Virtual Quantum Key Distribution Network Ecosystem: The National Czech QKD Network

Miralem Mehic, Emir Dervisevic, Peppino Fazio, and Miroslav Voznak

Abstract

Network emulators are essential in testing network systems, applications, and protocols. Emulators bridge the gap between simulation setups that lack realism in results and real-world trials that are accurate but often expensive, non-reproducible, and uncontrollable. This paper describes the simulations and emulations of the national Czech QKD network. Using emulation techniques, a unique ecosystem is formed that includes the processes of generating, processing, storing, and consuming cryptographic keys. The presented tool will undoubtedly spur future development, understanding, and teaching, and it is critical for testing novel applications and protocols applied to QKD networks.

Introduction

With rapid breakthroughs in quantum computing, network security specialists have never been under greater pressure to deliver innovative solutions to the impending security concerns of the post-quantum age. Large-scale quantum computing will significantly impact many areas of traditional information technology. The aim is to identify quantum-safe alternatives that will meet these challenges.

One of those solutions is Quantum Key Distribution (QKD), which secures the distribution of symmetric cryptographic keys between distant users by applying quantum physics principles [1]. However, the longevity of any new solution is determined by its ability to fit into existing telecommunications infrastructure. Integrating QKD into next-generation networks, such as 5G or 6G, is critical for building coherent, secure, and rapid telecommunications services. More recently, more comprehensive integration initiatives of QKD technology have been noticeable. The European Quantum Communication Infrastructure (EuroQCI) initiative stands out here, as it provides quantum communication infrastructure spanning the whole of the EU, including its overseas territories. The European Commission is collaborating with all 27 EU Member States and the European Space Agency (ESA) to design, develop, and deploy the EuroQCI, which will consist of a terrestrial segment based on fibre communications networks linking strategic sites at national and cross-border levels, as well as a space segment based on satellites. The EuroQCI will protect sensitive data and key infrastructures by incorporating quantum-based technologies into current communication networks, adding a layer of security based on quantum physics. However, with such initiatives comes the question of proper integration architectures and the feasibility study of deploying expensive and sensitive QKD equipment.

To perform preparatory actions for equipment integration, engineers can rely on simulation/emulation tools to test compatibility and calculate expected performance. These tools provide direct answers to questions about proper network configuration and organization, such as (not limited to): What performance of the QKD system can be expected if the existing optical multi-hop network is used? How many VPN tunnels can be established, and what are the executable key rates? How much network performance can be increased if an additional QKD link is installed between certain nodes?

While there is an assortment of simulation tools dealing with upcoming quantum technologies, none address the networking elements of QKD networks [2], [3], [4], [5]. They enable simulation of quantum processors, communication on conventional equipment, and the creation of quantum network software applications. Additionally, those simulation platforms do not implement a full TCP/IP stack or full key-management functionality. They are unsuitable for addressing practical issues connected with QKD networks that extend beyond the quantum/optical connectivity and encompass sophisticated features of key and network management [6], [7].

This paper presents an extended version of the QKDNetSim simulation module [6] and its emulated capabilities [8]. QKDNetSim is based on the

Miralem Mehic (corresponding author) is with the Department of Telecommunications, Faculty of Electrical Engineering, University of Sarajevo, 71000 Sarajevo, Bosnia and Herzegovina, also with the Department of Telecommunications, VSB-Technical University of Ostrava, 70800 Ostrava, Czechia, and also with the Marine Research Institute, Klaipeda University, 92295 Klaipėda, Lithuania; Emir Dervisevic is with the Department of Telecommunications, Faculty of Electrical Engineering, University of Sarajevo, 71000 Sarajevo, Bosnia and Herzegovina; Peppino Fazio is with the Department of Molecular Sciences and Nanosystems, Ca' Foscari University of Venice, 30172 Mestre, Italy, and also with the Department of Telecommunications, VSB-Technical University of Ostrava, 70800 Ostrava, Czechia; Miroslav Voznak is with the Department of Telecommunications, VSB-Technical University of Ostrava, 70800 Ostrava, Czechia, and also with the Marine Research Institute, Klaipeda University, 92295 Klaipėda, Lithuania.

Digital Object Identifier: 10.1109/MNET.2025.3540705
Date of Current Version: 13 May 2025
Date of Publication: 10 February 2025

Integrating QKD into next-generation networks, such as 5G or 6G, is critical for building coherent, secure, and rapid telecommunications services.

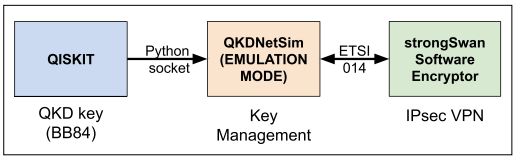

FIGURE 1. A layered virtual QKD network ecosystem. Qiskit generates raw keys that are pushed to QKDNetSim for key management. StrongSwan software encryptors establish a VPN IPsec tunnel using keys fetched from QKDNetSim.

generic trusted-relay paradigm, well-established QKD network components (such as QKD network components, key management system, and cryptographic applications), and interfaces such as ETSI QKD 014 [9] and ETSI QKD 004 [10]. The tool is critical for evaluating novel network management approaches applied to large-scale QKD networks [11]. With its emulation plugins, QKDNetSim is extended into a platform that simulates the entire QKD network but also provides additional interfaces to hardware/software QKD devices that generate keys as well as encryptors that consume those keys (Figure 1).

