entropy

Article

Linking QKD Testbeds across Europe

Max Brauer [1], Rafael J. Vicente [2], Jaime S. Buruaga [2], Rubén B. Méndez [2], Ralf-Peter Braun [1], Marc Geitz [1,*], Piotr Rydlichkowski [3], Hans H. Brunner [4], Fred Fung [4], Momtchil Peev [4], Antonio Pastor [5], Diego R. Lopez [5], Vicente Martin [2] and Juan P. Brito [2,*]

[1] T-Labs, Deutsche Telekom AG, 10781 Berlin, Germany; brauermax@gmx.de (M.B.); ralf-peter.braun@t-online.de (R.-P.B.)
[2] DLSIIS and Center for Computational Simulation, Universidad Politécnica de Madrid, 28660 Madrid, Spain; rafaelj.vicente@upm.es (R.J.V.); j.saezdeburuaga@upm.es (J.S.B.); ruben.bmendez@upm.es (R.B.M.); vicente@fi.upm.es (V.M.)
[3] Poznan Supercomputing and Networking Center, 61-139 Poznań, Poland; prydlich@man.poznan.pl
[4] Munich Research Center, Huawei Technologies Duesseldorf GmbH, 80992 Munich, Germany; hans.brunner@huawei.com (H.H.B.); fred.fung@huawei.com (F.F.); momtchil.peev@huawei.com (M.P.)
[5] Telefónica gCTIO/I+D, 28050 Madrid, Spain; antonio.pastor@telefonica.com (A.P.); diego.r.lopez@telefonica.com (D.R.L.)
* Correspondence: marc.geitz@telekom.de (M.G.); juanpedro.brito@upm.es (J.P.B.); Tel.: +49-1715408754 (M.G.); +34-910673073 (J.P.B.)

Abstract: Quantum-key-distribution (QKD) networks are gaining importance and it has become necessary to analyze the most appropriate methods for their long-distance interconnection. In this paper, four different methods of interconnecting remote QKD networks are proposed. The methods are used to link three different QKD testbeds in Europe, located in Berlin, Madrid, and Poznan. Although long-distance QKD links are only emulated, the methods used can serve as a blueprint for the secure interconnection of distant QKD networks in the future. Specifically, the presented approaches combine, in a transparent way, different fiber and satellite physical media, as well as common standards of key delivery interfaces. The testbed interconnections are designed to increase the security by utilizing multipath techniques and multiple hybridizations of QKD and post-quantum cryptography (PQC) algorithms.

Keywords: quantum networks; quantum communications; QKD; quantum cryptography

check for updates

Citation: Brauer, M.; Vicente, R.J.; Buruaga, J.S.; Méndez, R.B.; Braun, R.-P.; Geitz, M.; Rydlichkowski, P.; Brunner, H.H.; Fung, F.; Peev, M.; et al. Linking QKD Testbeds across Europe. *Entropy* **2024**, *26*, 123. https://doi.org/10.3390/e26020123

Academic Editors: Rosario Lo Franco and Osamu Hirota

Received: 9 November 2023
Revised: 22 December 2023
Accepted: 24 January 2024
Published: 31 January 2024

1. Introduction

Shor's algorithm [1] for quantum computers can solve the fundamental mathematical problems on which currently used public-key cryptography (PKC) is based in polynomial time. Therefore, the security PKC offers will be lost when mature quantum computers are available. It should be noted that the security of traditional PKC is based on a complexity assumption: the fundamental problems enabling it, such as the factoring of large integers, the discrete-logarithm problem, and the elliptic-curve discrete-logarithm problem (rooted itself in the algebraic structure of elliptic curves over finite fields), are not solvable in polynomial time, which makes performing some inversion tasks intractable. This assumption of "non-polynomiality" has been demonstrated to be false for quantum computers, specifically by the mentioned Shor's algorithm.

A presently developed "remedy" technology is post-quantum cryptography (PQC) [2], which is a quantum-safe version of PKC. PQC is a broad concept but at its core lie public key algorithms that, similar to their "classical" counterparts, are based on mathematical problems that cannot be inverted in polynomial time by any known algorithm including Shor's. In this sense they are "Shor resistant" and are assumed to be intractable also for any other quantum algorithms. The mentioned assumptions are underpinned by mathematical-complexity arguments in their versions with regard to quantum computing. The specific status of PQC development is represented by three protocols in the draft standardization phase, belonging to two classes: key agreement ones based on quantum-safe public key encryption-decryption of a secret key, key encapsulation mechanisms (KEM), in what follows PQC KEM, and digital signature ones (SIG), in what follows PQC SIG [3].

Quantum key distribution (QKD) is yet another technology that provides symmetric keys to two distant parties but employs protocols which are information-theoretic secure (ITS), the utmost level of security. These protocols are based on fundamental features of quantum mechanics and are not vulnerable against any type of adversaries, irrespective of their resources. This includes eavesdroppers equipped with quantum computers.

Present QKD protocols and implementations have only a finite reach. This reach can be extended by using chains or networks of trusted nodes, e.g., trusted satellites. These reach extensions follow protocols that are also ITS, as is the original QKD, albeit assuming the enforcement of additional non-cryptographic (organizational) measures, which allow absolute trust in the integrity of the intermediate nodes. In the following these extension protocols are called "key forwarding". Such extensions have been intensively tested in the European Union and China [4–8].

Like any other technology, QKD, QKD-network, and PQC implementations are weaker than the respective ideal protocols from the point of view of security. The implementations are vulnerable to so-called "side channels" that generally represent the differences between implemented and ideal protocols. QKD protocol security is a mathematical theorem, which follows from several (sufficient) conditions that cannot realistically be guaranteed in QKD implementations. The discussion of such issues is beyond our present objective and is generally addressed by the procedures of certification of security technologies [9–11]. In this text, it is acknowledged that an implementation is always less secure than an underlying protocol by using the term "side channels" to denote all differences.

The goal of this paper is to demonstrate a feasible connection of present-day European metropolitan-area QKD networks, as developed in OpenQKD and planned in EuroQCI [12,13], before long-distance QKD is available and chains of trusted nodes, quantum repeater links, or QKD satellites are installed in Europe. In this work, the long-distance QKD connections are only emulated QKD links.

Independently, all links were realized as dual-technology key generation, based on (emulated) QKD and PQC. This is the maximum that can be demonstrated without true long-distance QKD. We, therefore, present a blueprint of a future design featuring higher security and simulate it by emulated QKD links where true ones are not yet technically available. This shows a potential utilization of two methods in parallel [14] and emulates a "crypto-agility" realization, the secure combination or hybridization of different security technologies. Such a principal strategy is generally reasonable in a dynamic field, in which the security of protocols can be quickly re-evaluated. Specifically, the combination of multiple security-protocol implementations, that might be vulnerable to yet unknown, but most likely different, side channels, was demonstrated in this work. Such a scheme reduces risks, especially against non-coinciding side channels that might originally jeopardize the implementation of a single protocol.

It must be emphasized that only the parallel combination of QKD and PQC can increase security, while the sequential combination of protocols can only lead to a security reduction. In the case of a parallel implementation, a potential attacker must attack all vulnerabilities independently (except for their potential common intersection). In the case of a sequential combination, it is sufficient to attack the weakest element from the perspective of the attacker. For a more detailed clarification, some combination possibilities and the resulting security level are listed in Tables 1 and 2. The benefit of parallel combinations can be found in Table 1, while Table 2 shows the issues of serial combinations. The fourth row in Table 1 is introduced because it is often seen as convenient for managing the otherwise tedious authentication issues in large and evolving QKD networks. QKD will only be ITS if the keys required for message integrity (message authentication) in post-processing are generated using ITS mechanisms, e.g., with pre-shared keys (PSK) for the first QKD round and using a portion of the QKD generated key subsequently [15,16]. Using PQC for authentication limits the security level of the key generation to that of this authentication method and the ITS advantage of QKD is lost. In that case one could use PQC directly for key generation. In the end, the adopted level of security must be a conscious decision of the network operator.

Table 1. Security level of various combinations of different key-exchange technologies in parallel in a symbolic set-theoretic representation. This could be, e.g., discrete variable (DV) and continuous variable (CV) QKD protocols in pure QKD networks, PQC KEM and PQC SIG algorithms in pure PQC networks, and combinations of pairs of technologies for different tasks in mixed networks. QKD implementations are ITS minus ($\setminus$) side channels (SCs), while the security level of PQC is based on mathematical-complexity (MC) minus SCs. The combined security level of parallel key generation is the union ($\cup$) of the individual security levels. For key exchanges of the same security level this simplifies to this level but minus the intersection ($\cap$) of the SCs. If not otherwise noted, QKD with PSK will be assumed. Note that this classification is independent of whether the different technologies are applied over the same or different network paths.

ID	Key Exchange 1	Security	Key Exchange 2	Security	Combined Security Level
1	QKD_1	$\text{ITS}\setminus\text{SCs}_1$	QKD_2	$\text{ITS}\setminus\text{SCs}_2$	$\text{ITS}\setminus\{\text{SCs}_1 \cap \text{SCs}_2\}$
2	PQC_1	$\text{MC}\setminus\text{SCs}_1$	PQC_2	$\text{MC}\setminus\text{SCs}_2$	$\text{MC}\setminus\{\text{SCs}_1 \cap \text{SCs}_2\}$
3	QKD_1	$\text{ITS}\setminus\text{SCs}_1$	PQC_2	$\text{MC}\setminus\text{SCs}_2$	$\{(\text{ITS}\setminus\text{SCs}_1) \cup (\text{MC}\setminus\text{SCs}_2)\}$
4	QKD_1 + PQC SIG	$\text{MC}\setminus\text{SCs}_1$	QKD_2 + PQC SIG	$\text{MC}\setminus\text{SCs}_2$	$\text{MC}\setminus\{\text{SCs}_1 \cap \text{SCs}_2\}$

Table 2. Security level of various combinations of different key-exchange technologies in series in a symbolic set-theoretic representation. The combined security level of serial key generation is the intersection of the individual security levels. For the examples in this table, this is equal to the respectively lower security level minus the union of the SCs.

ID	Key Exchange 1	Security	Key Exchange 2	Security	Combined Security
1	PQC_1	$\text{MC}\setminus\text{SCs}_1$	QKD_2	$\text{ITS}\setminus\text{SCs}_2$	$\text{MC}\setminus\{\text{SCs}_1 \cup \text{SCs}_2\}$
2	QKD_1	$\text{ITS}\setminus\text{SCs}_1$	QKD_2	$\text{ITS}\setminus\text{SCs}_2$	$\text{ITS}\setminus\{\text{SCs}_1 \cup \text{SCs}_2\}$

At the moment, the cryptographically most secure combination is a direct, any-to-any PQC key exchange combined with an any-to-any QKD key exchange relying on key forwarding. For this paper, the PQC links were established directly between any two nodes, irrespective of metro-network boundaries. In contrast, the QKD key generation between nodes belonging to different metropolitan-area QKD networks used key forwarding over emulated long-distance QKD connections only between selected border nodes.

Additionally, a two-path approach, i.e., one that utilizes two different paths, will reduce risks even if a single implementation of a single technology is used. Risks will be further reduced if different implementations and, moreover, different technologies, deployed over different paths, are used. In the following these approaches will be dubbed "two-factor" ones, whereby in reality multiple factors are involved. Actual connections can be complex combinations of different paths and the different options listed in Tables 1 and 2. In this work, most PQC links were realized as a two-technology and two-path combination over a ground link and a satellite link, while the QKD key forwarding retrieved keys from QKD modules of different vendors in series but also in parallel.

The QKD-network testbeds used for the demonstration are located in Berlin (Germany), Madrid (Spain), and Poznan (Poland). These are symbolically shown in Figure 1. Each testbed represents a metropolitan-area quantum-optical network. The distances Berlin–Madrid and Poznan–Berlin are 1860 km and 230 km, respectively. The testbeds are too far apart for state-of-the-art, point-to-point QKD links, which is an additional motivation for the approach outlined in this Section (see option 1 in Table 2).

The long-distance QKD links in this work have been emulated using PQC KEM crypto protocols, as discussed in detail below. For transmission purposes, either normal (classical) terrestrial communication or a combination of classical terrestrial and classical satellite communication have been used.

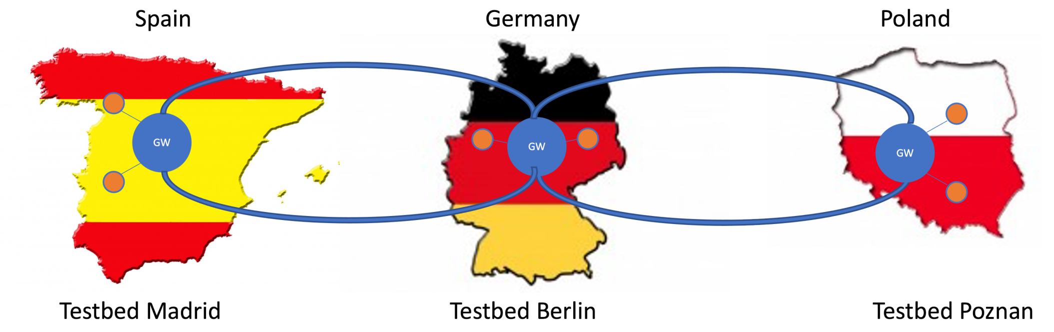


Figure 1. Connection of the quantum testbeds in Madrid (**left**), Berlin (**middle**), and Poznan (**right**) with emulated long-distance QKD links. The key exchange is indicated by the curved blue lines, which connect dedicated QKD gateway nodes (blue circles) in each testbed. The other QKD nodes in the respective testbeds are indicated by orange circles.

2. Participating QKD Testbeds

2.1. QKD Testbed in Berlin

The QKD testbed Berlin connected several offices, institutes, and network operating centers of Deutsche Telekom in the Berlin metropolitan area [17] using dark optical fiber. The testbed utilized a typical QKD-network three-layer architecture consisting of a quantum, key-management, and an application layer but introduced PQC key exchange in the quantum layer for hybridization purposes (as discussed in Section 1). Table 3 lists the systems deployed in either architectural layer; a photo of a testbed rack hosting the equipment is shown in Figure 2.

Table 3. Equipment deployed in the quantum, key-management, and application layers of the Berlin QKD-testbed architecture.

