Quantum Key Distribution Networks - Key Management: A Survey

EMIR DERVISEVIC, Department of Telecommunications, Faculty of Electrical Engineering, University of Sarajevo, Sarajevo, Bosnia and Herzegovina
AMINA TANKOVIC, Department of Telecommunications, Faculty of Electrical Engineering, University of Sarajevo, Sarajevo, Bosnia and Herzegovina
EHSAN FAZEL, Cisco Systems, Cambridge, United Kingdom of Great Britain and Northern Ireland
RAMANA KOMPELLA, Cisco Quantum Lab, Los Angeles, United States
PEPPINO FAZIO, DSMN, Ca' Foscari University of Venice, Venice, Italy and Department of Telecommunications, VSB - Technical University of Ostrava, Ostrava, Czech Republic
MIROSLAV VOZNAK, Department of Telecommunications, VSB - Technical University of Ostrava, Ostrava, Czech Republic
MIRALEM MEHIC, Department of Telecommunications, Faculty of Electrical Engineering, University of Sarajevo, Sarajevo, Bosnia and Herzegovina and Department of Telecommunications, VSB - Technical University of Ostrava, Ostrava, Czech Republic

Secure communication makes the widespread use of telecommunication networks and services possible. With the constant progress of computing and mathematics, new cryptographic methods are being diligently developed. Quantum Key Distribution (QKD) is a promising technology that provides an Information-Theoretically Secure (ITS) solution to the secret-key agreement problem between two remote parties. QKD networks based on trusted relay nodes are built to provide service to a larger number of parties at arbitrary distances. They function as an add-on technology to traditional networks, generating, managing, distributing, and supplying ITS cryptographic keys. Since key resources are limited, integrating QKD network services into critical infrastructures necessitates effective key management. As a result, this article provides a

The research was co-funded by the European Union (EU) within the REFRESH project — Research Excellence For REgion Sustainability and High-tech Industries ID No. CZ.10.03.01/00/22_003/0000048 of the European Just Transition Fund and also supported by the Ministry of Education, Youth and Sports of the Czech Republic (MEYS CZ) within a Student Grant Competition in the VSB — Technical University of Ostrava under project ID No. SGS SP2025/013.

Authors' Contact Information: Emir Dervisevic, Department of Telecommunications, Faculty of Electrical Engineering, University of Sarajevo, Sarajevo, Bosnia and Herzegovina; e-mail: emir.dervisevic@etf.unsa.ba; Amina Tankovic, Department of Telecommunications, Faculty of Electrical Engineering, University of Sarajevo, Sarajevo, Bosnia and Herzegovina; e-mail: atankovic1@etf.unsa.ba; Ehsan Fazel, Cisco Systems, Cambridge, United Kingdom of Great Britain and Northern Ireland; e-mail: salavifa@cisco.com; Ramana Kompella, Cisco Quantum Lab, Los Angeles, California, USA; e-mail: rkompell@cisco.com; Peppino Fazio, DSMN, Ca' Foscari University of Venice, Venice, Italy and Department of Telecommunications, VSB — Technical University of Ostrava, Ostrava, Czech Republic; e-mail: peppino.fazio@unive.it; Miroslav Voznak, Department of Telecommunications, VSB—Technical University of Ostrava, Ostrava, Czech Republic; e-mail: miroslav.voznak@vsb.cz; Miralem Mehic, Department of Telecommunications, Faculty of Electrical Engineering, University of Sarajevo, Sarajevo, Bosnia and Herzegovina and Department of Telecommunications, VSB — Technical University of Ostrava, Ostrava, Czech Republic; e-mail: miralem.mehic@vsb.cz.

ACM 0360-0300/2025/05-ART257
https://doi.org/10.1145/3730575

comprehensive review of key management approaches for trusted-relay QKD networks. They are analyzed to facilitate the identification of potential strategies and accelerate the future development of QKD networks.