This paper is organized as follows: the section "Fundamentals of QKD" briefly introduces fundamentals of QKD networks. The section "QKD Network Simulation Module (QKDNetSim)" briefly describes the most important component of the extended model of QKDNetSim. The simulation and emulation experimental setup are given in the section "QKDNetSim Experiment Setup." In the section "Results" obtained results are presented and discussed. The section "Conclusion" concludes the study.

Fundamentals of QKD

Unlike existing cryptographic solutions that rely on the computational difficulties of mathematical issues, Quantum Key Distribution (QKD) employs quantum physics principles to securely generate information-symmetric (ITS) cryptographic keys between users at the QKD link's edges. QKD necessitates a logical QKD link consisting of a direct optical connection, i.e., a quantum channel and an authorized public channel. The quantum channel transmits a random sequence of bits encoded in photons (particles of light), benefiting from some of their properties to act as information carriers. This transmission is distinctive because it cannot be successfully read or duplicated, and any eavesdropping attempt can be detected [7]. It allows two distant users to establish correlated, secret data validated for accuracy and privacy using standard techniques that need a trustworthy, authenticated public channel.

Because of the nature of quantum communication, QKD is a point-to-point technique with a restricted range. A network of QKD nodes and links has been realized to grow the system to a multi-user, distance-independent secret key agreement. In recent years, several successful network demonstrations[1] have demonstrated substantial progress in the development of quantum technology [1], [12].

The production of keys by current generation quantum systems may reach several hundred kbps over mid-range optical links [1]. However, this is usually insufficient to support high-demand communication flows, underscoring the significance of effective key management. Because of this, the associated QKD link's two ends feature limited-capacity key buffers, or storages, which are then employed for cryptographic operations over user data traffic after being gradually loaded with the produced keys (commonly referred to as "key material").

It becomes apparent that all intermediary nodes must be engaged to establish keys between remote nodes that are not directly linked. They work hop-by-hop to create the key relay path, a network of QKD connections. Known as a trusted-repeater QKD network, this is the most common architecture in practical applications. One can observe the similarities of the QKD network with mobile ad-hoc networks (MANETs) as highlighted in [7].

QKD Network Simulation Module (QKDNetSim)

A rising number of simulators for testing quantum algorithms are being developed in response to the growing interest in quantum technology. To conduct experiments and validate their results, researchers can build complex network topologies with a high degree of control within the simulation environment. In addition, the simulator's sophisticated parameter adjustments provide a more accurate and practical viewpoint on the issue while saving money compared to an actual testbed.

A simulation module called QKDNetSim was created to assess various methods for implementing QKD technology. It is produced in Network Simulator 3 (NS-3) as the best fit to satisfy the following criteria. The requirements include accuracy of results, extensibility, usability, and availability. A considerably simplified perspective of the QKD network and its resources was provided by the initial QKDNetSim architecture. Our enhanced QKDNetSim module is centered around key management approaches (when and how to utilize the key, which API to use to obtain the key, and other related topics) [6]. The latest stable version of QKDNetSim is 2.1.6, compatible with the NS-3 simulator v3.41[2].

Key management is the biggest obstacle to efficiently and safely operating a QKD network. Collecting keys from QKD modules—that is, QKD systems from various manufacturers that could utilize multiple QKD technologies—storing keys, transmitting keys across the QKD network, and providing keys to cryptographic applications on demand are among the key management activities. The way that KMS maintains and provisioned keys is crucial since the supply of QKD-derived key material is limited. QKDNetSim implements a dedicated KMS as an application. The most critical key management processes may be facilitated by supporting communication between

[1] A comprehensive overview of QKD network implementations is available in [1].

[2] QKDNetSim source code and supporting documentation are available at www.qkdnetsim.info.

KMSs installed on various network nodes. Our KMSs may manage keys by end-user requests after assessing their quality of service needs [11]. They reliably reserve, shrink, or merge keys as part of key management operations to stay in sync with their respective peers and respond to end-user customers.

QKD Post-Processing Application

Each QKD protocol implements a post-processing of the raw key[3] including Sifting, error estimation and reconciliation, privacy amplification, and authentication. Unlike previous versions of QKDNetSim, which implemented a simple post-processing application that imitates the QKD protocol of network activity [6], the latest version of QKDNetSim implements a full post-processing stack[4]. This further opens the door to testing various upgrades to the QKD protocol and the impact of such approaches on the network and application layers [13]. However, QKDNetSim cannot simulate quantum channel operations (polarizes, detectors, gates, etc.) and external tools can be used to mimic them reliably. One of the most common tools for these needs is Qiskit, which can be easily connected to QKDNetSim in the emulation mode. Some materials in the literature describe simulating QKD protocols (with or without post-processing) using Qiskit [14].