Layers	Equipment
Quantum layer	DV-QKD systems by ID Quantique, DV-QKD systems by Toshiba, PQC key-exchange system developed by Open Quantum Safe [18]
Key-management layer	key-management system (KMS) internally developed by DT, hardware security module (HSM) by Gemalto
Application layer	L1 hardware encryptors by Adva, L3 hardware encryptors by Thales

The Berlin testbed utilized an ETSI GS QKD 014 [19] API and the encryption keys generated by the key exchange systems were imported into a hardware security module or local key store as a "single point of trust". Also, an ETSI 014 API exposed the encryption keys to application encryptors on the application layer.

2.2. QKD Testbed in Poznan

The QKD testbed in Poznań [5] was implemented based on the POZMAN and PIONIER network infrastructures. The Poznan QKD testbed is under integration with the Polish QCI infrastructure and full publication of the infrastructure is pending. PIONIER is the Polish research and education backbone network and POZMAN is the metro area research and education network in Poznań. Both infrastructures connect multiple different research and public institutions and provide several services for their users and environment, both nationally and internationally. The PIONIER network also extends to Europe and reaches important research and education locations, such as CERN in Geneva. The QKD testbed in Poznań used mainly metro area dark fibers between PSNC offices and node locations/service hubs. For specific QKD use cases, the testbed used long-distance PIONIER backbone dark fibers that connected directly to the PSNC lab and metro infrastructure. The testbed implemented similarly as in the other locations a three-layer architecture consisting of a quantum, key management, and an application layer, the latter being based on already existing network services. Table 4 lists the systems deployed in either architectural layer; a picture of a testbed rack hosting the equipment is shown in Figure 3.

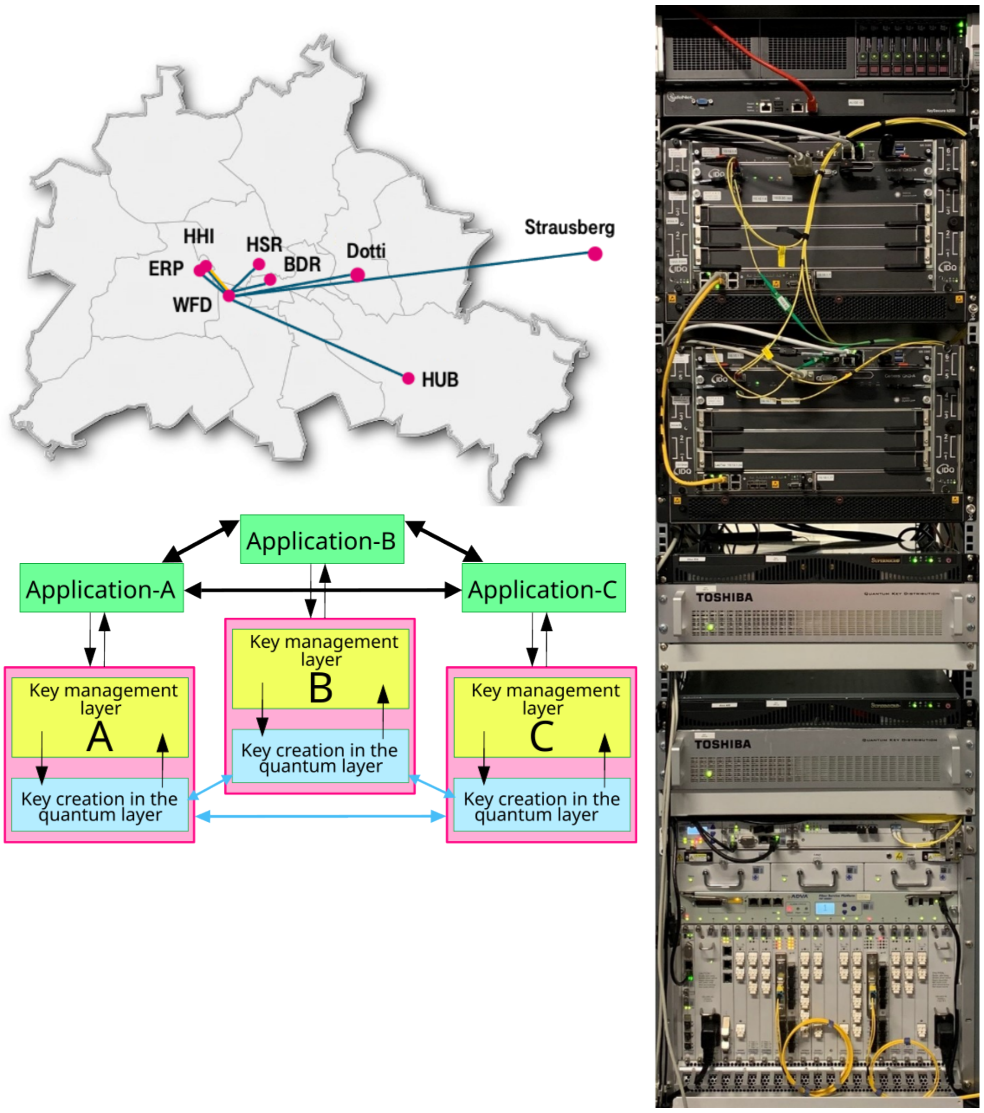


Figure 2. Dark-fiber topology in the Berlin metropolitan-area testbed (**top left**), deployed three-layer architecture (**bottom left**), rack hosting QKD modules, servers, HSMs, and encryptors (**right**). For more details the reader is referred to [4].

Table 4. Equipment deployed in the quantum, key-management, and application layer of the Poznań QKD-testbed architecture.

Layers	Equipment
Quantum layer	DV-QKD systems by ID Quantique, DV-QKD systems by Toshiba, PQC key-exchange system from Open Quantum Safe [18]
Key-management layer	Key-management system by ID Quantique, key-management system by Toshiba, security module implemented with open software solutions, OpenDNSSEC
Application layer	L1 hardware encryptors by Adva, L3 hardware encryptors by SENETAS

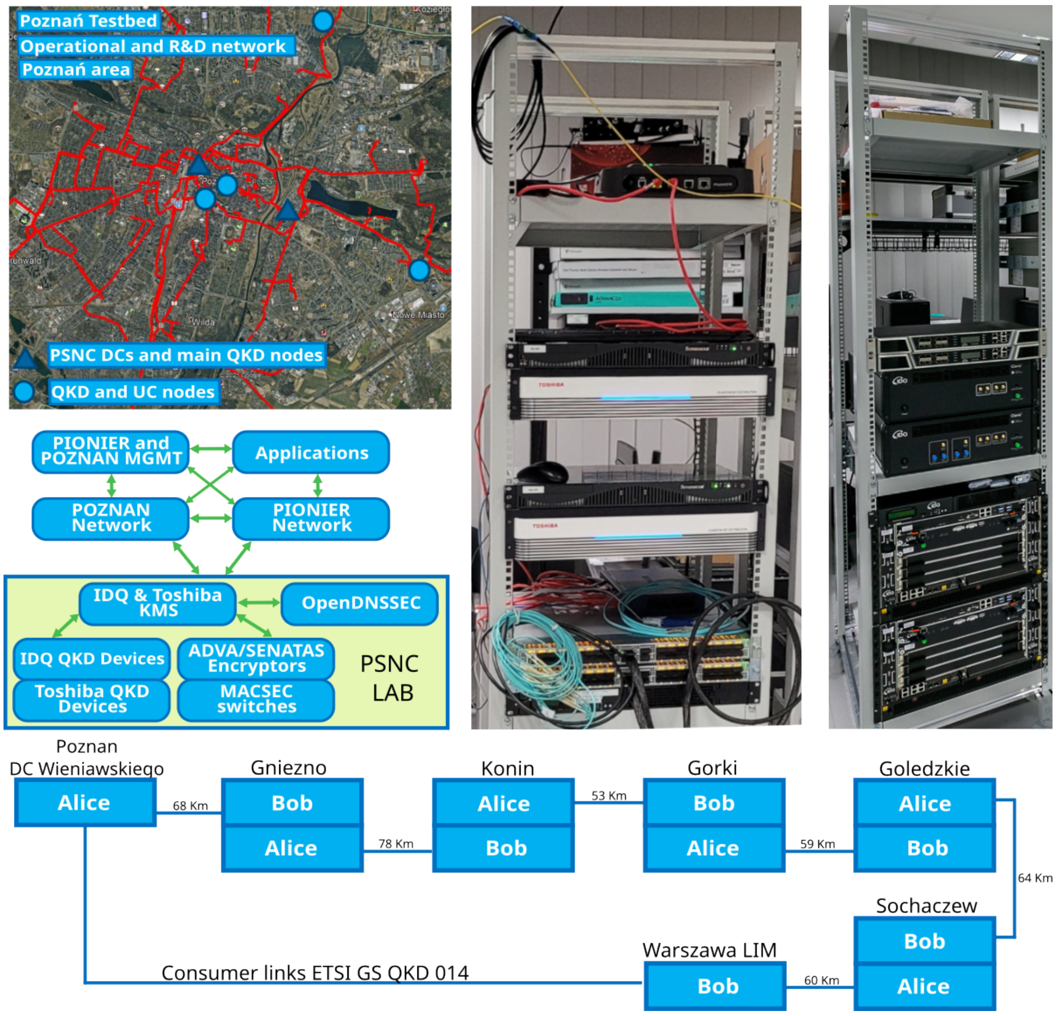


Figure 3. PSNC testbed together with the dark fiber topology in the Poznań metropolitan area (**top left**), deployed layer architecture for both metro and backbone networks—POZMAN and PIONIER (**center left**), racks hosting QKD equipment and encryptors (**right**), additionally connected trusted-node configuration of the long-distance QKD link between Poznań and Warsaw (**bottom**) [20].

The Poznań testbed used mainly an ETSI GS QKD 014 API and the encryption keys generated by the key exchange systems were imported into the key management system and subsequently directly consumed by services and/or applications or possibly by the open software security module. Also, an ETSI 014 API exposes the encryption keys to application encryptors on the application layer. As an alternative, the testbed also implemented the novel ETSI GS QKD 020 API [21], currently under development. However, due to incomplete support in hardware, it was not fully functional. The PQC protocols implemented in the PSNC testbed were based on the same mechanisms as in the Berlin testbed system and were developed using Open Quantum Safe [18]. This approach guaranteed the best compatibility and optimal convenience to quickly implement any changes in the configuration. The Poznan testbed was connected with a long-distance physical QKD link chain Poznan–Warsaw, presented in Figure 3. It is 380 km long with a trusted node configuration and using dedicated dark fibers for the quantum channel. At the ends of these two consumer applications are linked that have been configured to access the generated key material using the ETSI GS QKD 014 API.

This setup enabled connection of the metro and long-distance (national-scale) QKD services.

2.3. QKD Testbed in Madrid

The Madrid Quantum Network or Madrid Quantum Communication Infrastructure (MadQCI) testbed [6] was conceived as a field trial of a real QKD production network, connecting the infrastructures of two Spanish telecoms. On one side is Telefonica, the major telecom operator in Spain, and on the other is REDIMadrid, that connects all the research centers and Universities in the Madrid region. It combines dark fibers and real production channels simultaneously. We also note that the Madrid QKD testbed logically integrates a single QKD link, situated at the Munich Research Center of HWDU, separated by a (direct line) distance of 1445 km from Madrid. It is shown as a dot in Figure 4, connected to Madrid by a grey line.

The MadQCI follows a QKD software defined networking (SDN) design principle [7], allowing easy integration of QKD required hardware, such as QKD modules, encryptors, QRNGs, etc. (Note that the QKD network architecture design in this case does not follow the ITU-T standards to this end—ITU-T Y.3800 [22] and subsequent documents—as do the testbeds in Berlin and Poznan but relies on the quoted SDN alternative paradigm). The MadQCI is based also on the trusted node approach [23] implemented in Madrid as a set of disaggregated, but securely interconnected, hardware and software components, each of them with clear and strict responsibilities. Figure 4 shows this approach and Table 5 lists the hardware and software systems deployed on this testbed.

As this was implemented in Berlin and Poznan, the PQC infrastructure of the Madrid testbed also implements the Open Quantum Safe [18] library. The Madrid PQC implementation exposes the ETSI GS QKD 004 interface [24], with the quality of service based on the key rate. It uses Kyber 1024 as the key encapsulation mechanism, but NTRU is also possible.

Table 5. Equipment deployed in MadQCI. Note that the layout of this table is unlike the preceding two ones, as a consequence of the differing architecture of the Madrid QKD network.

Planes	**Equipment and Software Deployment**
Quantum forwarding plane	DV-QKD systems by ID Quantique, DV-QKD systems by Toshiba, CV-QKD systems by Huawei Technologies Duesseldorf (HWDU), PQC key-exchange developed by UPM, key-forwarding and key-store software modules developed by UPM
Control plane	Modules developed for the Madrid QKD-SDN software stack by UPM
Application plane	L1 hardware encryptors by Adva, L2 hardware encryptors by Rohde & Schwarz, L3 software encryptors developed by UPM, further security applications developed by UPM and Telefonica

Note that this table differs in form from the previous one due to the distinct QKD network architecture (an SDN one) used in MadQCI.

For key delivery ETSI GS QKD 004 API and ETSI GS QKD 014 API are used in a transparent way in the south- and north-bound interfaces of the UPM control (or intermediate) layer. It allows delivering key material to the application layer (e.g., the encryptors) over an appropriate interface independently of the key delivery interface used by the QKD devices.