CCS Concepts: • **Security and privacy → Key management**; *Security protocols*; *Information-theoretic techniques*; • **General and reference → Surveys and overviews;**

Additional Key Words and Phrases: Quantum key distribution networks, key management system design

ACM Reference Format:
Emir Dervisevic, Amina Tankovic, Ehsan Fazel, Ramana Kompella, Peppino Fazio, Miroslav Voznak, and Miralem Mehic. 2025. Quantum Key Distribution Networks - Key Management: A Survey. *ACM Comput. Surv.* 57, 10, Article 257 (May 2025), 36 pages. https://doi.org/10.1145/3730575

1 Introduction

Secure means of communication are becoming increasingly important as data traffic in communication networks grows and more services emerge due to their integration [1]. Sustaining widespread security mechanisms based on complex mathematical problems is proving challenging. Significant advances in computing and mathematics make it more challenging to ensure their security [2]. As a result, security experts have begun developing new cryptographic algorithms to address these challenges. The emergence of quantum computers is the most severe threat motivating these actions. Already-designed quantum algorithms pose a significant threat to public-key cryptosystems [3]. It is simply that there is not a large-scale quantum computer to run them, at least not for practical applications.

Over the last 2 decades, tremendous efforts have been made to develop new cryptographic, quantum-secure mechanisms that will eventually replace the existing ones. They have resulted in two frameworks for secure communication in the post-quantum world: **Post-Quantum Cryptography (PQC)** and Quantum Cryptography. PQC concepts are based on a similar approach to that of classical algorithms: complex mathematical problems that cannot be solved in practical time by both classical and quantum computers [4]. However, there is always the possibility that new quantum algorithms will be discovered in the near future, compromising their security. In contrast, quantum cryptography is based on the principles of quantum physics. Because the laws of quantum physics are unbreakable, the technology offers a long-term security solution. It is unaffected by advancements in computing or mathematics. However, it has considerable limitations.

Quantum Key Distribution (QKD) [5] is the most mature example of quantum technologies. It has been in experimental testing for over 2 decades and has only recently been used in commercial applications. QKD is a method for agreeing on secret keys, a problem that cryptographers have long faced. The real advantage of QKD over traditional key-agreement protocols is that the established keys are **Information-Theoretically Secure (ITS)** [6]. It is unique in many other aspects, including the method of implementation. QKD necessitates specialized hardware, whereas traditional mechanisms are typically implemented in software and use Internet services to negotiate a secret key.

QKD requires two channels: a quantum channel and an authenticated public channel, as shown in Figure 1(a). These two channels are commonly referred to as a *logical QKD link*. Quantum transmission carried over the quantum channel cannot be passively monitored. When quantum carriers are monitored, they change state and, with high probability, reveal the presence of an eavesdropper. For determining whether an eavesdropper is present and, if not, to correlate the data exchanged over the quantum channel, the authenticated public channel is required [7–10]. The result is an ITS secret key, i.e., a true random sequence of bits known only to two legitimate parties. In the following discussion, these keys between two directly linked users are referred to as *local keys*.

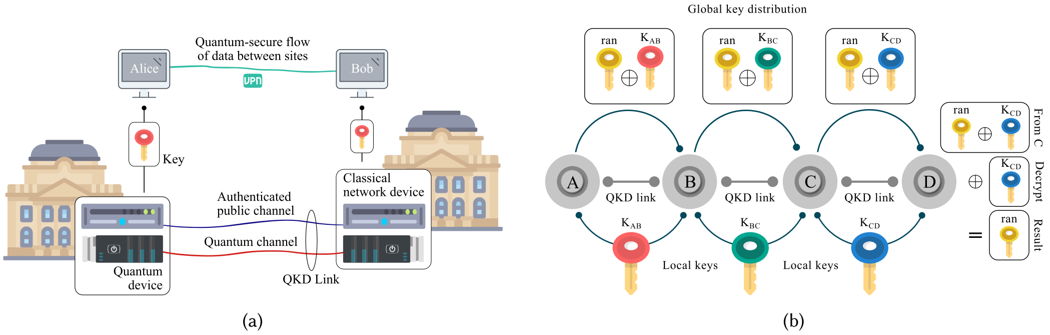


Fig. 1. (a) Quantum key distribution between two remote sites. QKD-derived key material is used to establish quantum-secure flow of data between two applications [11]. (b) Hop-by-hop global key distribution of a random key generated at a source node.