It is essential to note that the perfection of key and phenomena on the quantum layer plays a minor role in analyzing the network aspect. The fundamental parameters considered at the network layer are the key rate, the size of keys, and the speed of their management (storing/shrinking/combining) so that applications do not run out of required keys.

Interfaces

The key-supply interface describes access to the QKD network services by defining communication between cryptographic applications and KMS. The most popular are ETSI GS QKD 014 [9] and ETSI GS QKD 004 [10] and Cisco Secure Key Integration Protocol (SKIP).

The ETSI QKD 014 is a rest-full interface based on the HTTPS protocol and JSON-encoded data format. It includes three fundamental functions: GET_STATUS, GET_KEY, and GET_KEY_WITH_KEY_IDS. The parameters in the GET_KEY request include the size and quantity of requested keys. Key IDs that are unique and synchronized between KMS are issued to provision keys. After obtaining key IDs, the counterpart application can use the GET_KEY_WITH_KEY_IDS method to acquire identical keys. A session-based API with QoS functionalities is defined in ETSI QKD 004 specification [10]. The emulation experiments considered in this article are based on ETSI QKD 014, even though QKDNetSim supports ETSI QKD 004.

Cryptographic Applications

Two end-user (cryptographic) applications are implemented in QKDNetSim: QKD Application 004 (QKDApp004), which is based on ETSI QKD 004 specifications [10], and QKD Application 014 (QKDApp014), which is based on ETSI QKD 014 [9]. These applications can simulate how keys are used in data encryption and/or authentication procedures. Their objective is to consume keys and evaluate the overall system performance and KMS capabilities across various scenarios. They offer various configuration parameters, including encryption and/or authentication settings, data rate, packet sizes, and others based on the user's choices.

The QKDNetSim emulation included five QKD links, local KMS systems for each QKD node, and two strongSwan client-server end-user applications implemented in dedicated docker containers.

QKDNetSim Experiment Setup

QKDNetSim is developed as an independent simulation module that can be installed on the NS-3 network simulator. Thus, it allows the exploitation of all functionalities the NS-3 simulator provides. One of them is working in emulation mode, which can be realized in several ways[5].

The most popular approach is *TapBridge* mode, which enables the integration of a real host into NS-3 simulations. Specifically, the real host operates as one of the simulated network nodes, allowing real-world applications to interact seamlessly with the simulated environment. Another option is the *FdNetDevice*, which facilitates reading and writing network traffic using file descriptors. There are two modes of operation: *Emu* and *Tap*. In *Tap* mode, the device appears in the kernel as a virtual network interface (TAP device), and packets routed to this device are processed by the ongoing simulation. In *Emu* mode, the device uses raw sockets to interface with a physical network interface directly. For Emu mode to function correctly, the IP address assigned to the Emu device must belong to the same subnet as the physical network interface, which must also be configured to operate in promiscuous mode to capture traffic.

In this paper, we simulated and emulated the Czech QKD network, which consists of three major QKD nodes: Prague, Brno, and Ostrava. Given that the Czech QKD network is still under consideration, no precise data is available on the equipment that will be installed. However, calculations and simulations can provide basic information about expected performance since optical fibers are deployed and functional. The distance of optical fiber Prague-Brno is 304 km with a total attenuation of 70 dB, while the link Brno-Ostrava is 257 km with 62 dB. Implementation of direct QKD links over such distances is a challenging task. Thus, links are separated into smaller links of 100 km distance and 0.2303 db/km each: Prague1-Prague2, Prague2-Prague3, Prague3-Brno1; and smaller links of 85 km with 0.2417 db/km: Brno1-Brno2 and Brno2-Ostrava. Using a calculation of expected secret key rates [15], with the assumption of 1 GHz pulse source and efficiency of 42%, with Bob's efficiency of 50% in successfully capturing and coupling photons into the detection system, detector efficiency (20%), sifting (88%), and error correction/privacy amplification (37%) the estimated key rate over 100 km (0.2303 db/km) links is 85.083 kbps, while for links of 85 km (0.2317 db/km) key rate is estimated to 150 kbps.

[3] A correlated non-symmetric series of secret bits is created during the exchange of quantum information across the quantum channel. The term "raw key" is often used to describe this resultant sequence.

[4] Sifting is implemented as a process where incompatible measurements are discarded, leaving only data with matching bases. Error reconciliation uses Cascade, while privacy amplification relies on universal hashing to shorten the key and remove eavesdropper information. Authentication is based on SHA2 hashing to ensure the key integrity.

[5] More details available on https://www.nsnam.org/docs/models/html/emulation-overview.html.

Although the first QKD protocols were defined some 30 years ago, there is a wide range of open questions for improving and optimizing work.

Simulation Setup

In the simulation, we consider two scenarios: a simple case for validating the simulation module and a more practical case for detailed evaluation.