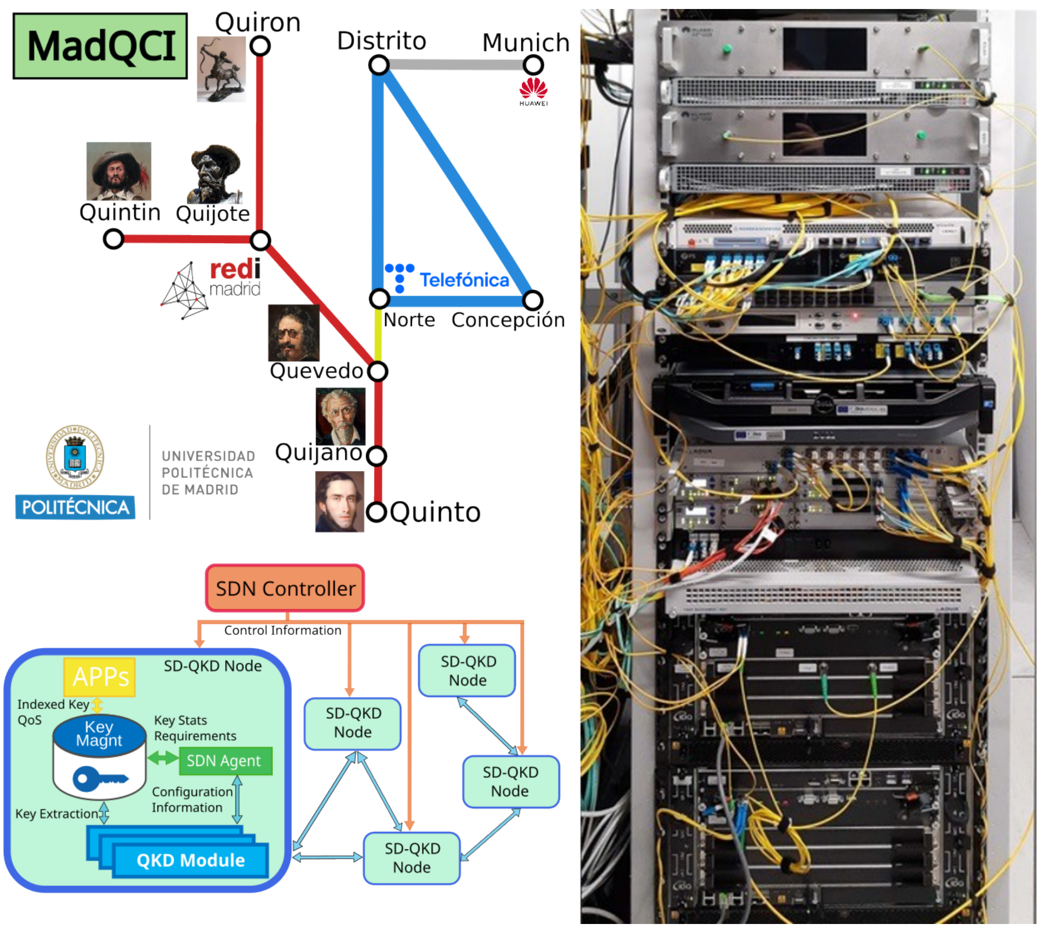


Figure 4. Madrid network—topology (**top left**); Madrid network—functional diagram (**bottom left**); Quijote ode (**right**). For more details the reader is referred to [6].

Due to its SDN nature, the MadQCI is highly based on open interfaces and standard tools, schemes and protocols, also typically used by the telecommunications industry. The same approach was followed with respect to the QKD modules, and important standards were integrated into the network. In addition to the aforementioned ETSI GS QKD 004 and 014 APIs for key delivery, ETSI GS QKD 015 [25] is used for interfacing the SDN-QKD node control agent and the SDN logically centralized network controller that orchestrates any key delivery between the end-to-end (E2E) nodes.

3. Key-Generation Technologies

In this section, we address different key generation technologies and paths that we have deployed. We start with a basic description of QKD key exchange systems and their integration into the telecommunication provider infrastructure. Due to the multiple network and QKD providers, protocols, and control systems involved in this project [4,6], we address how to exchange keys via QKD border nodes, via quantum-safe border nodes (PQC), and for further increased security, via a quantum-safe (PQC) network of disjoint network paths. Here, the availability of QKD in each separate network is taken for granted.

3.1. QKD Networks

QKD links generate encryption keys at distant network nodes. The security of QKD links is based on the security of the QKD protocols and their implementation. The QKD systems establish (at least on a protocol level) an ITS communication channel, meaning, as already stated, that the key exchange is secure independently of the resources of the attacker. (Note that sometimes ITS is defined as security against adversaries with unlimited computing resources, where the latter naturally include quantum computing ones. The most general attack by an adversary with arbitrary quantum resources can be viewed as a quantum computer plugged in the communication line, not a remote quantum computer. In this sense, the two definitions are identical although an adversary with unlimited computing resources can erroneously be understood in the more limited sense of an adversary with only unlimited REMOTE - quantum or classical - computing resources.) Each QKD module exposes a key delivery API. This could be a proprietary interface, or a standard one like ETSI GS QKD 004 or ETSI GS QKD 014. These interfaces allow the higher level layer(s) to store securely all the key material generated by each QKD module in a KMS/HSM. In our experiments in the Madrid testbed, we combined both ETSI interfaces in a transparent manner, storing key material per node that was delivered through both interfaces. (As already mentioned, at the Berlin and Poznan testbeds only ETSI GS QKD 014 had been used).

As already also discussed, the QKD networks are typically based on the principle of trusted nodes (secure nodes from which no secret information can be retrieved) [23]. The trusted nodes are physically connected through quantum channels (and classical associated channels). The key material generated on such a QKD link is local to the (two) trusted nodes that each link connects, meaning these keys are only known to the legitimate users at the endpoints of this link. A QKD network is a set of (potentially dynamic) interconnected QKD links, each of them with their local key material. This means that if two end-points want to share the same key when they do not share a direct quantum link, the network needs to transport key material utilizing the key generated over quantum links. As noted, this process is typically denoted as a key forwarding, key transport, or key relay, and is a secure communication of the final E2E key using the key material generated per link to protect the final key transport from node to node until the destination node is reached. Once the destination has obtained the key material, it is possible to establish secure E2E communication. This requires precise synchronization of the distributed KMS among all the intermediate nodes involved in a key transport path for any E2E key delivery on the network.

Before we continue, we point out that the main (although by no means the only) application of symmetric key material is encryption, i.e., the utilization of the key for encryption purposes. While the one-time pad encryption (combined with almost strongly universal$_2$ hashing—see [26] and references therein—for ensuring message integrity) is ITS, this method is too "key hungry" to be readily applied. For this reason, traditional block cipher algorithms, such as AES or ChaCha, with 256-bit keys are widely used. While not being ITS, these are currently believed to be (at least) quantum-safe as the best quantum attack known against symmetric cryptography is based on the Grover algorithm for quantum computers that allows executing a brute force attack with "just" a quadratic speed-up.

3.1.1. Long-Distance (Emulated) QKD Links

The long-distance links as of today in Europe can only be realized using a QKD emulation technology (here we use PQC methods), since, as already mentioned, neither long-distance QKD, nor trusted repeating chains and/or (the constellation of) QKD trusted satellites are yet deployed on the continent. For this reason, all long-distance links have been realized using PQC KEM protocols and SIG authentication. We have, however, as discussed in the Introduction chosen a two-factor approach, in which different implementations of PQC KEMs have been used and different physical paths have been employed. Specifically, different networks, including the terrestrial Internet and commercial satellite systems were used for the demonstrations. The proposed approaches are discussed below.

We have extended the QKD key forwarding concept to a border node problem to be able to transport key material on network segments and "long hauls", for example, in the same metro network, but also between different metropolitan networks, even on an intercontinental scale. In this work, we propose four border-node methods—link-based border node, long-haul link-based border node, long-haul application-based border node, and long-haul application-based border nodes with multi-path diversity—that we have deployed on selected points of presence (PoPs) of the various metro networks. (Note that two metro networks, the Telefonica and REDIMadrid ones, can be seen as a part of the Madrid network, which logically includes also the Munich single link. For the purposes of the present paper, these three Madrid network segments can be seen as three independent networks).

A general remark: The four methods have a purely experimental purpose, but they may be viewed as a demonstrator or rather an emulator for a trans-European long-haul QKD network.

Link-Based Border Node—Method 1

This method, specifically as presented here, is appropriate for connecting QKD networks with similar or even identical architectural designs. It was used in MadQCI to connect the networks of two different telecommunication providers, the REDIMadrid QKD network and the Telefónica QKD network. Specifically, the link between border nodes is the one between the nodes Quevedo (REDIMadrid) and Norte (Telefónica) and is shown in Figure 4 as a yellow line. It is used simultaneously by an Id Quantique system running in the 1310 nm band and a number of HWDU links running in the 1550 nm band.

A hybridized key, produced by XOR-ing the keys generated over a pair of those direct QKD links (an Id Quantique link and one HWDU link), is shared between the border nodes of the two QKD networks (MadQCI segments) and used as a bridge to transfer keys from any node in one network to any other node in the other network using the hop-by-hop approach. (Note that this key is further XOR-ed with a PQC key between any two communicating nodes, something that appears redundant from an abstract cryptographic protocol perspective but is security-relevant in view of the lower security level of the implementations, as discussed in Section 1).

From an operational perspective the link-based border node method appears to be analogous to the operation of a regular metro network QKD link. However, taking into account that the link connects two different QKD networks, we extended the functionality of the quantum forwarding plane (the set of functionalities and devices required to forward the QKD keys through the network, see Table 5) to be able to transport the final keys from one administrative domain to another. This extension also includes the control mechanisms (control planes in the QKD SDN case) of the networks. As in this case both administrative domains are QKD-SDN based, they have their separate respective SDN controllers that, for security reasons, are not allowed to communicate directly one with the other, in contrast to the case of a single QKD network. The extensions stem from the need to connect the control plane and the quantum forwarding plane in one network to their counterparts in another network, and, obviously, both extensions follow the same paradigm. When an application requires secure communication with another one that is located in a different network, the source and the destination of the application are in distinct (mutually foreign) network domains. For this reason, the key forwarding configuration of the nodes needs to be carried out in one network, from the source to the border node. The latter is configured as a relay node to its counterpart border node in the second network. From there, key forwarding is configured to the destination node. The border nodes are the only ones allowed to communicate with external networks, and typically this operation requires a strict service level agreement and a corresponding negotiation protocol. The SDN controllers of each network also need to configure all the necessary intermediate relay nodes in their respective domains, together with any resource allocation needed. Once the key has been transported from the source node in one network to the destination node in the other, an E2E secure communication can take place.

Long-Haul Link-Based Border Node—Method 2

This method is logically similar to the previous one, and it is adapted to the case in which no distant QKD keys are available. It has been used to connect two QKD networks: one is a MadQCI segment, the Telefónica Network, and the other is the network segment represented by the remote QKD link at the HWDU facilities in Munich. The long distance between both network segments cannot presently be bridged by true QKD links, as already discussed. As an alternative, and since both segments are based on the same SDN paradigm and design, a long-distance QKD link is emulated using PQC between the corresponding border nodes at Telefonica Research in Distrito and the Munich Research Center of HWDU, The emulated long-distance QKD link runs two distinct PQC KEM protocols, the outputs of which are hybridized, as in Method 1, to create a border node to border node key (i.e., through the emulated distant QKD link).

The rest of the process is strictly analogous to that described in Method 1. We point out, however, that a modification of the SDN control mechanism (the control plane) is needed. Specifically, ETSI GS QKD 015 [25] needs to be overridden and extended by a "link type" property that can be QKD, PQC, RAW (i.e., traditional communication), or other. Essentially, the SDN node control instance (the SDN agent [25]) manages this as another QKD link, but it knows that this is a PQC link. The ETSI GS QKD 015 has been extended to give support for this specific feature, adding the property "link type", so that it can be QKD, RAW, PQC or other. This is the most general description of Method 2, which is similar to Method 1.

In the present realization, however, for simplicity and to demonstrate that different domains could be easily merged through the SDN approach, we decided to manage the Telefónica and Munich segments from a unified SDN point of view. This means that there is only one SDN controller managing all the QKD trusted nodes in the Telefónica segment and the Munich link as if this were the same physical network. Such an approach simplifies the final E2E key delivery and demonstrates the potential operation in case of the availability of meta- (or cross-border) controllers.

We further decided to implement this link using the ETSI GS QKD 004 key delivery interface because of the quality of service options offered by this API. It allows us to emulate this link choosing a constant key rate. We selected 256bps and a key length of 32 bytes. We point out that even this conservative speed choice allows us to simulate a set of trusted nodes with key provisioning. The PQC algorithm used for this type of link is presented in detail in the discussion on the subsequent model. For implementation, again, the Open Quantum Safe [18] library was selected.

Long-Haul Application-Based Border Node—Method 3

This approach is oriented to the interconnection of networks that are different in terms of their design, for example, networks based on the traditional QKD network layered architecture (Berlin, Poznan), with a network based on a different architectural paradigm, such as, e.g., a QKD-software-defined network (Madrid). (This approach has also been used to connect Berlin and Poznan, although Method 2. would also have been possible). The idea is to have an application service running on the authorized nodes of each network. This application service is administrated in each domain by the respective network operator. Due to the long distance between the testbeds, both sides of the application establish an emulated QKD link (a PQC link instead of a QKD one), albeit in a two-factor manner, which is based on the hybridization of two implementations but over the same path. In detail, the algorithm consists of the following steps:

1. Retrieve random numbers (RNDs) and matching identifier RNDIDs from a (quantum) random number generator ((Q)RNG).

2. Encrypt the random numbers using different PQC KEM algorithms independently and hybridize the outputs to a single key string.
3. Add meta-information. For example, the RNDID, a key validity period, or the name of the sending node is added to the encrypted random number. The entire package is finally signed using a different PQC SIG algorithm for each KEM choice.
4. After data serialization, the data package is sent from one border node to the other border node.
5. Finally, the sending side pushes the encryption key, the identifier, and the corresponding metadata into a KMS that may be an HSM or an encrypted file share.

At the receiving node, the key exchange protocol consists of the following steps:

1. After the reception of a message, the sender is identified by its IP address.
2. The message is deserialized and the signature of the sender is validated using the PQC SIG algorithms. The appropriate public SIG key(s) of the sender is (are) determined from the IP address recorded at step 1.
3. If validated, the receiver de-hybridizes and subsequently decrypts the encrypted random number, the identifier, and the corresponding metadata.
4. Finally, the receiving side pushes the transported key, the identifier, and the corresponding metadata into a KMS.

Since the algorithm is applied on the application level, no low-level details of the network architecture are required for this service. The border node service is running constantly, and it stabilizes a constant PQC link where randomized keys could be obtained by several methods (one of them being the use of a QRNG). The keys are protected and shared between both sides of the border-node to border-node link and used as transport keys to protect the final E2E key material. Note that if (one of) the SIG or KEM implementations is broken this might result in a denial-of-service attack as the final keys between the distant nodes might not coincide or the messages may be considered unauthorized.