The traits that distinguish QKD from traditional approaches also add to the difficulties of large-scale deployment and application. QKD, which requires a direct physical connection between two users, is primarily a point-to-point technology. Furthermore, the properties of quantum transmission prevent the use of classic amplifiers, limiting the range of the technology [12]. As a result, QKD networks based on trusted relay nodes are being built to achieve the goal of global QKD deployment, overcoming connectivity and distance limitations [13]. Key distribution or key relaying over a network of trusted nodes is shown in Figure 1(b). The source node generates a random secret transmitted to the destination. The random secret, known as a *global key*, is **One-Time-Pad (OTP)** encrypted between each pair of trusted-relay nodes using local keys. Assuming that intermediate nodes are trusted and the random key is a genuinely random sequence of bits, the source and distant destination nodes establish an ITS global key. Instead of distributing a random key, the local key that the source shares with its first neighbor on route to the destination can be used.

Although the majority of attention is still focused on the implementation of QKD itself, intending to achieve greater distances [14–19] and key rates [20–22], the level of attention devoted to the operation of QKD networks is gradually increasing [23]. The establishment of testbeds worldwide [24] has encouraged the development of key functionalities that must be addressed to achieve applicable technology.

1.1 Motivation

Key management is one of the essential functionalities of QKD networks [25] that is often neglected. However, the issue of effective key management must be addressed for the QKD network service to gain traction in modern telecommunications networks [26]. A key manager is a device that performs various functions over keys, including key storage, key lifecycle management, key relaying, and key supply. Based on the comprehensive list of tasks, it is apparent that the key manager is a critical component of the QKD network infrastructure. It works with a finite quantity of cryptographic keys and provides them on request in accordance with the policy in place. As a result, service viability depends on effective key manager design. This article examines and compares existing designs in terms of supported functionalities.

In this survey, we address the following questions:

- How does the nature of a QKD process establish the need for key management?
- How does key management in the context of QKD networks differ from traditional approaches?
- What are the functional requirements for key management in a wide-scale QKD network?

Table 1. Condensed Compilation of Terms Encompassing Prior Survey Research

Reference	Year	QKD fundamentals	QKD (point-to-point) achievements and implementations	QKD network fundamentals	QKD standardization	QKD key management
Alléaume et al. [27]	2014	✓	✓			
Morris et al. [28]	2014	✓	✓	✓		
Diamanti et al. [29]	2016	✓	✓	✓		
Geihs et al. [30]	2019	✓	✓	✓	✓	
Xu et al. [31]	2020	✓	✓			
Cavaliere et al. [32]	2020	✓	✓	✓	✓	
Sharma et al. [33]	2021	✓	✓	✓		
Amer et al. [34]	2021	✓	✓	✓		
Tsai et al. [35]	2021	✓	✓	✓		
Cao et al. [36]	2022	✓	✓	✓	✓	
Mehic et al. [26]	2023	✓		✓	✓	
Our survey	2024	✓		✓	✓	✓

Comparison to this survey.

- How have different testbeds and works approached the issue of key management? The discussion includes the approaches to key storage design, support for multiple applications, and more.
- How do these different approaches compare with each other in terms of functionality?
- What are the key challenges that are still not addressed for functional and effective key management?

1.2 Comparison with Existing Surveys

The surveys listed below and summarized in Table 1 cover the topic of quantum technologies.