The first experiment uses a pair of applications to establish unidirectional end-to-end communication between Prague and Ostrava nodes. The communication is a one-time pad (OTP) encrypted without authentication. As a result, the key consumption rate should equal the application data rate. The key consumption rate (equivalent to the data rate in this scenario) is constrained by key generation rates of the underlying quantum connections.

The connection between Prague and Brno is established via two trusted-repeater nodes (Prague2 and Prague3) or three QKD links of equal length of 100 km. QKD links generate 3200—byte keys at the rate of 85.083 kbps. The default key size for keys formatted at the KMS is 512 bits or 64 bytes. Because these links are a bottleneck on the relay path between Prague and Ostrava, the data rate is expected to be limited to 85 kbps, assuming a small number of pre-generated keys.

This hypothesis is tested through a series of simulations in which the desired data rate is increased from 50 to 110 kbps in increments of 15. Because of the high data (i.e., key consumption) rate, we modified the cryptographic application to obtain keys at the desired rate rather than relying on hard re-keying. Simulations ran for 100 seconds, with QKD devices operational from 0 seconds to generate keys in advance while application data traffic started at 10 seconds. The application sends 64—byte packets (i.e., requires keys of 64 bytes in size to encrypt 64 byte of data). In addition, it requests two encryption keys from the KMS in a single query.

The second simulation experiment investigates the performance of secure communication with multiple active parallel data sessions between Prague and Ostrava. We simulate a number of parallel AES—256 encrypted sessions [50, 100, 150, 200, 250]. The sessions are unidirectional, without authentication, with data transmitted from Prague to Ostrava at 24 kbps. The size of transmitted packets is 3000 bytes. The key lifetime for each session is set to match the 1-second rekeying interval.[6] The application requests a single key per query.

It is assumed that half of the available key rate is allocated to secure traffic from Prague to Ostrava. The other half is reserved for the reverse direction and provisioning requests between Prague and Brno and Brno and Ostrava. As a result, the total key consumption for active sessions should not exceed 42.5 kbps (half the rate of the bottleneck links).

Simulations ran for 300 seconds (equivalent to 5 minutes of running time), with QKD devices operational from 0 seconds and application data traffic starting at 10 seconds.

Emulation Setup

As shown in Fig. 2, distinct docker containers were installed on a single host computer for emulation objectives. Containers are executable, standalone, lightweight software packages with all the components needed to run a program. Code, runtime, system tools, libraries, and settings are among them. The main distinction between virtual machines and containers is that the former only virtualizes software layers above the operating system. In contrast, the latter virtualizes an entire computer to the hardware layers. Docker runs apps within its containers and offers consistency and compatibility across various computing settings, from a high-end data center to a single PC. Using the docker environment makes it possible to simulate different scenarios more flexibly. It is possible to run multiple containers with different applications simultaneously that will generate or request keys[7].

a) QKDNetSim and Docker Organization: QKDNetSim was installed in a separate docker container. As shown in Fig. 2, we used *EmuFdNetDevice* for external communication with other docker containers and the host. Every part of the QKD ecosystem has been running in the same network subnet. The QKDNetSim emulation included five QKD links, local KMS systems for each QKD node, and two strongSwan client-server end-user applications implemented in dedicated docker containers. Since docker containers are implemented on the same host device, they can be implemented in host or network mode. When running docker containers in host mode, they share the host's network stack and interfaces directly. Modern NICs and operating systems often offload checksum calculations to the hardware to improve performance. But, when a packet is intercepted by software before the NIC calculates its checksum, it appears to have an invalid checksum. NS-3 might receive such packets before the NIC has a chance to compute the checksum, resulting in "bad checksum" errors. In that case, it is necessary to ensure checksum offloading is turned off on the host's network interfaces using tools such as *ethtool*. When a docker container runs in bridge network mode, it gets its network namespace and virtual network interface. Also, it is isolated from the host network and other containers since it communicates via a virtual bridge. It offers a separate network namespace, requires port mapping, better isolation, and slightly lower performance.

b) QKDNetSim and Qiskit: For each QKD link, 256-bit keys with estimated rates were emulated using Qiskit[8] and pushed to QKDNetSim KMS. The settings are adjusted so that Qiskit generates smaller keys, allowing faster execution. However, the default key size on KMS was set to 2048 bits. This means that KMS will combine the collected keys of smaller sizes through subprocess *merge* to form a larger key with a unique identifier [6]. The keys were stored in local buffers and marked ready for usage.

c) IPsec and QKD: StrongSwan client and server applications were implementing IKEv2/IPSec VPN with a pre-shared key (PSK). Since strongSwan does not have the option of refreshing IPSec keys based on the amount of transferred traffic, it is possible to use refreshing based on

[6] In current security frameworks, key lifetimes are typically much longer, often spanning several hours. To enhance security, the general goal is to reduce the key lifetime. In this simulation, we assume a high-security scenario where keys are renewed every second.