A quantum-safe long-haul application-based border node was implemented as a regular QKD key consumer application on top of the KMS layer using ETSI GS QKD 004 (ETSI GS QKD 014 could also be appropriate). This approach was designed for the key exchange between different (architectural and functional) networks. Applying the key interchange on the application layer hides the low-level details of the network and delegates the key transport to the operator's network level. Indeed, the mentioned connectivity application only needs to receive the secure key material to transport independently how this key material has been created or transported in each network. Note that with this approach, the unique requirement for the QKD network is the availability of a key delivery interface on each QKD border node to obtain the key before a quantum-safe KEM is used to encrypt the messages sent to the application in the adjacent network. Note that no conflict occurs if the interfaces in both networks are not the same.

For example, assume that one network is a realization of a QKD SDN architecture and the second follows a layered architecture QKD network design. In the SDN QKD network, the request to transport a key between the QKD nodes on both sides of a link/chain (source and destination) is directed to the network-internal SDN controller. The SDN controller organizes all the intermediate QKD trusted nodes in relay mode to transport the key up to the border node, providing all the necessary resources. As a result, the secure key is ready on the application's side of the first network border node. Then, it is transported to the border node of the second network, decrypted, and injected into the KMS of the layered network, which takes over and negotiates for the E2E keys to transport the keys to the final destination. (This is in contrast to controller-assisted transport in a QKD SDN network.) At this point, E2E secure communication from one network to the other can be carried out. Note that this application could also be implemented as a separate application that receives keys (for example, in one port) irrespective of their origin.

In future, when direct QKD links between the border nodes of different QKD networks will be available, (one of) the consumer-application(s) supporting this method, will simply utilize the direct QKD link key as the key source and encrypt the payload-key by a symmetric ITS method, i.e., a one-time pad.

Long-Haul Application-Based Border Nodes with Multi-Path Diversity—Method 4

This approach adds multi-path security to the previous one using two disjoint network links. We used the public Internet or the "ground link" and a satellite-based link via the commercial Iridium network, or the "space link". We chose the Iridium network because it is a commercial network with worldwide coverage and affordable access. The setup is shown in Figure 5. Note that in the Introduction, we previously mentioned that the two-path diversity is preferable from the point of view of security. Again, the long-distance forces us to use emulations of QKD. A two-path algorithm, based on different PQC KEM and SIG protocols is detailed below. It differs from the one outlined in the previous approach in the sense that non-coinciding random number strings are sent in this two-factor version along different routes and subsequently these are combined, rather than encrypting a single random string with different methods and then hybridizing the result. Similarly, if (one of) the SIG or KEM implementations is broken, this might result in a denial-of-service attack as the final keys between the distant nodes might not coincide or the messages may be considered unauthorized.

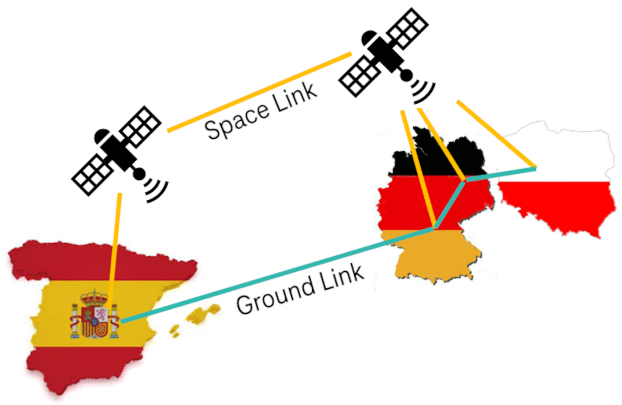


Figure 5. The disjoint network is realized by a "space link" via the Iridium network and a "ground link" via the public Internet. The network connects the gateway nodes of the Madrid, Berlin, and Poznan QKD testbeds. The Munich Research Center of Huawei in Germany serves as a pseudo-internal node of the Madrid network.

The PQC-based two-path key exchange protocol involves the following steps for the sending node:

1. Retrieve two random numbers (denoted by RND_1 and RND_2) and a matching identifier per RND (denoted by a single RNDID) using a (Q)RNG. The index 1 is liaised to the space link, whereas the index 2 denotes the "ground link".
2. Encrypt the random numbers using different PQC KEM (key encapsulation mechanism) algorithms. Depending on the chosen path, a different public KEM key is applied. Any appropriate KEM algorithm may be used. We used Kyber on the space link and NTRU on the ground link, because these algorithms showed good performance with current implementations [27].
3. Meta-information, for example the RNDID, a key validity period, or the name of the sending node is added to the encrypted random number. The entire package is finally signed using a different PQC SIG algorithm for each network path. Any appropriate SIG algorithm may be used. We used Falcon on the space link and Dilithium on the ground link.
4. After data serialization, one key package is sent via the space link, the other key package via the ground link.

5. On successful data transmission, the sending side combines the two random numbers RND_1 and RND_2 to compute an encryption key (KEY) using a key derivation function (KDF), so that KEY = KDF(RND_1, RND_2, PSK), where PSK is some (possibly empty) pre-shared key string. Any KDF standardized by NIST or ETSI may be chosen [28,29]. (In case real QKD links and not only emulated QKD links are used, this choice of KDF must be restricted to functions that ensure the epsilon-composability of the output; e.g., a simple XOR-ing function can be considered.) The KEYID is set identical to the RNDID and will be required for the key negotiation protocols. Finally, the sending side pushes the encryption key, the identifier, and the corresponding metadata into a KMS.

At the receiving node, the key exchange protocol consists of the following steps:

1. After the reception of a message on either network $path_i$, $i = 1, 2$, the sender is identified by its IP address.
2. The message is deserialized and the signature of the sender is validated using the PQC SIG algorithm. The appropriate public SIG key(s) of the sender is (are) determined from the IP address recorded at step 1.
3. If validated, the receiver decrypts the encrypted random number RND_i, the identifier $RNDID_i$, and the corresponding metadata using the PQC algorithm KEM_i.
4. The decrypted random numbers, their identifiers, and the metadata are then sent to a queue for further processing.

If two random numbers RND_1 and RND_2 with the same identifier RNDID are found in the queue, the numbers RND_i will be combined so that KEY = KDF (RND_1, RND_2, PSK) is computed (see above the comments on the KDF choice). Finally, the receiving side pushes the final key, the identifier and the corresponding metadata into a KMS. This solution makes use of Open Quantum Safe (OQS) [18] for the implementation of the PQC KEM and SIG algorithms. The PQC algorithms are compiled into openssl. A PQC-enabled version of the nginx web server with multiple workers is used and the Python code uses multithreading to increase the performance.

The security system preserves the secrecy of the final key, as long as a single path of the disjoint network paths remains secure. This means that even if a PQC algorithm used on one of the paths is successfully attacked in the future, the other PQC algorithm acting on the disjoint path (if not also broken) would guarantee the overall security of the system (up to a denial-of-service attack, as already discussed). The KDF combines the two random numbers in such a way that even the knowledge of one of the two RNDs would not allow the adversary to compute the resulting final key [30], a property known as robust combination. As discussed, the security of the solution can even be increased by adding more disjoint paths and securing the key exchange using different PQC algorithms. As well as the combination of satellite and terrestrial networks, there are other commercial networks which are disjoint, for example, the networks of competing mobile network service providers or European research fiber networks or commercial fiber networks. Once existing, even a satellite QKD or long-haul quantum optical key exchange link may be added [31,32] to make the solution information theoretically secure (albeit by a more careful selection of the KDF, as mentioned) with a significant side channel reduction.

The long-haul application-based border nodes were implemented in the Berlin, Poznan, and Madrid testbeds on specific QKD trusted nodes, using physical servers. The Iridium "space network" and the "Internet" are used to exchange the encryption keys over disjoint network paths. Using a virtual machine with two CPUs and 16 GB RAM, the presented software solution manages to transfer 4 kBits (which corresponds to 16 AES-256 keys per second). The virtual machine was equipped with a PQC-enhanced web server and client applications running the key exchange. To do so, and to compensate for the long latency of about 600 ms per request, the best performance was achieved when sending blocks of 50 to 75 keys per https session. The bottleneck of the implementation turned out to be the recombination function to grab the two random numbers and apply the KDF. We believe that the software performance can be tremendously improved by choosing a more powerful coding language, like RUST or C, and by changing the software architecture to asynchronous queuing with multiple stateless microservices acting. The solution itself is scalable with more microservices exchanging keys between the endpoint sharing the capacity of a larger number of satellite access antennas. The usage of an LEO (low Earth orbit) satellite constellation, like Star Link [33], may also increase the performance of the implementation. Geographically, the solution may be scaled globally by integrating more endpoints and using existing or coming satellite constellations in conjunction with "standard" Internet connections.

3.1.2. Experimental Results

The following section summarizes the key performance metrics of the key distribution channels between the border nodes of the three testbeds. Each border node (as, in fact, any trusted node) was equipped with a computer system and software to run the PQC-enabled key exchange protocols. The networks were linked by a logical, classical VPN network.

Table 6 shows the four key distribution methods, their deployment location, and the name of the key encapsulation and signature algorithms applied. The key exchange rate is shown. It is important to note that the (classical) PQC key exchange protocol makes use of TCP/IP, which corrects transmission errors by design. It is also important to mention that the key exchange metrics originate from a single working application or thread. By adding more CPU threads to the key exchange application, the performance can easily be scaled. Therefore, using a CPU with a higher number of threads yields higher key exchange rates.

Table 6. Summary of the four key exchange methods to integrate the QKD testbeds.

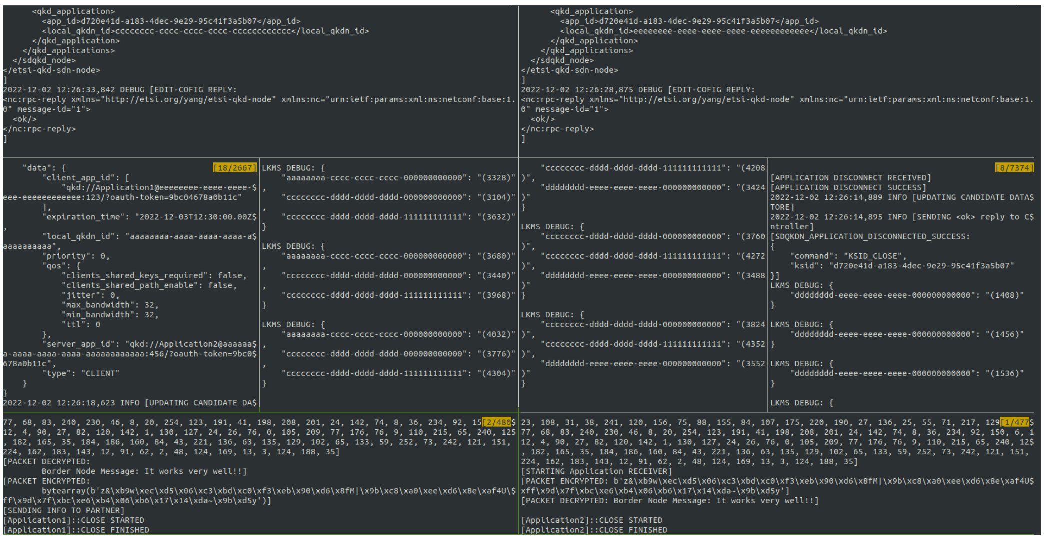
Method	From-To	QKD/PQC Algorithm	Key Rate
1	REDIMadrid to Telefonica	QKD & Kyber/Falcon and NTRU/Dilithium	QoS based on 256 Bit/s
2	Madrid to Munich	Kyber/Falcon and NTRU/Dilithium	QoS based on 256 Bit/s
3	Madrid to Berlin, Madrid to Poznan	Kyber/Falcon and NTRU/Dilithium	QoS based on 256 Bit/s
4	Berlin to Madrid, Berlin to Poznan	Kyber/Falcon and NTRU/Dilithium	4 kBit/s

Demonstration of Method 1

The following figure shows a successful E2E key exchange between nodes in the REDIMadrid segment and the Telefónica segment of the Madrid testbed. The left side of Figure 6 shows the main entities involved in this communication on the REDIMadrid side. The right side shows the Telefónica side. On top, the respective SDN controllers are to be seen. These govern each respective domain, whereby each of them does not have details of the other network. In our experiment, there are two nodes in each network that are involved in the communication, an internal regular trusted node and the border node: Sansa and Rickard, respectively, on the REDIMadrid side, and Lyanna and Eddard on the Telefónica side. Their respective LKMSes and associated QKD links are shown in the middle of the figure. An application asks for a key in each respective network and each SDN controller orchestrates all the nodes, including the border node, to ensure a successful E2E key delivery, as shown in the lower part of the figure.

