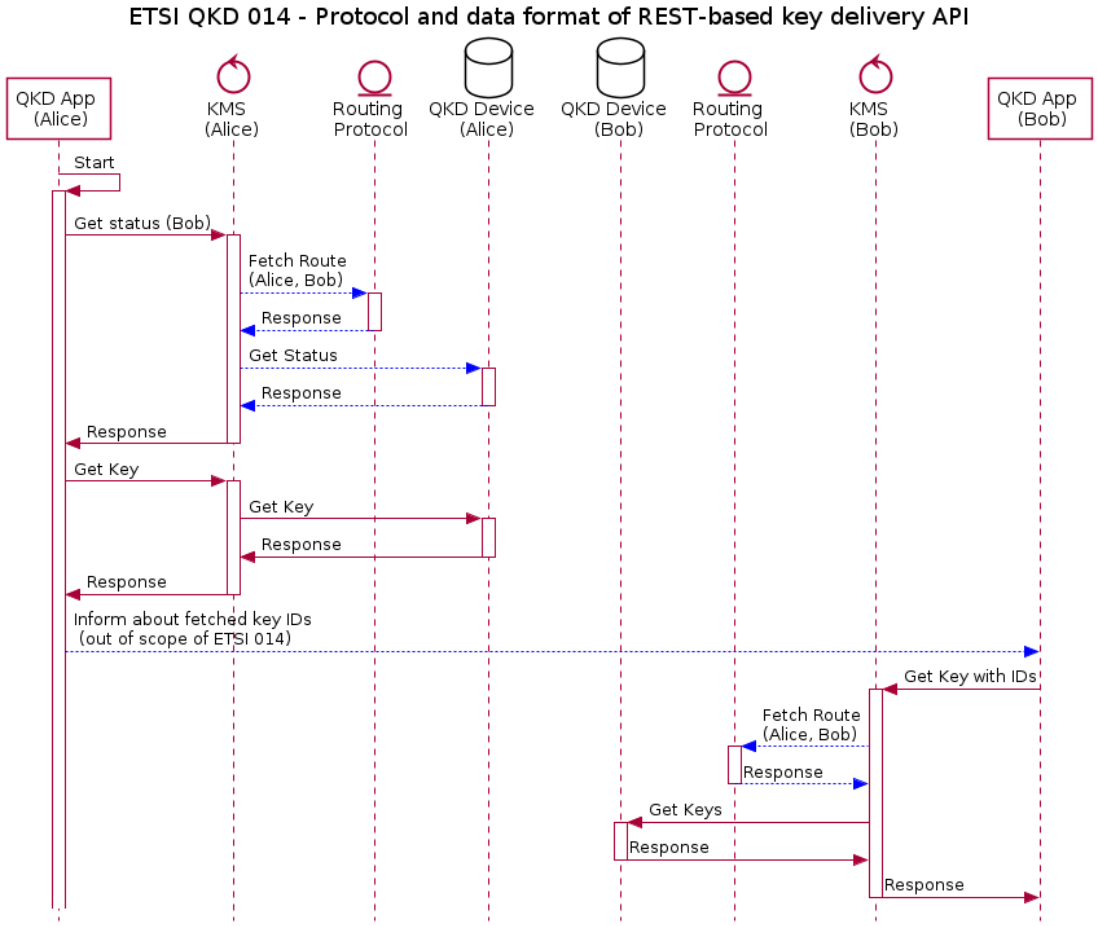

Fig. 2. Sequence diagram of ETSI 014 – Protocol and data format of REST-based key delivery API. The blue dotted lines are beyond the scope of the ETSI 014 standard [10].