[7] This also helps optimize computational resources since post-processing is performed in separate containers.

[8] The python BB84 code generating keys using Qiskit is available at www.qkdnetsim.info/qiskit_bb84.txt.

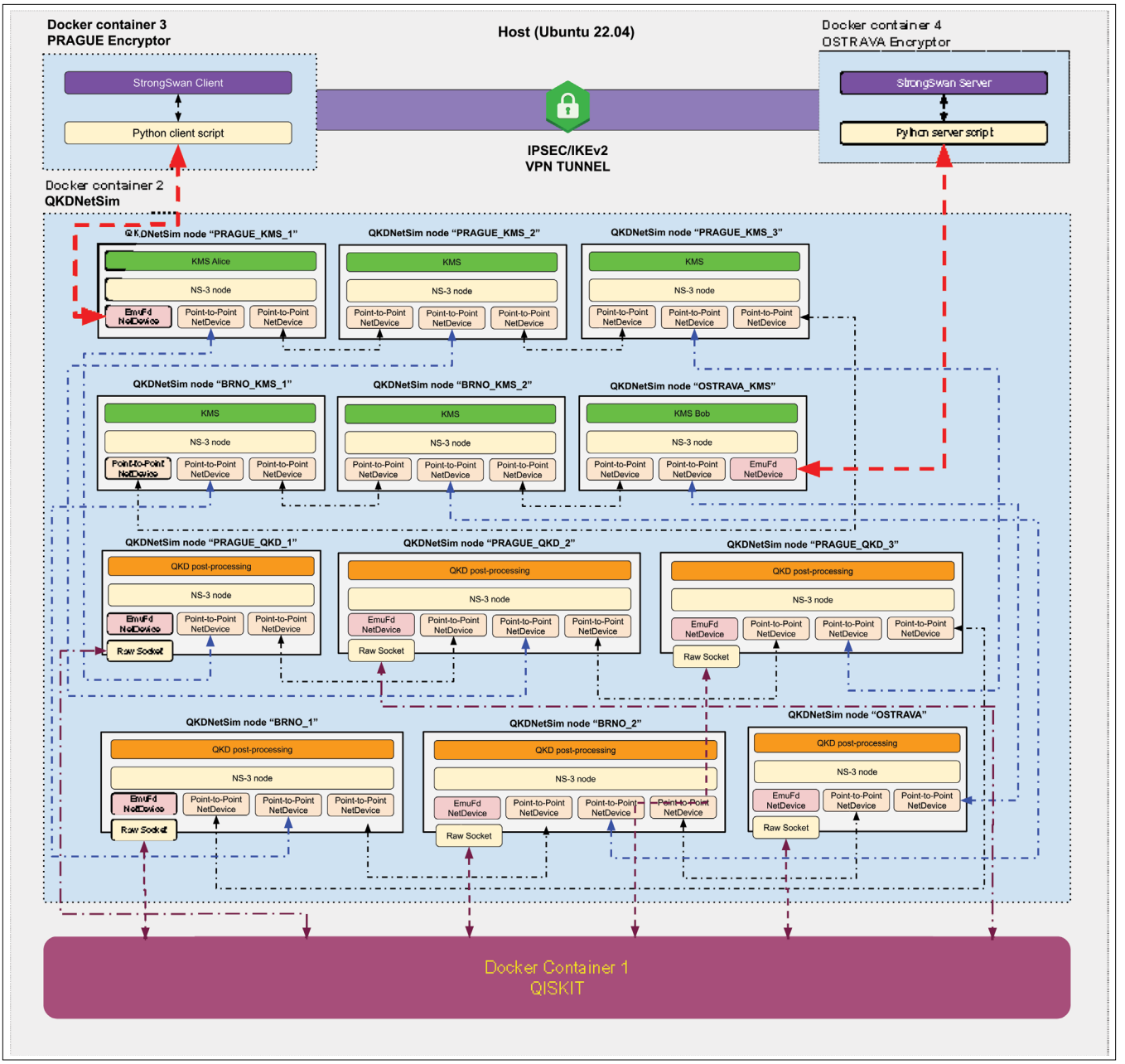

FIGURE 2. An emulated Czech QKD network that includes the entire QKD ecosystem with the processes of generation, storage/management, and consumption of keys. Multiple docker containers are installed on a single host machine. Qiskit is installed in the lowest container, generating keys sent to the middle container where QKDNetSim with separate EmuFdNetDevice devices enable emulation connections and key processing. The strongSwan client and server applications are installed in top containers, and an IPSec/IKEv2 VPN tunnel is established between them. The keys for VPN sessions are obtained from QKDNetSim KMSs using Python scripts that replace PSK keys in strongSwan config files.

elapsed time. For these needs, a bash script has been implemented that is periodically executed every 60 seconds[9] by performing the following steps: using *wget* tries to get a key that was generated by Qiskit and stored on QKDNetSim's KMS; If a 200 OK HTTP response is received from KMS, the created key is stored in */etc/ipsec.secrets* and strongswan is instructed to initiate the establishment of a new IPsec session with the fetched key; if the key is not obtained, the session cannot be established and will be tried again in the next iteration. IPsec session duration was set to 20 seconds. However, the last session remains active if the key is unavailable. This session will be terminated when the next key is obtained, and a new session is successfully established.