- Alléaume et al. [27] conducted an analysis and comparison of secret-key agreement techniques, QKD being one of them, assessing its performance (recorded until 2014). Additionally, the study discusses approaches for constructing QKD networks and outlines two applications of QKD-derived key material in securing communication.
- Morris et al. [28] examined different QKD protocols and network deployments, with a particular emphasis on the scale and performance of links.
- Diamanti et al. [29] discussed QKD protocols and their experimental deployments as well as approaches for constructing QKD networks.
- Geih et al. [30] presented the achieved QKD performances, approaches for constructing QKD networks, and efforts towards standardization.
- Xu et al. [31] examined the security of practical implementations of QKD using realistic flawed devices.
- Cavaliere et al. [32] presented the achieved link performances and challenges of QKD running within the same fiber as classical channels. Furthermore, the study discusses the physical characteristics of optical components and it briefly examines represented network deployments and ongoing standardization efforts.
- Sharma et al. [33] analyzed QKD protocols and methods of integrating the technology into optical networks, addressing issues such as wavelength and time slot assignment for the quantum channel. The study further discusses networking aspects and summarizes existing real-world integrations of QKD into optical networks.
- Amer et al. [34] discussed various QKD protocols and their implementations as well as established manufacturers in the field of QKD. The study also briefly explains QKD network architecture and represented testbeds worldwide.
- Tsai et al. [35] explored networking aspects and summarized key results achieved with represented testbeds. The primary focus is on network structure, achieved key rates and distances, and routing.

- Cao et al. [36] provided a comprehensive overview of state-of-the-art QKD protocols, performance, and practices for integrating quantum channels with classical channels within the same optical infrastructure. The study also describes QKD network architecture and essential building blocks. However, the key management layer is only briefly described in terms of basic requirements and lacks detailed analysis. Additionally, the survey covers ongoing progress in QKD standardization.
- Mehic et al. [26] conducted a survey on the integration of QKD with 5G networks. The study covers the fundamentals of QKD protocols and networking, discusses the integration of QKD in optical networks, and briefly touches on standardization efforts.

Based on the selected surveys, the narrowed list of research objectives discussed includes:

- Analyzing QKD protocols and their implementations.
- Examining the state-of-the-art advancements in QKD technology and network deployments.
- Exploring real-world applications of QKD networks.
- Summarizing standardization efforts related to QKD protocols and networks.

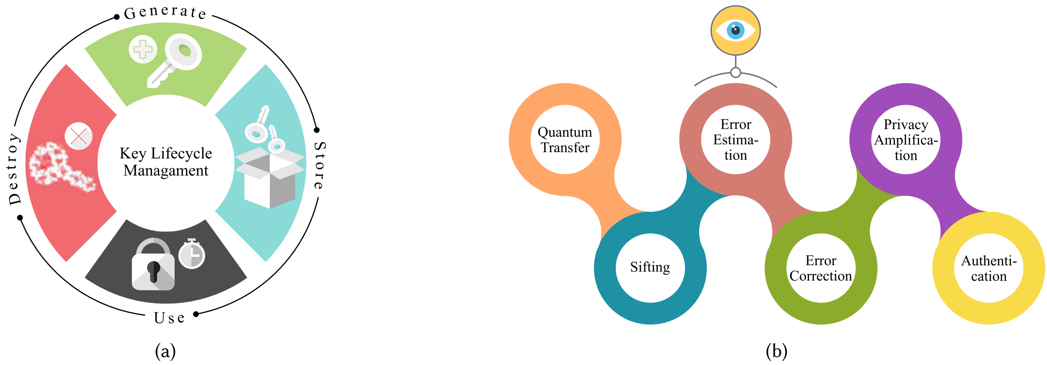
A thorough examination of the literature, however, reveals the lack of a comprehensive system-view engineering perspective on QKD key management. While the **International Telecommunication Union Telecommunication Standardization Sector (ITU-T)** has issued a recommendation to assist with the design, deployment, and operation of key management in QKD networks, this standard primarily outlines general key management functions and reference points for communication without offering detailed solutions for achieving these objectives [25]. To our knowledge, no references in the available literature thoroughly investigate the approaches to addressing key management issues. Consequently, this survey offers a chronological overview of existing approaches to the realization of key managers, unveiling the evolution of their functionality. This evolution begins with the need for simple key storage and progresses to sophisticated mechanisms aimed at improving efficiency and providing reliable service to a larger number of users. The identification of basic approaches and contributions from existing solutions led to the identification of existing gaps. Existing approaches were compared and analyzed to identify suitable approaches for developing an efficient key management system.