An attacker may choose to intentionally obtain a large amount of keys to consume all key resources and potentially influence routing mechanisms to redirect traffic to those links or devices that are under his control [11], [12]. In this context, an attacker will generate a large number of GET_KEY queries to obtain as many keys as possible in a short period of time. Because the primary resource managed in the QKD network is the cryptographic key, KMSs must agree on the number of keys that can be served between domains using dedicated Service Level Agreements (SLAs). Additionally, since the processing of each of the SAE requests on the KMS increases the computational load, it is necessary to define the traffic capacity that can be served from other domains. This is also a weak point that an attacker can count on. Generating a large number of queries can affect the amount of traffic exchanged between KMSs, ultimately leading to non-serving legitimate users.

Similarly, the attacker can degrade the QKD network's level of service simply by overwhelming KMS with requests with an unknown destination SAE. In this case, the attacker does not use QKD network's most scarce resource - key material, but instead forces KMS to perform unnecessary processing of requests, potentially degrading the level of service experienced by other SAEs. This scenario is analyzed in this study, and is explained in more details in Section V.

V. Simulation Setup

The simulations were carried out using QKDNetSim module [7], [8], developed in ns-3 network simulator, which is the first computing simulation environment for QKD networks that focuses on the public channel performance and networking in the classical domain with a maximum simplification of QKD process.

The experiment, illustrated in Figure 1, considers a pair of end-user applications that use QKD-derived keys in a One-Time Pad (OTP) data encryption process to achieve the most advanced secure communication in terms of data confidentiality. End-user applications communicate with their respective KMS entities using ETSI GS QKD 014 interface. The intensity of their data communication is variable, i.e. it ranges from 30kbps, 50kbps, 100kbps to 150kbps. Also, the generated packets' size varies and takes values from a set of 100, 300, 500, 800, and 1100 bytes.

Because the emphasis is on the end-user experience of accessing keys, the underlying QKD network has a simple topology consisting of point-to-point links between two QKD nodes. End-user applications are contained within the secure perimeter of their respective QKD nodes, which include, among other modules, KMS entities.

Since the KMS cannot process a large number of requests simultaneously, they are stored in a queue where they can be sorted by priority. For our needs, a Waiting Fair Queueing (WFQ) mechanism with a maximum capacity of 100 packages has been implemented. As a result, all the requests are stored in a queue and wait for their turn to be processed. The processing of the requests has a deterministic time-cost. Also, the link capacity between the two KMSs is set to a peak intensity of 10 Mbps. Although this value is certainly higher in practice, it is sufficient to simulate and analyze security challenges.

To simulate a DoS attack against the KMS entity, the malicious applications are placed within the secure perimeter of one QKD node. Malicious applications are simplified end-user applications that do not perform any processing of received responses. Their goal is to send invalid requests to KMS at regular intervals. To model the intensity of the DoS attack, the experiment is carried out with a varying number of malicious applications. Also, the parameter p is defined, which determines the time interval between sending consecutive GET_KEY queries to the KMS with randomly defined values. The parameter p takes values from the set 0.05, 0.1, 0.3 and 0.5 seconds. A total of 240 simulations with different seed values of the random number generator were performed.

VI. Simulation Results

Figure 3 shows the total number of requests sent to the KMS depending on the number of malicious applications and the intensity of their attacks. It can be noticed that the simulated load growth on the KMS is linear and that the essential parameter represents the intensity of the overall attack.

Figure 4 shows the effects of malicious applications on the number of encrypted packets exchanged between legitimate SAE applications. As the number of malicious attackers and the intensity of attacks increase, increasing congestion on the KMS results in fewer acquired keys necessary to encrypt data traffic.