Results

Fig. 3-left presents the results of the first simulation experiment, focusing on an OTP-encrypted session. It compares the generated, relayed, and served key bits across different data rates. To enhance clarity of figures, a negative offset of 50,000 bits was applied to the served key bits (which are generally lower than the relayed key bits) as the relayed and served lines overlap.

The number of relayed key bits generally surpasses the served ones, as the key manager

[9] If OTP were used, at least one key bit is used for each bit of data. At least because encryption is considered only without additional keys that would be used for authentication. Reducing the frequency of sending key requests and increasing the number of keys requested with each request is possible. A balance needs to be found so that most requests are successful, and frequency of 60 seconds was chosen in this experiment.

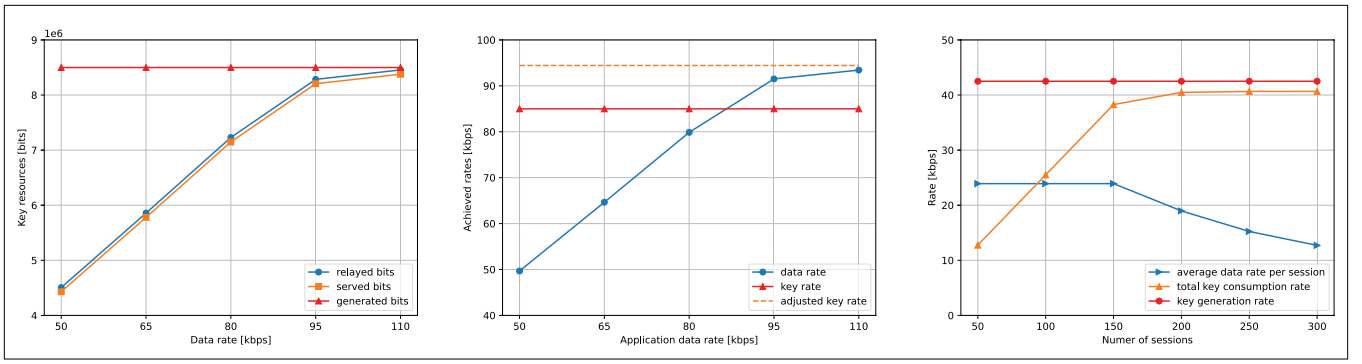

FIGURE 3. left) The amount of generated, relayed, and served key bits for the varying application traffic intensity (first simulation experiment); middle) The comparison of achievable and desired data rates (first simulation experiment). The achievable data rate surpasses the key generation rate of the bottleneck links because QKD links accumulated sufficient key resources during the initial 10 seconds; right) Achieved data rates per session and overall key consumption rate compared to the number of simultaneous AES—256 encrypted sessions.

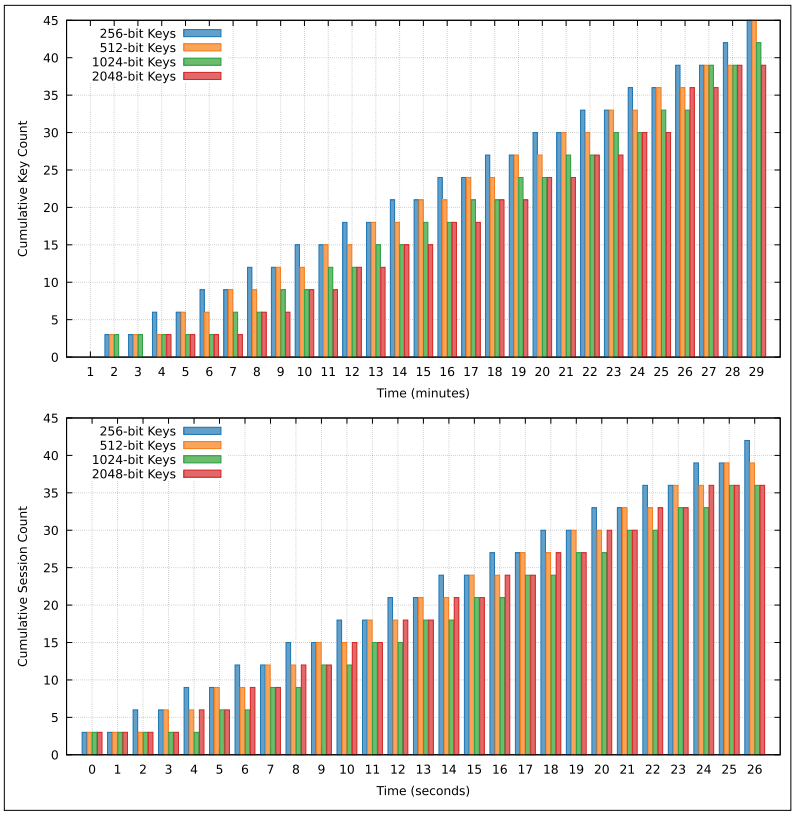

FIGURE 4. Top - The cumulative count of fetched keys versus time. Bottom - The cumulative count of newly established IPsec/IKEv2 sessions versus time.

proactively monitors global key availability. It ensures continuous distribution of new keys whenever the total relayed key bits drop below the threshold of 64,000 bits. The relayed key bits are equal across all QKD links.