This survey provides interested readers with a high-level system engineering viewpoint on the trusted-relay QKD network and its organization. Researchers, practitioners of quantum technology, and PhD students in the field of applied networking security will benefit from this survey and synthesis of perspectives on the confluence of modern technologies.

1.3 Contribution

The major contributions of this survey are outlined as follows:

(1) Offers a comprehensive insight into the operation and application of QKD networks through the perspective of key management.
(2) Provides a detailed overview of the functionality of key managers and elucidates the reasons for their existence.
(3) Provides an overview of existing approaches to implementing key manager functionality.
(4) Unravels the networking details of existing testbeds that are usually missing from the review literature.
(5) Discusses and analyzes existing solutions to identify suitable approaches for key management in QKD networks.
(6) Outlines the existing gaps in the research to provide clear guidelines for future work in the field of QKD key management.



(a) (b)

Fig. 2. (a) Most essential cryptographic key lifecycle management. (b) QKD process is performed in several sequential steps. The process begins with the quantum transmission of a random sequence of key bits. The correctness of information received through quantum transmission is highly dependent on measurement. Information obtained from incompatible measurements is discarded during a sifting procedure. The next step involves estimating the error rate and using its value to discover the eavesdropper. If the error rate is less than the threshold, the process proceeds to the error correction step. At the very end, a privacy amplification step is performed. The entire process must be authenticated.

1.4 Article Organization

The article is organized as follows. Section 2 describes key management issues and highlights the distinctions between QKD and classic key management. Section 3 provides a comprehensive review of existing key management solutions in chronological order of appearance. Section 4 provides a comparative analysis of existing solutions, taking into account several key requirements and functionalities of key managers. This discussion has led to the identification of key challenges, which are outlined as guidelines for future directions in Section 5. Section 6 concludes our study.

2 Key Management

Key management is a widely recognized concept in nearly all systems that support secure communication services. One framework we can use as an example is **IPsec (Internet Protocol security)**, which manages the cryptographic keys needed to establish **Virtual Private Networks (VPNs)** [37]. To establish a secure VPN tunnel, the two peers must negotiate a key using the Diffie-Hellman key exchange protocol [38]. The established key must be kept safe for the duration of its lifetime. The lifetime is expressed in data units (bytes) or elapsed time (seconds). It specifies how long a key can be used before it expires. Using an expired key is not recommended because it reduces encryption security and puts previously transmitted data at risk. Expired keys must be properly destroyed to prevent disclosure to third-party attackers who may use the "store now, decrypt later" strategy to decrypt harvested data using leaked keys. Once the keys have been destroyed, new ones should be established to continue the encryption of the dataflow. Figure 2(a) illustrates this essential key lifecycle management and is recognized in IPsec and all security frameworks.

In contrast, the key management issues in QKD networks are highly distinct owing to the intrinsic uniqueness of the QKD process. If QKD worked similarly to traditional key establishment techniques, it would replace the Diffie-Hellman key exchange from the earlier example. However, the QKD process consists of several steps, as illustrated in Figure 2(b), which could take several minutes until the process outputs ITS keys. This is primarily why key management in QKD networks is a critical enabler of its services: generate larger amounts of cryptographic keys ahead of