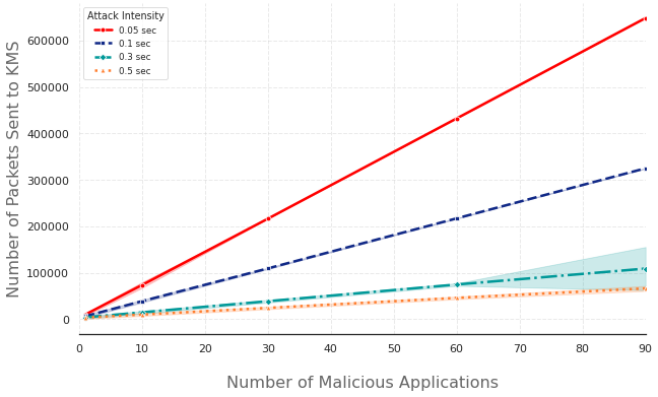

Fig. 3. Number of malicious packets sent to KMS vs. number of malicious applications and their attack intensity

Figure 5 shows the effects of malicious applications on the consumption of cryptographic QKD keys. Namely, it is possible to notice a significant consumption of cryptographic keys by legitimate SAE applications without an attack. But by increasing the load on the KMS, all requests from malicious applications are stored in the same queue, which leads to squeezing or delays in processing requests arising from legitimate SAE applications.

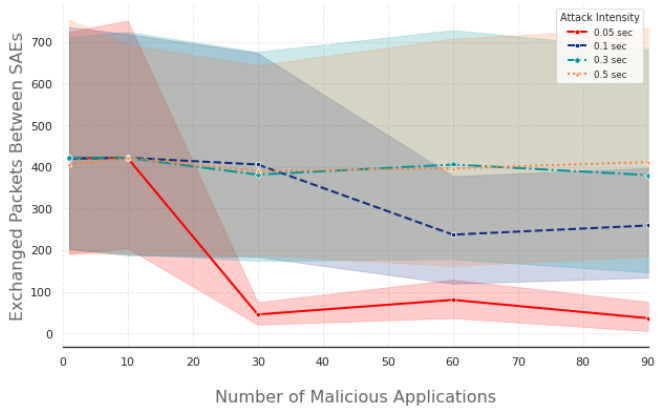

Fig. 4. The influence of malicious applications and their attack intensity on the number of exchanged encrypted packets between legitimate user's application.

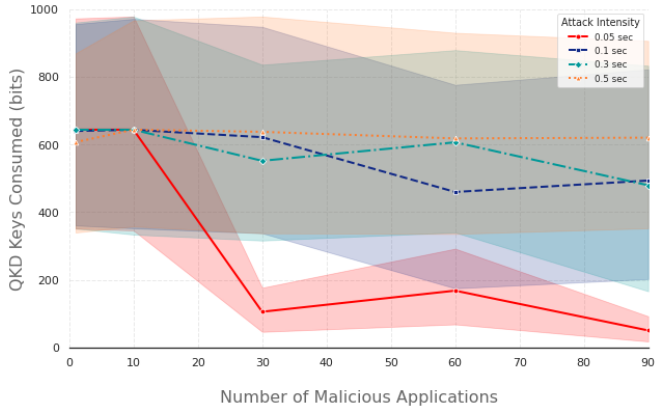

Fig. 5. The influence of malicious applications and their attack intensity on the average QKD key consumption (bits) by the legitimate user's application.

It can be observed that legitimate users receive significantly less service at increased attack intensities. It can also be observed that the variability of the results (shaded areas) is substantially less with increased attack intensity. Increasing the number of malicious users is not a dominant factor in the performance of KMS as long as they generate requests with lower intensity.

VII. Conclusion

This paper presents the impact of malicious applications on the functionality of KMS network entities. With the tendency to integrate QKD technology in everyday telecommunication networks, it is necessary to pay attention to security challenges that can lead to the suspension of functional work. In addition to focusing on the number of cryptographic keys available to SAE applications, KMS must also pay attention to the number of requests generated by SAE applications. If an attacker comes into possession of such terminal devices, he can generate attacks to hinder the functional operation of network entities. As a result, in collaboration with network controllers and managers, KMS entities must be capable of detecting abnormal behavior of registered applications and temporarily suspending their access to QKD network services in such instances. KMS may implement internal security mechanisms or rely on adapting well-known authentication, authorization, and accounting (AAA) solutions from conventional IP networks.