The relayed and served key bit amounts ideally follow a ramp-like progression. However, as these values near the total generated key bits, they eventually reach saturation. The amounts of relayed and served key bits never exceed the total generated key bits of the bottleneck links.

Fig. 3-middle compares the achieved data rates with the desired ones, illustrating that the achieved rates reach saturation at levels slightly higher than the key generation rate of the bottleneck links. This occurs because the QKD links operated for 10 seconds before the application began consuming keys, enabling the accumulation of substantial key resources. These resources temporarily allowed the application data rate to exceed the key generation rate on the path. The dotted line in the graph represents the adjusted key generation rate (of bottleneck links) calculated over 90 seconds (instead of the entire 100 seconds).

Fig. 3-right illustrates the results of running multiple parallel sessions encrypted using AES-256. It demonstrates that, under the specific configuration described in the section "Simulation Setup," up to 150 simultaneous sessions can operate without any performance degradation. Increasing the number of sessions results in a lower average data rate per session. This is because increasing the number of sessions leads to an increase in overall key demand. Such key demands cannot be met due to the limited key generation rate of 42.5 kbps for the Prague-Ostrava direction. To support more sessions, either the key generation rate of the QKD links should be increased, or the overall key demand should be reduced. The key lifetime can be extended to minimize key demand, allowing sessions to send more data encrypted with the same key.

Fig. 4 (top) shows the cumulative count of fetched keys versus time. It can be noticed that the 256-bit keys were fetched more frequently since their generation and delivery time is the lowest. It is important to note here the impact of subprocess *merge* on KMS, which combines keys of smaller sizes to form a larger key. As shown in Fig. 4 (bottom), the smaller number of obtained keys was reflected in the smaller number of formed IPsec/IKEv2 sessions. Either to reduce the length of keys used for establishing sessions or to reduce the frequency of obtaining keys (increasing the duration of IPsec sessions).

The implications of the size and number of available keys for meeting user requests are demonstrated even for docker containers installed on a single host computer. Therefore, our results highlight the significance of key organization based on user requirements to improve key delivery. Since the merge subprocess does not always succeed[10], keeping the keys organized ahead of

[10] The success and duration of *merge* KMS subprocess are contingent upon the synchronization of key storage systems and the lack of conflicts arising from the usage of the same keys by other applications.

time ensures that the user can access them [11]. The average throughput of established IPsec/IKEv2 sessions was 3567.3 Mbps measured using tool *iPerf3*.

Conclusion

Although the first QKD protocols were defined some 30 years ago, there is a wide range of open questions for improving and optimizing work. Additionally, there is a unique question about how to use and integrate QKD technology in everyday IP networks. Considering the high cost and sensitivity of the equipment, simulation, and emulation platforms enable the study and provide answers to critical questions regarding the network aspect.

This paper considers the national Czech QKD network. The results show that different key organization and consumption scenarios can be considered. In the emulation scenario, docker containers could be quickly cloned and installed in multiple data centers, forming more extensive and complex network topologies. Considering QKD network simulators and emulators are not currently available, QKDNetSim provides significant advantages to the QKD technology research community. QKDNetSim will facilitate the adoption of a wide range of applications inside the QKD network, contributing to the actual deployment of QKD technology. The main contribution of this article is an overview of the simulation and emulation techniques used to evaluate QKD networking.

Acknowledgment

This work was supported by the Lithuania Research Council (LMTLT) under Agreement P-ITP-24-9.