Demonstration of Method 2

In Figure 7, the links and the amount of key material stored per link between the Meera and Jojen nodes in Munich (link: dddddddd-0000-0000-0000-eeeeeeeeeeee) and between the Catelyn node in Concepción and Eddard node in Distrito (link: aaaaaaaa-0000-0000-0000-cccccccccccc) are shown. The long-haul link-based border node connection operates on the link cccccccc-dddd-dddd-dddd-cccccccccccc. In the figure, this link is indicated in red. It connects Eddard in Distrito with Meera in Munich with a QKD-simulated link using PQC.



```
    <qkd_application>
      <app_id>d720e41d-a183-4dec-9e29-95c41f3a5b07</app_id>
      <local_qkdn_id>cccccccc-cccc-cccc-cccc-cccccccccccc</local_qkdn_id>
    </qkd_application>
  </qkd_applications>
 </sdqkd_node>
</etsi-qkd-sdn-node>
]
2022-12-02 12:26:33,842 DEBUG [EDIT-COFIG REPLY:
<nc:rpc-reply xmlns="http://etsi.org/yang/etsi-qkd-node" xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.
0" message-id="1">
 <ok/>
</nc:rpc-reply>
]
```

```
    <qkd_application>
      <app_id>d720e41d-a183-4dec-9e29-95c41f3a5b07</app_id>
      <local_qkdn_id>eeeeeeee-eeee-eeee-eeee-eeeeeeeeeeee</local_qkdn_id>
    </qkd_application>
  </qkd_applications>
 </sdqkd_node>
</etsi-qkd-sdn-node>
]
2022-12-02 12:26:28,875 DEBUG [EDIT-COFIG REPLY:
<nc:rpc-reply xmlns="http://etsi.org/yang/etsi-qkd-node" xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.
0" message-id="1">
 <ok/>
</nc:rpc-reply>
]
```

```
    "data": {                                  [18/2667]
        "client_app_id": [
            "qkd://Application1@eeeeeeee-eeee-eeee-$
eee-eeeeeeeeeeee:123/?oauth-token=9bc04678a0b11c"
        ],
        "expiration_time": "2022-12-03T12:30:00.00Z$
,
        "local_qkdn_id": "aaaaaaaa-aaaa-aaaa-aaaa-a$
aaaaaaaaaa",
        "priority": 0,
        "qos": {
            "clients_shared_keys_required": false,
            "clients_shared_path_enable": false,
            "jitter": 0,
            "max_bandwidth": 32,
            "min_bandwidth": 32,
            "ttl": 0
        },
        "server_app_id": "qkd://Application2@aaaaaa$
a-aaaa-aaaa-aaaa-aaaaaaaaaaaa:456/?oauth-token=9bc0$
678a0b11c",
        "type": "CLIENT"
    }
}
2022-12-02 12:26:18,623 INFO [UPDATING CANDIDATE DA$
```

```
LKMS DEBUG: {
    "aaaaaaaa-cccc-cccc-cccc-000000000000": "(3328)"
,
    "cccccccc-dddd-dddd-dddd-000000000000": "(3104)"
,
    "cccccccc-dddd-dddd-dddd-111111111111": "(3632)"
}

LKMS DEBUG: {
    "aaaaaaaa-cccc-cccc-cccc-000000000000": "(3680)"
,
    "cccccccc-dddd-dddd-dddd-000000000000": "(3440)"
,
    "cccccccc-dddd-dddd-dddd-111111111111": "(3968)"
}

LKMS DEBUG: {
    "aaaaaaaa-cccc-cccc-cccc-000000000000": "(4032)"
,
    "cccccccc-dddd-dddd-dddd-000000000000": "(3776)"
,
    "cccccccc-dddd-dddd-dddd-111111111111": "(4304)"
}
```

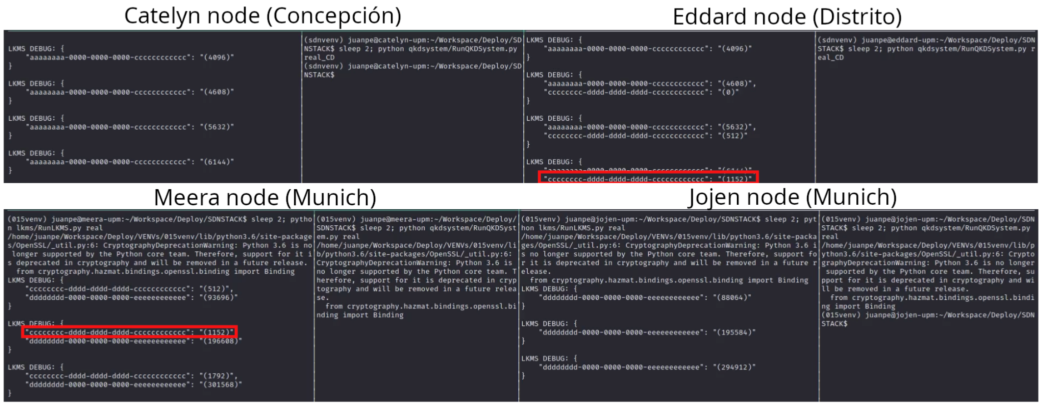
```
    "cccccccc-dddd-dddd-dddd-111111111111": "(4208
)",
    "dddddddd-eeee-eeee-eeee-000000000000": "(3424
)"
}

LKMS DEBUG: {
    "cccccccc-dddd-dddd-dddd-000000000000": "(3760
)",
    "cccccccc-dddd-dddd-dddd-111111111111": "(4272
)",
    "dddddddd-eeee-eeee-eeee-000000000000": "(3488
)"
}

LKMS DEBUG: {
    "cccccccc-dddd-dddd-dddd-000000000000": "(3824
)",
    "cccccccc-dddd-dddd-dddd-111111111111": "(4352
)",
    "dddddddd-eeee-eeee-eeee-000000000000": "(3552
)"
}
```

```
                                              [8/7374]
[APPLICATION DISCONNECT RECEIVED]
[APPLICATION DISCONNECT SUCCESS]
2022-12-02 12:26:14,889 INFO [UPDATING CANDIDATE DATA$
TORE]
2022-12-02 12:26:14,895 INFO [SENDING <ok> reply to C$
ntroller]
[SDQKDN_APPLICATION_DISCONNECTED_SUCCESS:
{
    "command": "KSID_CLOSE",
    "ksid": "d720e41d-a183-4dec-9e29-95c41f3a5b07"
}]
LKMS DEBUG: {
    "dddddddd-eeee-eeee-eeee-000000000000": "(1408)"
}

LKMS DEBUG: {
    "dddddddd-eeee-eeee-eeee-000000000000": "(1456)"
}

LKMS DEBUG: {
    "dddddddd-eeee-eeee-eeee-000000000000": "(1536)"
}

LKMS DEBUG: {
```

```
77, 68, 83, 240, 230, 46, 8, 20, 254, 123, 191, 41, 198, 208, 201, 24, 142, 74, 8, 36, 234, 92, 15[2/480$
12, 4, 90, 27, 82, 120, 142, 1, 130, 127, 24, 26, 76, 0, 105, 209, 77, 176, 76, 9, 110, 215, 65, 240, 125
, 182, 165, 35, 184, 186, 160, 84, 43, 221, 136, 63, 135, 129, 102, 65, 133, 59, 252, 73, 242, 121, 151,
224, 162, 183, 143, 12, 91, 62, 2, 48, 124, 169, 13, 3, 124, 188, 35]
[PACKET DECRYPTED:
        Border Node Message: It works very well!!]
[PACKET ENCRYPTED:
        bytearray(b'z&\xb9w\xec\xd5\x06\xc3\xbd\xc0\xf3\xeb\x90\xd6\x8fM|\x9b\xc8\xa0\xee\xd6\x8e\xaf4U\$
ff\x9d\x7f\xbc\xe6\xb4\x06\xb6\x17\x14\xda~\x9b\xd5y')]
[SENDING INFO TO PARTNER]
[Application1]::CLOSE STARTED
[Application1]::CLOSE FINISHED
```

```
23, 108, 31, 38, 241, 120, 156, 75, 88, 155, 84, 107, 175, 220, 190, 27, 136, 25, 55, 71, 217, 129[1/477$
77, 68, 83, 240, 230, 46, 8, 20, 254, 123, 191, 41, 198, 208, 201, 24, 142, 74, 8, 36, 234, 92, 150, 6, 1
12, 4, 90, 27, 82, 120, 142, 1, 130, 127, 24, 26, 76, 0, 105, 209, 77, 176, 76, 9, 110, 215, 65, 240, 12$
, 182, 165, 35, 184, 186, 160, 84, 43, 221, 136, 63, 135, 129, 102, 65, 133, 59, 252, 73, 242, 121, 151,
224, 162, 183, 143, 12, 91, 62, 2, 48, 124, 169, 13, 3, 124, 188, 35]
[STARTING Application RECEIVER]
[PACKET ENCRYPTED: b'z&\xb9w\xec\xd5\x06\xc3\xbd\xc0\xf3\xeb\x90\xd6\x8fM|\x9b\xc8\xa0\xee\xd6\x8e\xaf4U$
xff\x9d\x7f\xbc\xe6\xb4\x06\xb6\x17\x14\xda~\x9b\xd5y']
[PACKET DECRYPTED: Border Node Message: It works very well!!]

[Application2]::CLOSE STARTED
[Application2]::CLOSE FINISHED
```

Figure 6. Application key transport between the REDIMadrid domain and the Telefónica domain through a QKD link-based border node. The left part represents the REDIMadrid domain; the right part represents the Telefónica domain. The upper part represents the SDN controllers of each network, both controllers being NETCONF based. The central part represents the LKMS of each node involved in the communication: on the left, the source node of REDIMadrid receiving a QKD key request through ETSI GS QKD 004; in the center-left, there is the border node in the REDIMadrid side; on the center-right, the border node of the Telefónica side; and on the right, the destination node of the communication, showing the application disconnect. In the lower part, the left side represents the source application and the right side the destination application.



```
LKMS DEBUG: {
    "aaaaaaaa-0000-0000-0000-cccccccccccc": "(4096)"
}

LKMS DEBUG: {
    "aaaaaaaa-0000-0000-0000-cccccccccccc": "(4608)"
}

LKMS DEBUG: {
    "aaaaaaaa-0000-0000-0000-cccccccccccc": "(5632)"
}

LKMS DEBUG: {
    "aaaaaaaa-0000-0000-0000-cccccccccccc": "(6144)"
}
```

```
(sdnvenv) juanpe@catelyn-upm:~/Workspace/Deploy/SD
NSTACK$ sleep 2; python qkdsystem/RunQKDSystem.py
real_CD
(sdnvenv) juanpe@catelyn-upm:~/Workspace/Deploy/SD
NSTACK$
```

```
LKMS DEBUG: {
    "aaaaaaaa-0000-0000-0000-cccccccccccc": "(4096)"
}

LKMS DEBUG: {
    "aaaaaaaa-0000-0000-0000-cccccccccccc": "(4608)",
    "cccccccc-dddd-dddd-dddd-cccccccccccc": "(0)"
}

LKMS DEBUG: {
    "aaaaaaaa-0000-0000-0000-cccccccccccc": "(5632)",
    "cccccccc-dddd-dddd-dddd-cccccccccccc": "(512)"
}

LKMS DEBUG: {
    "aaaaaaaa-0000-0000-0000-cccccccccccc": "(6144)"
    "cccccccc-dddd-dddd-dddd-cccccccccccc": "(1152)"
```

```
(sdnvenv) juanpe@eddard-upm:~/Workspace/Deploy/SDN
STACK$ sleep 2; python qkdsystem/RunQKDSystem.py r
eal_CD
```

```
(015venv) juanpe@meera-upm:~/Workspace/Deploy/SDNSTACK$ sleep 2; pytho
n lkms/RunLKMS.py real
/home/juanpe/Workspace/Deploy/VENVs/015venv/lib/python3.6/site-package
s/OpenSSL/_util.py:6: CryptographyDeprecationWarning: Python 3.6 is no
 longer supported by the Python core team. Therefore, support for it i
s deprecated in cryptography and will be removed in a future release.
  from cryptography.hazmat.bindings.openssl.binding import Binding
LKMS DEBUG: {
    "cccccccc-dddd-dddd-dddd-cccccccccccc": "(512)",
    "dddddddd-0000-0000-0000-eeeeeeeeeeee": "(93696)"
}

LKMS DEBUG: {
    "cccccccc-dddd-dddd-dddd-cccccccccccc": "(1152)"
    "dddddddd-0000-0000-0000-eeeeeeeeeeee": "(196608)"
}

LKMS DEBUG: {
    "cccccccc-dddd-dddd-dddd-cccccccccccc": "(1792)",
    "dddddddd-0000-0000-0000-eeeeeeeeeeee": "(301568)"
}
```

```
(015venv) juanpe@meera-upm:~/Workspace/Deploy/
SDNSTACK$ sleep 2; python qkdsystem/RunQKDSyst
em.py real
/home/juanpe/Workspace/Deploy/VENVs/015venv/li
b/python3.6/site-packages/OpenSSL/_util.py:6:
CryptographyDeprecationWarning: Python 3.6 is
no longer supported by the Python core team. T
herefore, support for it is deprecated in cryp
tography and will be removed in a future relea
se.
  from cryptography.hazmat.bindings.openssl.bi
nding import Binding
```

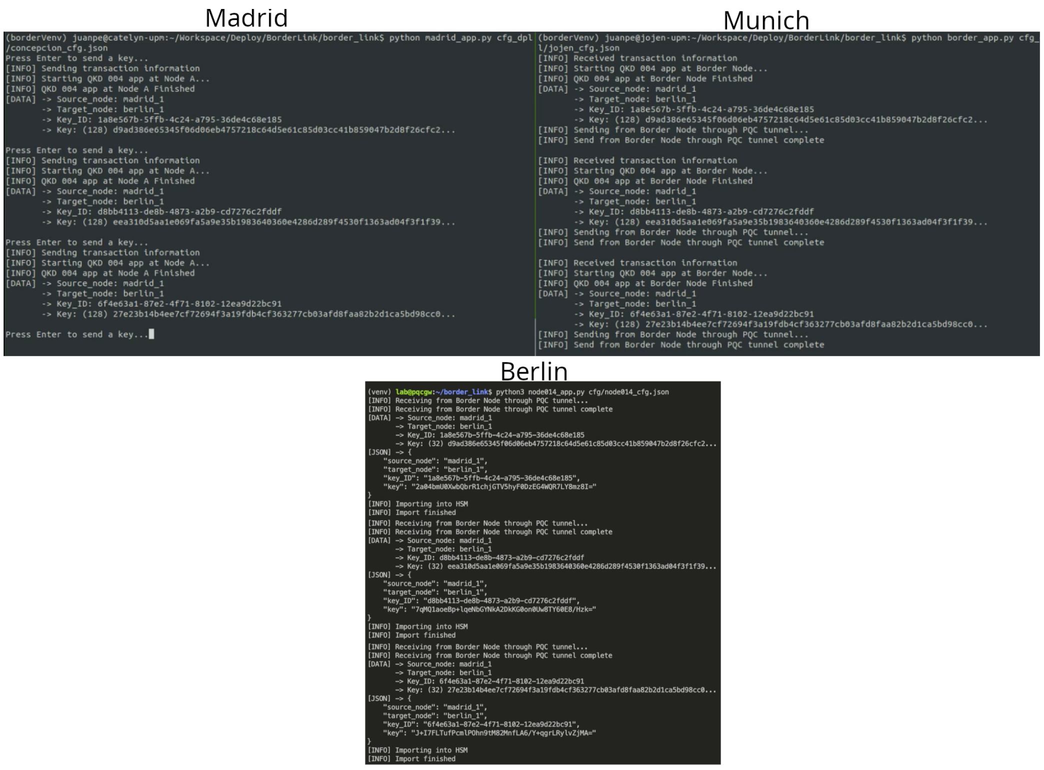
```
(015venv) juanpe@jojen-upm:~/Workspace/Deploy/SDNSTACK$ sleep 2; pyt
hon lkms/RunLKMS.py real
/home/juanpe/Workspace/Deploy/VENVs/015venv/lib/python3.6/site-packa
ges/OpenSSL/_util.py:6: CryptographyDeprecationWarning: Python 3.6 i
s no longer supported by the Python core team. Therefore, support fo
r it is deprecated in cryptography and will be removed in a future r
elease.
  from cryptography.hazmat.bindings.openssl.binding import Binding
LKMS DEBUG: {
    "dddddddd-0000-0000-0000-eeeeeeeeeeee": "(88064)"
}

LKMS DEBUG: {
    "dddddddd-0000-0000-0000-eeeeeeeeeeee": "(195584)"
}

LKMS DEBUG: {
    "dddddddd-0000-0000-0000-eeeeeeeeeeee": "(294912)"
}
```

```
(015venv) juanpe@jojen-upm:~/Workspace/Deploy/SDN
STACK$ sleep 2; python qkdsystem/RunQKDSystem.py
real
/home/juanpe/Workspace/Deploy/VENVs/015venv/lib/p
ython3.6/site-packages/OpenSSL/_util.py:6: Crypto
graphyDeprecationWarning: Python 3.6 is no longer
 supported by the Python core team. Therefore, su
pport for it is deprecated in cryptography and wi
ll be removed in a future release.
  from cryptography.hazmat.bindings.openssl.bindi
ng import Binding
(015venv) juanpe@jojen-upm:~/Workspace/Deploy/SDN
STACK$
```

Figure 7. Links between Concepción, Distrito, and Munich. The QKD-simulated link using PQC between Distrito and Munich is indicated in red.