The main contribution of this paper is the analysis of DoS and DDoS attacks on KMS network entities in QKD networks.

Acknowledgment

The research leading to the published results was supported by the NATO SPS G5894 project "Quantum Cybersecurity in 5G Networks (QUANTUM5)", partly by the H2020 project OPENQKD under grant agreement No. 857156 and SGS reg. no. SP2022/5 conducted at VSB -Technical University of Ostrava, Czech Republic. This work was also supported by the Ministry of Science, Higher Education and Youth of Canton Sarajevo, Bosnia and Herzegovina under Grant No. 27-02-11-41251-13/21.

References

[1] M. Dianati and R. Alléaume, "Architecture of the secoqc quantum key distribution network," in *2007 First International Conference on Quantum, Nano, and Micro Technologies (ICQNM'07)*. IEEE, 2007, pp. 13–13.

[2] M. Mehic, O. Maurhart, S. Rass, D. Komosny, F. Rezac, and M. Voznak, "Analysis of the Public Channel of Quantum Key Distribution Link," *IEEE Journal of Quantum Electronics*, vol. 53, no. 5, pp. 1–8, oct 2017.

[3] G. Brassard, L. Salvail, S. Louis, L. Salvail, and S. Louis, "Secret-key reconciliation by public discussion," *Advances in Cryptology - EUROCRYPT93*, vol. 765, pp. 410–423, 1994.

[4] M. Mehic, M. Niemiec, S. Rass, J. Ma, M. Peev, A. Aguado, V. Martin, S. Schauer, A. Poppe, C. Pacher, and M. Voznak, "Quantum Key Distribution," *ACM Computing Surveys*, vol. 53, no. 5, pp. 1–41, oct 2020.

[5] E. Dervisevic and M. Mehic, "Overview of Quantum key distribution technique within IPsec architecture," in *Proceedings of the 18th International Conference on Information Systems for Crisis Response and Management ISCRAM 2021*. ACM, 2021, pp. 1–10.

[6] M. Mehic, S. Rass, P. Fazio, and M. Voznak, *Quantum Key Distribution Networks: A Quality of Service Perspective*. Springer International Publishing, 2022, in Press.

[7] M. Mehic, O. Maurhart, S. Rass, M. Voznak, M. Miralem, M. Oliver, R. Stefan, and Miroslav Voznak, "Implementation of Quantum Key Distribution Network Simulation Module in the Network Simulator NS-3," *Quantum Information Processing*, vol. 16, no. 10, p. 253, oct 2017.

[8] F. Kutschera, E. Dervisevic, L. Behan, D. López, M. Mehic, M. Voznak, H. Hübel, A. Pastor, and L. Cepeda, "Data acquisition and simulation tools for virtual qkd testbed access–examples from the openqd project," in *2021 European Conference on Optical Communication (ECOC)*. IEEE, 2021, pp. 1–4.

[9] V. Martin, J. P. Brito, C. Escribano, M. Menchetti, C. White, A. Lord, F. Wissel, M. Gunkel, P. Gavignet, N. Genay, O. Le Moult, C. Abellán, A. Manzalini, A. Pastor-Perales, V. López, and D. López, "Quantum technologies in the telecommunications industry," *EPJ Quantum Technology*, vol. 8, no. 1, p. 19, dec 2021.

[10] ETSI ISG QKD, "ETSI ISG QKD 014 - Quantum Key Distribution (QKD); Protocol and data format of REST-based key delivery API," vol. 1, pp. 1–22, 2019.

[11] S. Rass and K. Sandra, "Indirect Eavesdropping in Quantum Networks," in *The Fifth International Conference on Quantum, Nano and Micro Technologies, ICQNM*, 2011, pp. 83–88.

[12] S. Rass and S. König, "Turning Quantum Cryptography against itself: How to avoid indirect eavesdropping in quantum networks by passive and active adversaries," *International Journal On Advances in Systems and Measurements*, vol. 5, no. 1 2, pp. 22–33, 2012.