References

[1] M. Mehic et al., "Quantum key distribution: A networking perspective," *ACM Comput. Surv.*, vol. 53, no. 5, pp. 1–41, 2020.
[2] A. Dahlberg and S. Wehner, "SimulaQron—A simulator for developing quantum Internet software," *Quantum Sci. Technol.*, vol. 4, no. 1, Sep. 2018, Art. no. 015001.
[3] T. Coopmans et al., "NetSquid, a NETwork simulator for QUantum information using discrete events," *Commun. Phys.*, vol. 4, no. 1, pp. 1–15, Jul. 2021.
[4] X. Wu et al., "SeQUeNCe: A customizable discrete-event simulator of quantum networks," *Quantum Sci. Technol.*, vol. 6, no. 4, Oct. 2021, Art. no. 045027.
[5] R. Satoh et al., "QuISP: A quantum internet simulation package," 2021, *arXiv:2112.07093*.
[6] E. Dervisevic, M. Voznak, and M. Mehic, "Large-scale quantum key distribution network simulator," *J. Opt. Commun. Netw.*, vol. 16, no. 4, p. 449, 2024.
[7] M. Mehic et al., *Quantum Key Distribution Networks: A Quality of Service Perspective*. Cham, Switzerland: Springer, 2022.
[8] M. Mehic et al., "Emulation of quantum key distribution networks," *IEEE Netw.*, vol. 39, no. 1, pp. 116–123, Jan. 2025.
[9] *014. Quantum Key Distribution (QKD); Protocol and Data Format of REST-Based Key Delivery API*, Standard ETSI ISG QKD, 2019, pp. 1–22, vol. 1.
[10] *004—Quantum Key Distribution (QKD); Application Interface*, Standard ETSI ISG QKD, 2020, pp. 1–22, vol. 1.
[11] E. Dervisevic et al., "Quantum key storage for efficient key management," 2024, *arXiv:2408.04598*.
[12] M. Mehic et al., "Quantum cryptography in 5G networks: A comprehensive overview," *IEEE Commun. Surveys Tuts.*, vol. 26, no. 1, pp. 302–346, 1st Quart., 2024.
[13] E. Dervisevic, M. Voznak, and M. Mehic, "Bases selection with pseudo-random functions in BB84 scheme," *Heliyon*, vol. 10, no. 1, Jan. 2024, Art. no. e23578.
[14] M. H. Saeed et al., "Implementation of QKD BB84 protocol in Qiskit," in *Proc. 19th Int. Bhurban Conf. Appl. Sci. Technol. (IBCAST)*, Aug. 2022, pp. 689–695.
[15] A. Shields, "Performance limits for quantum key distribution networks," in *Proc. Presentation ITU Workshop Quantum Inf. Technol.*, Shanghai, China, 2019, pp. 5–7.

Biographies

Miralem Mehic (miralem.mehic@ieee.org) (Senior Member, IEEE) (miralem.mehic@ieee.org) received the Ph.D. degree in telecommunications from the VSB-Technical University of Ostrava, Czechia. He also studied at the AGH University of Science and Technology, Kraków, Poland; Alpen-Adria-Universität Klagenfurt, Austria; and the Department of Digital Safety and Security Business Units–Optical Quantum Technology, Austrian Institute of Technology (AIT), Vienna and Klagenfurt, Austria. His research interests include quality of service and management of QKD networks, with a focus on real-time traffic and the utilization of network resources.

Emir Dervisevic was born in Berlin, Germany, in 1995. He is currently pursuing the Ph.D. degree with the Department of Telecommunications, Faculty of Electrical Engineering, University of Sarajevo. Since 2020, he has been a part of the International Scientific Research Projects Horizont 2020–Open European Quantum Key Distribution Testbed (OPENQKD) and the NATO SPS MYP G5894–Quantum Cybersecurity in 5G Networks (QUANTUM5). He is actively developing the quantum key distribution network simulation module. His research interests include quantum key distribution networks, network management, network security, and cryptography.

Peppino Fazio (Member, IEEE) received the Ph.D. degree in electronics and communications engineering from the University of Calabria (UNICAL), Italy, in January 2008, and the Habilitation degree. From May 2008 to September 2008, he was a Visiting Research Fellow with the GRC Research Group, UPV of Valencia, Spain. He was an Associate Professor in April 2017. After 12 years of Assistant Professorship in UNICAL, he is currently an Associate Professor with the Department of Molecular Sciences and Nanosystems (DSMN), Ca' Foscari University of Venice (UNIVE). He has also been collaborating with the VSB-Technical University of Ostrava, Czechia, since 2017, as a Senior Research Fellow. He has been a member of various start-up companies. He is the co-author of more than 135 articles (55 in international journals) and one book, all indexed in Scopus and/or WoS. He co-advised more than 70 B.S. and M.S. students and three Ph.D. students. His research interests include mobile communication networks, IP QoS architectures, wireless and wired networks, mobility modeling for WLAN environments, mobility analysis for prediction purposes, routing, vehicular networking, MANET, VANET, and quantum key distribution networks. He was a recipient of national and international awards (e.g., the Intel Business Challenge Award in 2013).

Miroslav Voznak (Senior Member, IEEE) received the Ph.D. degree in telecommunications in 2002 and the Habilitation degree in 2009. He is a currently a Professor with the Faculty of Electrical Engineering and Computer Science, VSB-Technical University of Ostrava. He was appointed as a Full Professor in 2017. According to WoS, he has published more than 200 articles in SCIE journals, such as IEEE Communications Surveys and Tutorials, IEEE Journal on Selected Areas in Communications, and IEEE Communications Magazine. He participated in seven projects within EU funding programs, mostly as an institutional coordinator and more than 20 national projects. His research interests include the IoT, QoS/QoE, wireless networks, network security, and big data analytics in networks. Since 2020, he has been ranked regularly among the World's Top 2% Scientists (Career Impact DB) by Stanford University in the subfield networking and telecommunications.