Demonstration of Method 3

Figure 8 shows screen views of the full key transport from the Cathelyn node in Concepción to the Jojen node in Munich and the final transmission from there to Berlin. On the left side of the figure, a QKD application is started to communicate from Madrid to Berlin. The key transport includes a first QKD link from Concepción to Distrito, then a long-haul link-based border node (Method 2) from Distrito to the Meera node in Munich. The third link is again a QKD link in Munich from Meera to Jojen, where the application-based border node method is realized; the received key material is encrypted by two different key encapsulation algorithms and sent to Berlin. On the Berlin side, the reception side of the application is waiting to receive the key material and to distribute it inside the network.



```
(borderVenv) juanpe@catelyn-upm:~/Workspace/Deploy/BorderLink/border_link$ python madrid_app.py cfg_dpl
/concepcion_cfg.json
Press Enter to send a key...
[INFO] Sending transaction information
[INFO] Starting QKD 004 app at Node A...
[INFO] QKD 004 app at Node A Finished
[DATA] -> Source_node: madrid_1
       -> Target_node: berlin_1
       -> Key_ID: 1a8e567b-5ffb-4c24-a795-36de4c68e185
       -> Key: (128) d9ad386e65345f06d06eb4757218c64d5e61c85d03cc41b859047b2d8f26cfc2...

Press Enter to send a key...
[INFO] Sending transaction information
[INFO] Starting QKD 004 app at Node A...
[INFO] QKD 004 app at Node A Finished
[DATA] -> Source_node: madrid_1
       -> Target_node: berlin_1
       -> Key_ID: d8bb4113-de8b-4873-a2b9-cd7276c2fddf
       -> Key: (128) eea310d5aa1e069fa5a9e35b1983640360e4286d289f4530f1363ad04f3f1f39...

Press Enter to send a key...
[INFO] Sending transaction information
[INFO] Starting QKD 004 app at Node A...
[INFO] QKD 004 app at Node A Finished
[DATA] -> Source_node: madrid_1
       -> Target_node: berlin_1
       -> Key_ID: 6f4e63a1-87e2-4f71-8102-12ea9d22bc91
       -> Key: (128) 27e23b14b4ee7cf72694f3a19fdb4cf363277cb03afd8faa82b2d1ca5bd98cc0...

Press Enter to send a key...
```

```
(borderVenv) juanpe@jojen-upm:~/Workspace/Deploy/BorderLink/border_link$ python border_app.py cfg_
l/jojen_cfg.json
[INFO] Received transaction information
[INFO] Starting QKD 004 app at Border Node...
[INFO] QKD 004 app at Border Node Finished
[DATA] -> Source_node: madrid_1
       -> Target_node: berlin_1
       -> Key_ID: 1a8e567b-5ffb-4c24-a795-36de4c68e185
       -> Key: (128) d9ad386e65345f06d06eb4757218c64d5e61c85d03cc41b859047b2d8f26cfc2...
[INFO] Sending from Border Node through PQC tunnel...
[INFO] Send from Border Node through PQC tunnel complete

[INFO] Received transaction information
[INFO] Starting QKD 004 app at Border Node...
[INFO] QKD 004 app at Border Node Finished
[DATA] -> Source_node: madrid_1
       -> Target_node: berlin_1
       -> Key_ID: d8bb4113-de8b-4873-a2b9-cd7276c2fddf
       -> Key: (128) eea310d5aa1e069fa5a9e35b1983640360e4286d289f4530f1363ad04f3f1f39...
[INFO] Sending from Border Node through PQC tunnel...
[INFO] Send from Border Node through PQC tunnel complete

[INFO] Received transaction information
[INFO] Starting QKD 004 app at Border Node...
[INFO] QKD 004 app at Border Node Finished
[DATA] -> Source_node: madrid_1
       -> Target_node: berlin_1
       -> Key_ID: 6f4e63a1-87e2-4f71-8102-12ea9d22bc91
       -> Key: (128) 27e23b14b4ee7cf72694f3a19fdb4cf363277cb03afd8faa82b2d1ca5bd98cc0...
[INFO] Sending from Border Node through PQC tunnel...
[INFO] Send from Border Node through PQC tunnel complete
```

```
(venv) lab@pqcgw:~/border_link$ python3 node014_app.py cfg/node014_cfg.json
[INFO] Receiving from Border Node through PQC tunnel...
[INFO] Receiving from Border Node through PQC tunnel complete
[DATA] -> Source_node: madrid_1
       -> Target_node: berlin_1
       -> Key_ID: 1a8e567b-5ffb-4c24-a795-36de4c68e185
       -> Key: (32) d9ad386e65345f06d06eb4757218c64d5e61c85d03cc41b859047b2d8f26cfc2...
[JSON] -> {
    "source_node": "madrid_1",
    "target_node": "berlin_1",
    "key_ID": "1a8e567b-5ffb-4c24-a795-36de4c68e185",
    "key": "2a04bmU0XwbQbrR1chjGTV5hyF0DzEG4WQR7LY8mz8I="
}
[INFO] Importing into HSM
[INFO] Import finished
[INFO] Receiving from Border Node through PQC tunnel...
[INFO] Receiving from Border Node through PQC tunnel complete
[DATA] -> Source_node: madrid_1
       -> Target_node: berlin_1
       -> Key_ID: d8bb4113-de8b-4873-a2b9-cd7276c2fddf
       -> Key: (32) eea310d5aa1e069fa5a9e35b1983640360e4286d289f4530f1363ad04f3f1f39...
[JSON] -> {
    "source_node": "madrid_1",
    "target_node": "berlin_1",
    "key_ID": "d8bb4113-de8b-4873-a2b9-cd7276c2fddf",
    "key": "7qMQ1aoeBp+lqeNbGYNkA2DkKG0on0Uw8TY60E8/Hzk="
}
[INFO] Importing into HSM
[INFO] Import finished
[INFO] Receiving from Border Node through PQC tunnel...
[INFO] Receiving from Border Node through PQC tunnel complete
[DATA] -> Source_node: madrid_1
       -> Target_node: berlin_1
       -> Key_ID: 6f4e63a1-87e2-4f71-8102-12ea9d22bc91
       -> Key: (32) 27e23b14b4ee7cf72694f3a19fdb4cf363277cb03afd8faa82b2d1ca5bd98cc0...
[JSON] -> {
    "source_node": "madrid_1",
    "target_node": "berlin_1",
    "key_ID": "6f4e63a1-87e2-4f71-8102-12ea9d22bc91",
    "key": "J+I7FLTufPcmlPOhn9tM82MnfLA6/Y+qgrLRylvZjMA="
}
[INFO] Importing into HSM
[INFO] Import finished
```

Figure 8. On the left side of the image an application is started to send key material from Madrid to Berlin. Three different keys are sent from the Cathelyn node in Concepción to the Jojen node in Munich, where the application-based border node is running and sends the key material to the Berlin border node using a PQC link. On the lower part of the image, the key material received in Berlin is shown.

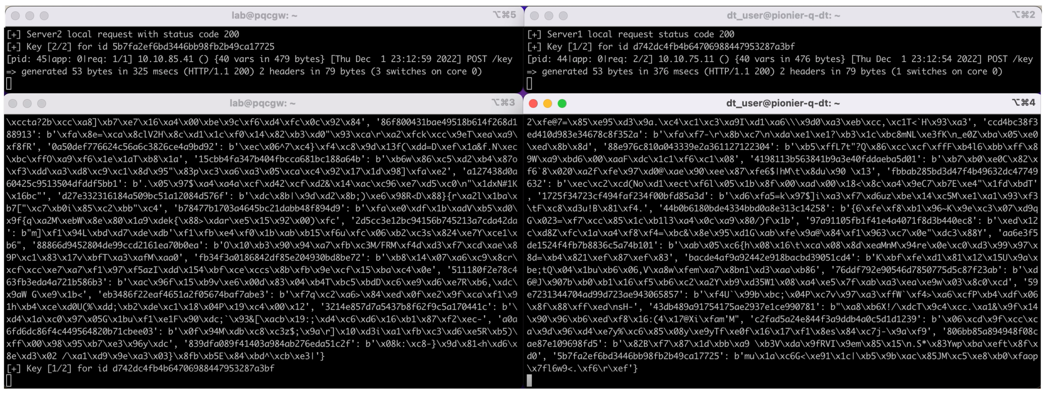
Demonstration of Method 4

Figure 9 shows a screen view of the method 4 key exchange between the border nodes of the Berlin and the Poznan QKD networks. On each border node, a server and a client application is launched. The server-side application listens for exchange requests of random numbers (RNDs). The client-side application transmits blocks of 50 PQC-encrypted and signed random numbers and identifiers via the ground link (called Keys 1/2 in Figure 9). At the same time, the client-side application sends an equally sized, but different, block of PQC-encrypted and signed random numbers and identifiers via the space link (called Keys 2/2 in Figure 9).

The server is required to store the keys exchanged on the ground link for a certain time period to address the longer latency of the satellite channel. This is realized via queues in the memory of the server. A microservice finally queries and combines the related keys using a key derivation function. The client application needs to manage the speed of the key transmission process in accordance with the network capacity. As a result, the same set of random bits is exported to the local key store of the border nodes by the client- and server-side applications.

3.1.3. Key Forwarding through Border Nodes

This paper presents various solutions to realize a border node key exchange between individual European QKD deployments, making use of QKD and PQC as the applied emulations of QKD. This yields an overarching architecture, where the border nodes are deployed to interconnect the QKD deployments. As shown in Figure 10, random numbers are forwarded from a source QKD node to the border node (communication secured by QKD keys), from there to another border node of a target QKD infrastructure (secured for now by an emulated QKD, or a QKD satellite in the future), and further on to a target QKD node. As a result, the source and the target QKD nodes share the same random number, which they may utilize as, e.g., a secure key to, for example, encrypt the classical data payload. As described in the introductory section of this paper, the distant link (border node integration) was realized using PQC links due to the lack of appropriate quantum technology. Clearly, the emulated QKD links stand for true QKD ones in the context of a final architecture blueprint.



```
[+] Server2 local request with status code 200
[+] Key [2/2] for id 5b7fa2ef6bd3446bb98fb2b49ca17725
[pid: 45|app: 0|req: 1/1] 10.10.85.41 () {40 vars in 479 bytes} [Thu Dec  1 23:12:59 2022] POST /key
=> generated 53 bytes in 325 msecs (HTTP/1.1 200) 2 headers in 79 bytes (3 switches on core 0)
▯
```

```
[+] Server1 local request status code 200
[+] Key [1/2] for id d742dc4fb4b64706988447953287a3bf
[pid: 44|app: 0|req: 2/2] 10.10.75.11 () {40 vars in 476 bytes} [Thu Dec  1 23:12:54 2022] POST /key
=> generated 53 bytes in 376 msecs (HTTP/1.1 200) 2 headers in 79 bytes (1 switches on core 0)
▯
```

```
\xccta?2b\xcc\xa8]\xb7\xe7\x16\xa4\x00\xbe\x9c\xf6\xd4\xfc\x0c\x92\x84', '86f800431bae49518b614f268d1
88913': b'\xfa\x8e=\xca\x8clV2H\x8c\xd1\x1c\xf0\x14\x82\xb3\xd0"\x93\xca\r\xa2\xfck\xcc\x9eT\xea\xa9\
xf8fR', '0a50def776624c56a6c3826ce4a9bd92': b'\xec\x06^7\xc4}\xf4\xc8\x9d\x13f(\xdd=D\xef\x1a&f.N\xec
\xbc\xffO\xa9\xf6\x1e\x1aT\xb8\x1a', '15cbb4fa347b404fbcca681bc188a64b': b'\xb6w\x86\xc5\xd2\xb4\x87o
\xf3\xdd\xa3\xd8\xc9\xc1\x8d\x95"\x83p\xc3\xa6\xa3\x05\xca\xc4\x92\x17\x1d\x98]\xfa\xe2', 'a127438d0a
60425c9513504dfddf5bb1': b'.\x05\x97$\xa4\xa4a\xcf\xd42\xcf\xd2&\x14\xac\xc96\xe7\xd5\xc0\n"\x1dxN#1K
\x16bc"', 'd27e332316184a509bc51a12084d576f': b'\xdc\x8b|\x9d\xd2\x8b;)\xe6\x98R<D\x88}{r\xa2l\x1ba\x
b7["\xc7\xb0i\x85\xc2\xbb"\xc4', 'b78477b1703a4645bc21dabb48f894d9': b'\xfa\xe0\xdf\x1b\xadV\xb5\xd0\
x9f{q\xa2M\xebW\x8e\x80\x1a9\xdek{\x88>\xdar\xe5\x15\x92\x00)\xfc', '2d5cc3e12bc94156b745213a7cda42da
': b"m]\xf1\x94L\xbd\xd7\xde\xdb'\xf1\xfb\xe4\xf0\x1b\xab\xb15\xf6u\xfc\x06\xb2\xc3s\x824\xe7Y\xce1\x
b6", '88866d9452804de99ccd2161ea70b0ea': b'O\x10\xb3\x90\x94\xa7\xfb\xc3M/FRM\xf4d\xd3\xf7\xcd\xae\x8
9P\xc1\x83\x17v\xbfT\xa3\xafM\xaa0', 'fb34f3a0186842df85e204930bd8be72': b'\xb8\x14\x07\xa6\xc9\x8cr\
xcf\xcc\xe7\xa7\xf1\x97\xf5azI\xdd\x154\xbf\xce\xccs\x8b\xfb\x9e\xcf\x15\xba\xc4\x0e', '511180f2e78c4
63fb3eda4a721b586b3': b'\xac\x96f\x15\xb9v\xe6\x00d\x83\x04\xb4T\xbc5\xbdD\xc6\xe9\xd6\xe7R\xb6,\xdc\
x9aW G\xe9\x1b<', 'eb3486f22eaf4651a2f05674baf7abe3': b'\xf7q\xc2\xa6>\x84\xed\x0f\xe2\x9f\xca\xf1\x9
1h\xb4\xce\xd0U(%\xdd;\xb2\xde\xc1\x18\x04P\x19\xc4\x00\x12', '3214e857d7a5437b8f62f9c5a170441c': b'\
xd4\x1a\xc0\x97\x05G\x1bu\xf1\xe1F\x90\xdc;`\x93&[\xacb\x19:;\xd4\xc6\xd6\x16\xb1\x87\xf2\xec-', 'a0a
6fd6dc86f4c449564820b71cbee03': b'\x0f\x94M\xdb\xc8\xc3z$;\x9a\r]\x10\xd3i\xa1\xfb\xc3\xd6\xe5R\xb5)\
xff\x00\x98\x95\xb7\xe3\x96y\xdc', '839dfa089f41403a984ab276eda51c2f': b'\x08k:\xc8-}\x9d\x81<h\xd6\x
8e\xd3\x02 /\xa1\xd9\x9e\xa3\x03}\x8fb\xb5E\x84\xbd^\xcb\xe3|'}
[+] Key [1/2] for id d742dc4fb4b64706988447953287a3bf
▯
```

```
2\xfe@7=\x85\xe95\xd3\x9a.\xc4\xc1\xc3\xa9I\xd1\xa6\\\x9d0\xa3\xeb\xcc,\xc1T<`H\x93\xa3', 'ccd4bc38f3
ed410d983e34678c8f352a': b'\xfa\xf7-\r\x8b\xc7\n\xda\xe1\xe1?\xb3\x1c\xbc8mNL\xe3fK\n_e0Z\xba\x05\xe0
\xed\x8b\x8d', '88e976c810a043339e2a361127122304': b'\xb5\xffL7t"?Q\x86\xcc\xcf\xffF\xb4l6\xbb\xff\x8
9W\xa9\xbd6\x00\xaaF\xdc\x1c1\xf6\xc1\x08', '4198113b563841b9a3e40fddaeba5d01': b'\xb7\xb0\xe0C\x82\x
f6`8\x020\xa2f\xfe\x97\xd0@\xae\x90\xee\x87\xfe6$|hM\t\x8du\x90 \x13', 'fbbab285bd3d47f4b49632dc47749
632': b'\xec\xc2\xcd(No\xd1\xect\xf6l\x05\x1b\x8f\x00\xad\x00\x18<\x8c\xa4\x9eC7\xb7E\xe4"\x1fd\xbdT'
, '1725f34723cf494faf234f00bfd85a3d': b'\xd6\xfa5=k\x97$]i\xa3\xf7\xd6uz\xbe\x14\xc5M\xe1\xa1\x93\xf3
\tF\xc8\xd3u!B\x81\xf4.', '44b0b6180bde4334bbd0a8e313c14258': b'{6\xfe\xf8\xb1\x96~K\x9e\xc3\x07\xd9q
G\x023=\xf7\xcc\x85\x1c\xb1l3\xa4\x0c\xa9\x80/)f\x1b', '97a91105fb1f41e4a4071f8d3b440ec8': b'\xed\x12
c\xd8Z\xfc\x1a\xa4\xf8\xf4=\xbc&\x8e\x95\xd1G\xab\xfe\x9a@\x84\xf1\x963\xc7\x0e"\xdc3\x88Y', 'aa6e3f5
de1524f4fb7b8836c5a74b101': b'\xab\x05\xc6{h\x08\x16\t\xca\x08\x8d\xeaMnM\x94re\x0e\xc0\xd3\x99\x97\x
8d=\xb4\x821\xef\x87\xef\x83', 'bacde4af9a92442e918bacbd39051cd4': b'K\xbf\xfe\xd1\x81\x12\x15U\x9a\x
be;tQ\x04\x1bu\xb6\x06,V\xa8w\xfem\xa7\x8bn1\xd3\xaa\xb86', '76ddf792e90546d7850775d5c87f23ab': b'\xd
6@J\x907b\xb0\xb1\x16\xf5\xb6\xc2\xa2Y\xb9\xd35W1\x08\xa4\xe5\x7f\xab\xa3\xea\xe9w\x03\x8c0\xcd', '59
e7231344704ad99d723ae943065857': b'\xf4U`\x99b\xbc;\x04P\xc7v\x97\xa3\xffW`\xf4>\xa6\xcfP\xb4\xdf\x06
\x8f\x88\xff\xed\nsH-', '43db489a91754175ae2937e1ce990781': b"\xa8\xb6X!/\xdcT\x9c4\xcc.\xa1&\x9f\x14
\x90\x96\xb6\xed\xf8\x16:(4\x17@Xi\xfam'M", 'c2fad5a24e844f3a9ddb4a0c5d1d1239': b'\x06\xcd\x9f\xcc\xc
a\x9d\x96\xd4\xe7y%\xc6\x85\x08y\xe9yTf\xe0f\x16\x17\xf1\x8es\x84\xc7j-\x9a\xf9', '806bb85a894948f08c
ae87e109698fd5': b'\x82B\xf7\x87\x1d\xbb\xa9 \xb3V\xda\x9fRVI\x9em\x85\x15\n.S*\x83Ywp\xba\xeft\x8f\x
d0', '5b7fa2ef6bd3446bb98fb2b49ca17725': b'mu\x1a\xc6G<\xe91\x1c|\xb5\x9b\xac\x85JM\xc5\xe8\xb0\xfaop
\x7fl6w9<.\xf6\r\xef'}
▮
```

Figure 9. Bi-directional exchange of encryption keys between the Berlin and Poznan border nodes. The Poznan server (**top**, **right**) and the Berlin client (**bottom**, **left**) exchange keys. The server log states the reception of the ground link transmission (Keys 1/2); so does the client log. In this state of the exchange process, the space link keys transmission has not yet been finalized. The Berlin server (**top**, **left**) and the Poznan client (**bottom**, **right**) exchange keys (in the opposite direction). The server log already states the full reception of the keys of the second (space) segment, while the client shows the final (KDF-combined) block of 50 keys and their key identifiers, respectively.

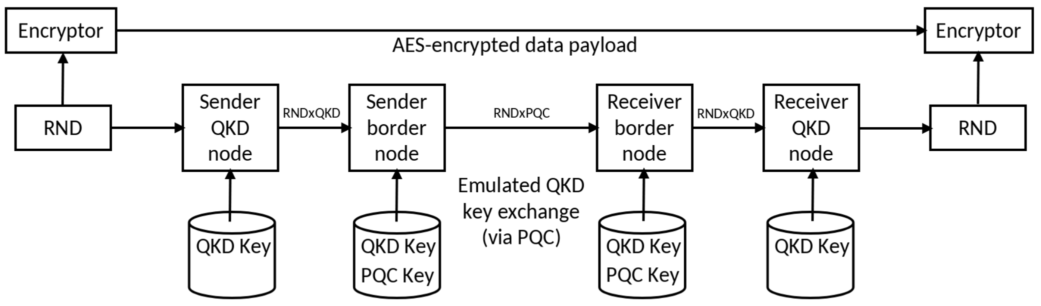


Figure 10. Integration of an emulated QKD key-exchange system into a QKD architecture. The keys of the emulated QKD key exchange are stored in the local key store of the border nodes. A sending QKD node forwards a random number through the trusted-node chain of the senders border node and the recipients border node to the receiving QKD node. The random number is either directly used as a final secure key or two random numbers, transferred across disjoint network links, are combined using a KDF for the final secure key.

The resulting network is fully meshed. Each gateway node operates a PQC key exchange server that listens for incoming connections. Additionally, each gateway node could initiate a key exchange through the key exchange client, yielding a bi-lateral key exchange. The integration of the PQC key exchange works through standard interfaces like any other key supplier. The testbeds integrate the PQC key exchange by pushing the keys, their identifiers, and their metadata directly into the KMS at every location, where they can be consumed by encryptors or applications via the ETSI GS QKD 004 or 014 API [19,24] and an appropriate key negotiation process. Figure 11 shows an overview of the four border node solutions deployed in this project. (Note that the application-based long-haul border node connection is always realized following the multi-path diversity approach).

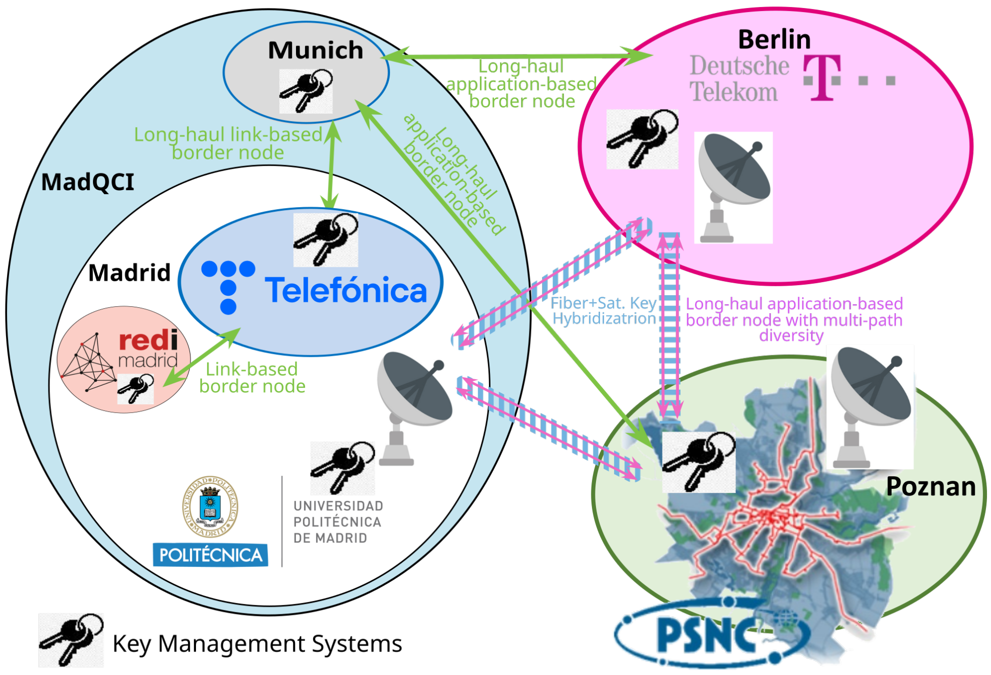

Munich
Long-haul application-based border node
Berlin
Deutsche Telekom
Long-haul link-based border node
Long-haul application-based border node
MadQCI
Madrid
Telefónica
redi madrid
Link-based border node
Fiber+Sat. Key Hybridizatrion
Long-haul application-based border node with multi-path diversity
UNIVERSIDAD POLITÉCNICA DE MADRID
POLITÉCNICA
Poznan
PSNC
Key Management Systems

Figure 11. Long-distance (emulated) QKD links connecting the metropolitan-area QKD networks of Madrid and Munich, Berlin, and Poznan.

Here, a few general comments are provided on the overall standing of the four proposed methods. We underline that these were adapted to the present situation, when long-distance QKD links do not yet exist in Europe, whereby these could be applied also if the availability of long-distance QKD is scarce. The situation changes if long-distance QKD is available. We would first point out that in this case Method 1 will be identical to Method 2. On the other hand, generally, Method 1 and 2 can also be applied to dissimilar networks if appropriate extension of the work, presently performed on ETSI GS QKD 020, is carried out. Possibly, however, this is not needed, as one can always apply Method 3. In this sense the prospective role of Method 3 depends on decisions to be taken by standardization bodies. Method 4 details the multi-path diversity extension that (as mentioned already in Section 1) can always be used to enhance security.

As far as Table 1 in Section 1 is concerned, we draw the attention of the reader to the fact that we recommend the utilization of row number 3 there. Until long-distance QKD is available to a sufficient extent, we will be forced to the situation represented by row number 1 in Table 2. For the same reason, for the long-distance connections we are using in this work, we rely on the approach of Table 1 row 2. Note independently that if Method 1 only relies on QKD, we would be using row 1 of Table 1.

3.2. Hybridization of All Generated Keys

Security is usually a crucial parameter in network communications. We, therefore, propose to integrate a hybridization scheme to compute the hybrid final keys at each network node. Previous approaches [34] combine PKC keys with QKD and indicate the possibility even of using PQC. In the present work, the hybridization (i) follows the recent Muckle scheme [14] that puts forward a hybrid method for authenticated key exchange (specifically, it is based on a secure hybridization, i.e., combination, of several keys of QKD, PQC, and PKC origin, and additionally, the authentication of the key distribution using PSK, instead of PQC SIG), and (ii) relies on extended hybridization involving multiple media/paths. We also explicitly promote this strategy by careful analysis of the potential security benefits of combining practical implementations of the protocols (not to be confused with the mathematically formulated protocols themselves).

The key exchange module between PoPs was modified to manage not only the QKD keys, but also the PQC ones. The systems also allow the use of several key exchange modules in parallel, even with the same PoPs. The KMS systems receive the keys (QKD and PQC or any other key exchange mechanism) in a transparent manner from these key exchange modules, so the KMS can establish different quantum-safe key exchange sessions with other PoPs, using QKD links, PQC links, or a combination of both in a simple way.

To do that, the key exchange modules run in parallel on each node delivering keys to the KMS. The payload functionality of the key exchange modules is vendor- and technology-independent, and the only exposed interfaces are the ETSI GS QKD 004 or 014 ones. The KMSes only receive notice of a new link between two PoPs. This design enables the generic integration of additional links, and the keys generated by PQC are internally managed by a KMS exactly as any key material generated by QKD. All these interfaces can be seen as quantum-safe key delivery interfaces, whereby (Q)RNG serves as the key source [35]. The PQC link is implemented as a TCP connection, with package delivery granted and in order. This allows simulation of a potentially infinite protected stream of keys with PQC between any two peers of the network. Identity authentication of the endpoints is required only when a new TCP stream is started. The use of PQC links enables having long-distance quantum-safe links where QKD cannot reach right now. The current design, exposing the standard QKD interfaces, makes it possible to replace PQC by QKD links when the technology is mature enough, with minimal impact on the rest of the architecture. The hybridization of the QKD and PQC keys will, as described earlier, deliver a key exchange system with significantly reduced side channels due to the use of principally different technologies and implementations thereof.

The key hybridization process is performed by applying a hybridization KDF. The key hybridization process is performed as an internal process on the KMS that manages different internal key stores. The key hybridization process needs to process the appropriate key bits so that a new, hybrid key is computed, stored, and handed over to the applications and encryptors. Alternatively, hybridization may be left to the application, since the application oversees enforcing the required security level itself by picking a key or a combination of keys exchanged under the right security paradigm.

4. Conclusions

QKD networks are a key building block for quantum-safe communications. The interest in related research fields and subsequent industrialization is increasing rapidly. There is an emerging necessity for deploying QKD metro networks, but also for connecting these metro networks over long distances. The present article demonstrates several viable approaches for achieving quantum-safe key exchange between three of the major production-grade QKD testbeds in Europe: Berlin, Madrid, and Poznan. These three testbeds are different in terms of their network architecture, functionalities, and management, reflecting the different ways that a telecommunications company may operate its infrastructure. Different key exchange realizations were defined, including key exchange over different physical media (fiber and satellite) to enable E2E communication between all the nodes of all the participating networks.

A cross-European, E2E key exchange was designed, which follows the SDN principles adapted to QKD. This approach enables the interconnection of nodes belonging not only to the same but also to different networks by QKD-powered quantum-safe links. As an example, two network domains in Madrid, REDIMadrid and Telefónica, were connected through an SDN-based QKD layer. This approach was extended to include PQC in parallel with QKD, combining both technologies simultaneously to augment the strength of each link. Using the SDN paradigm, Madrid and Munich were connected through emulated QKD links to demonstrate the security transparency for long-distance links also.

Additionally, an E2E key exchange based on the application layer was proposed. This method is very generic and can easily be adapted to any infrastructure. It does not require any specific type of network design but requires local management of key-forwarding requests.

Diversifying the key generation across disjoint network paths, e.g., via fiber, satellite, or mobile network links, adds extra security. Moreover, if the key generation over such paths is based on different technologies, such as PQC and prospective long-distance QKD, this effect will be enhanced. Multiple different key-combination variants based on such multi-path and multi-technology key streams were realized in the presented connection of the testbeds. This approach also leads to a significant reduction in side channels that would come about in real-world implementations of the ideal protocols.

The findings presented in this work open the door to long-haul interconnectivity between QKD metro networks. Multiple quantum-safe technologies were combined in parallel to define the next generation of highly secure, pan-European interconnectivity. The proposal can readily be adopted and offers a scalable security layer that is extendable to the entire continent and even across continents.

Author Contributions: Concept development, M.B., M.G., P.R., H.H.B., M.P. and J.P.B.; Methodology, M.B., R.-P.B., H.H.B. and M.P.; Software, M.B., R.J.V., J.S.B., R.B.M., M.G., F.F. and J.P.B.; Validation, J.S.B., R.B.M. and P.R.; Formal analysis, M.B., H.H.B., M.P., V.M. and J.P.B.; Investigation, M.B., R.J.V., J.S.B., R.B.M., R.-P.B., M.G., P.R., H.H.B., V.M. and J.P.B.; Resources, R.J.V., J.S.B. and R.B.M.; Data curation, M.B. and R.J.V.; Writing—original draft, M.G.; Writing—review & editing, R.J.V., J.S.B., R.B.M., H.H.B., M.P. and V.M.; Supervision, M.G., A.P., D.R.L. and V.M.; Project administration, M.G. All authors have read and agreed to the published version of the manuscript.

Funding: We acknowledge the support from the "MADQuantum-CM" project funded by Comunidad de Madrid (Programa de Acciones Complementarias) and by the Recovery, Transformation and Resilience Plan—Funded by the European Union—(NextGenerationEU, PRTR-C17.I1), the QUBIP project, funded by the European Union under the Horizon Europe framework program under Grant Agreement No. 101119746, QuantERA II Programme, European Union's Horizon 2020 research and innovation program under Grant Agreement No 101017733, and with funding organizations: The Foundation for Science and Technology—FCT, Agence Nationale de la Recherche—ANR, and Spanish Agencia Estatal de Investigación—AEI, and the European Union's Horizon Europe research and innovation funding program under the project "Quantum Secure Networks Partnership" (QSNP, grant agreement No 101114043).

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: Authors Max Brauer, Ralf-Peter Braun, Marc Geitz were employed by the company T-Labs, Deutsche Telekom AG. Authors Hans H. Brunner, Fred Fung, Momtchil Peev were employed by the company Munich Research Center, Huawei Technologies Duesseldorf GmbH. Authors Antonio Pastor and Diego R. Lopez were employed by the company Telefónica gCTIO/I+D. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Shor, P.W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Rev.* **1999**, *41*, 303–332. [CrossRef]
2. Moody, D. Let's Get Ready to Rumble. The NIST PQC "Competition". In Proceedings of the First PQC Standardization Conference, Fort Lauderdale, FL, USA, 12–13 April 2018; pp. 11–13.
3. NIST, Information Technology Laboratory, C.S.R.C. Post Quantum Cryptography, Draft FIPS 203, FIPS 204 and FIPS 205, Which Specify Algorithms Derived from CRYSTALS-Dilithium, CRYSTALS-KYBER and SPHINCS+. 2023. Available online: https://csrc.nist.gov/projects/post-quantum-cryptography (accessed on 8 November 2023).
4. Braun, R.P.; Geitz, M. The OpenQKD Testbed in Berlin. In Proceedings of the 2021 Asia Communications and Photonics Conference (ACP), Shanghai, China, 24–27 October 2021; pp. 1–3.
5. Rydlichkowski, P. OPENQKD project Work Package 7 review. In Proceedings of the QKD Days, Madrid, Spanish, 13 December 2022.
6. Martin, V.; Brito, J.P.; Ortíz, L.; Brito-Méndez, R.; Sáez-Buruaga, J.; Vicente, R.; Sebastián-Lombraña, A.; Rincón, D.; Pérez, F.; Sánchez, C.; et al. MadQCI: A Heterogeneous and Scalable SDN QKD Network Deployed in Production Facilities, 2023. Available online: https://arxiv.org/abs/2311.12791v2 (accessed on 8 November 2023).

7. Aguado, A.; Lopez, V.; Lopez, D.; Peev, M.; Poppe, A.; Pastor, A.; Folgueira, J.; Martin, V. The Engineering of Software-Defined Quantum Key Distribution Networks. *IEEE Commun. Mag.* **2019**, *57*, 20–26. [CrossRef]
8. Qi, W. Overview of Quantum Communication Industry Development in China. In Proceedings of the ETSI QSC Workshop 2023. 2023. Available online: https://docbox.etsi.org/Workshop/2023/02_QUANTUMSAFECRYPTOGRAPHY/TECHNICALTRACK/WORLDTOUR/CASQUANTUMNETWORK_QI.pdf (accessed on 8 November 2023).
9. ETSI Group Specification QKD-016: Common Criteria Protection Profile V1.1.1. 2023. Available online: https://www.etsi.org/deliver/etsi_gs/QKD/001_099/016/01.01.01_60/gs_QKD016v010101p.pdf (accessed on 8 November 2023).
10. *ISO/IEC 23837-1:2023*; Information Security—Security Requirements, Test and Evaluation Methods For Quantum Key Distribution—Part 1: Requirements. International Organization for Standardization: Geneva, Switzerland. Available online: https://www.iso.org/standard/77097.html (accessed on 8 November 2023).
11. *ISO/IEC 23837-2:2023*; Information Security—Security Requirements, Test and Evaluation Methods For Quantum Key Distribution—Part 2: Evaluation and Testing Methods. International Organization for Standardization: Geneva, Switzerland. Available online: https://www.iso.org/standard/77309.html (accessed on 8 November 2023).
12. European Commission. OpenQKD, 2019–2023. Available online: https://openqkd.eu (accessed on 8 November 2023).
13. European Commission. European Quantum Communication Infrastructure (EuroQCI). 2023. Available online: https://digital-strategy.ec.europa.eu/de/policies/european-quantum-communication-infrastructure-euroqci (accessed on 8 November 2023).
14. Dowling, B.; Hansen, T.B.; Paterson, K.G. Many a Mickle Makes a Muckle: A Framework for Provably Quantum-Secure Hybrid Key Exchange. Cryptology ePrint Archive, Paper 2020/099. 2020. Available online: https://eprint.iacr.org/2020/099 (accessed on 8 November 2023).
15. Müller-Quade, J.; Renner, R. Composability in quantum cryptography. *New J. Phys.* **2009**, *11*, 085006. [CrossRef]
16. Pacher, C.; Abidin, A.; Lorunser, T.; Peev, M.; Ursin, R.; Zeilinger, A.; L:arson, J.A. Attacks on quantum key distribution protocols that employ non-ITS authentication. *Quantum Inf. Process.* **2016**, *15*, 327–362. [CrossRef]
17. Braun, R.P.; Geitz, M.; Döring, R. Berlin OpenQKD Testbed Evaluating Quantum Key Distribution in Provider Networks. In *ICSCC 2023, Proceedings of the 8th International Conference on Systems, Control and Communications, Chongqing, China, 20–22 October 2023*; International Conference Proceedings Series; ACM: New York, NY, USA, 2023; ISBN 979-8-4007-0781-0.
18. The Open Quantum Safe Project. liboqs—An Open Source C Library for Quantum-Safe Cryptographic Algorithms. 2022. Available online: https://github.com/open-quantum-safe/liboqs (accessed on 8 November 2023).
19. ETSI Group Specification on QKD-014: Protocol and Data Format of REST-Based Key Delivery API V1.1.1. 2019. Available online: https://www.etsi.org/deliver/etsi_gs/QKD/001_099/014/01.01.01_60/gs_qkd014v010101p.pdf (accessed on 8 November 2023).
20. National Laboratory for Photonic and Quantum Technologies NLPQT Project. 2023. Available online: http://nlpqt.fuw.edu.pl/en/ (accessed on 8 November 2023).
21. ETSI Group Specification on QKD-020: Interoperable KMS API (Draft). 2023. Available online: https://portal.etsi.org/webapp/WorkProgram/Report_WorkItem.asp?WKI_ID=63115 (accessed on 8 November 2023).
22. ITU-T Rec.Y.3800, Standardization Sector, Overview on Networks Supporting Quantum Key Distribution. 2019. Available online: https://www.itu.int/itu-t/recommendations/rec.aspx?id=13990&lang=en (accessed on 8 November 2023).
23. Peev, M.; Pacher, C.; Alléaume, R.; Barreiro, C.; Bouda, J.; Boxleitner, W.; Debuisschert, T.; Diamanti, E.; Dianati, M.; Dynes, J.F.; et al. The SECOQC quantum key distribution network in Vienna. *New J. Phys.* **2009**, *11*, 075001. [CrossRef]
24. ETSI Group Specification on QKD-004: Application Interface V2.1.1. 2020. Available online: https://www.etsi.org/deliver/etsi_gs/QKD/001_099/004/02.01.01_60/gs_qkd004v020101p.pdf (accessed on 8 November 2023).
25. ETSI Group Specification on QKD-015: Control Interface for Software Defined Networks V2.1.1. 2022. Available online: https://www.etsi.org/deliver/etsi_gs/QKD/001_099/015/02.01.01_60/gs_QKD015v020101p.pdf (accessed on 8 November 2023).
26. Carter, J.L.; Wegman, M.N. Universal Classes of Hash Functions. *J. Comput. Syst. Sci.* **1979**, *18*, 143–154. [CrossRef]
27. Dóring, R.; Geitz, M. Post-quantum cryptography in use: Empirical analysis of the tls handshake performance. In Proceedings of the NOMS 2022—2022 IEEE/IFIP Network Operations and Management Symposium, Budapest, Hungary, 25–29 April 2022; pp. 1–5.
28. Chen, L. *Recommendation for Key Derivation Using Pseudorandom Functions*; Special Publication 800-108; NIST: Gaithersburg, MD, USA, 2009.
29. ETSI TS Cyber, TS 103 744: Quantum-Safe Hybrid Key Exchanges, V1.1.1 (2020-12). 2020. Available online: https://www.etsi.org/deliver/etsi_ts/103700_103799/103744/01.01.01_60/ts_103744v010101p.pdf (accessed on 8 November 2023).
30. Federal Office for Information Security (BSI). Cryptographic Mechanisms: Recommendations and Key Lengths. 2022. Available online: https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TG02102/BSI-TR-02102-1.pdf (accessed on 8 November 2023).
31. Döring, R.; Geitz, M.; Braun, R.P. Post-Quantum Cryptography key exchange to extend a high security QKD plattform into the mobile 5G and 6G networks. In *ICCNT 2023, Proceedings of the 7th International Conference on Communication and Network Technology, Madrid, Spain, 18–20 September 2023*; Lecture Notes on Data Engineering and Communications Technologies; Springer: Cham, Switzerland, 2023.
32. Geitz, M.; Döring, R.; Braun, R.P. Hybrid QKD & PQC Protocols Implemented in the Berlin OpenQKD Testbed. In Proceedings of the ICFSP. 2023. Available online: https://ieeexplore.ieee.org/document/10372894 (accessed on 8 November 2023).

33. Yadav, A.; Agarwal, M.; Agarwal, S.; Verma, S. Internet From Space Anywhere and Anytime—Starlink. In Proceedings of the Advancement in Electronics & Communication Engineering, Ghaziabad, India, 14–15 July 2022. [CrossRef]
34. Aguado, A.; Lopez, V.; Martinez-Mateo, J.; Szyrkowiec, T.; Autenrieth, A.; Peev, M.; Lopez, D.; Martin, V. Hybrid conventional and quantum security for software defined and virtualized networks. *J. Opt. Commun. Netw.* **2017**, *9*, 819–825. [CrossRef]
35. Herrero-Collantes, M.; Garcia-Escartin, J.C. Quantum random number generators. *Rev. Mod. Phys.* **2017**, *89*, 015004. [CrossRef]