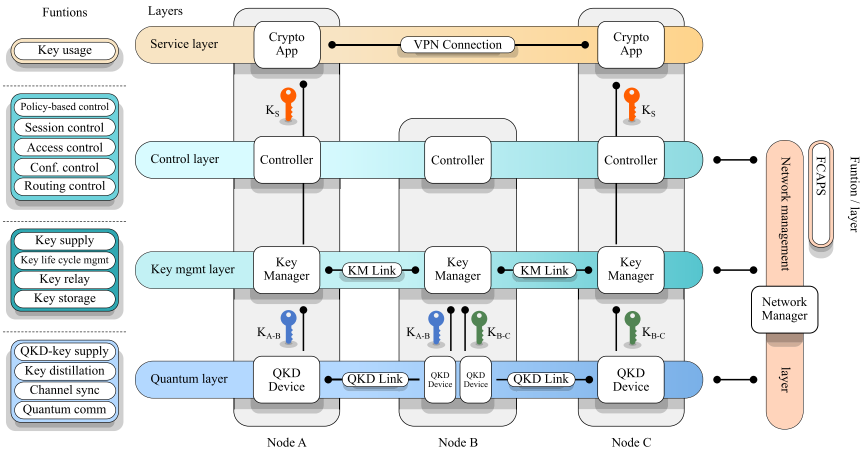


Fig. 3. The layered architecture of QKD network according to the ITU-T Y.3800 recommendation [39]. In this illustration, the User Network Management Layer is omitted, as it does not directly influence the QKD network segment. It resides within the user network, managing the service layer. However, the ITU-T specifies a potential communication point between the management layers of the two networks, indicating how these layers can interact if necessary.

time and then supply them in a timely manner on demand to traditional security frameworks such as IPsec.

Key management sits in the middle of the layers of the QKD network architecture as illustrated in Figure 3 [39]. All layers presented are fundamentally responsible for or influence key management in some manner [40]. However, while discussing key management in the context of QKD networks, one must surely refer to the key management layer. Only the key management layer can access cryptographic keys; other layers can only affect the key management strategy or gather key metadata, but not the key value.

At the key management layer, there is a functional element known as **Key Manager (KM)**. It is also known as a **Key Management System (KMS)** in the literature. The ITU-T recommendation [25] divides the functional requirements of KM into two separate agents: **Key Management Agent (KMA)** and **Key Supply Agent (KSA)**, as illustrated in Figure 4. Owing to their distinct key management tasks, this separation is defined for practical reasons. They can be installed on different machines within the same secure environment. The KMA includes the following functions: secure key storage, global key distribution, and key lifecycle management, whereas the KSA includes a key supply function. The KSA may optionally integrate a key combination function that combines QKD-derived key material with keys obtained through alternative exchange methods, including post-quantum cryptography, as needed [41].

2.1 Secure Key Storage

In contrast to traditional cryptographic methods for secret key exchange, which are performed relatively on demand, QKD requires a significant time window and is thus performed in advance regardless of demand on cryptographic keys. This is a well-accepted practice of decoupling key generation and key consumption processes, achieved by introducing secure key storage. The key storage regularly receives fresh cryptographic keys. This event is managed by a quantum layer, which should ensure reliable and uninterrupted operation that delivers keys at a steady

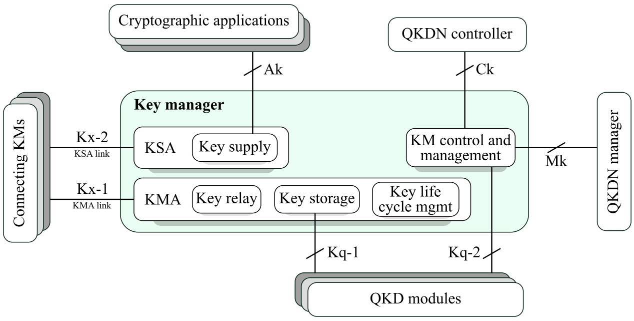


Fig. 4. The functional architecture of the key manager according to the ITU-T Y.3802 recommendation. The following reference points can be identified. **Kq-1** supplies QKD-generated keys from the QKD module to the KM. **Kq-2** is a control communication interface between the KM and the QKD module. **Kx-1** facilitates authenticated information exchange necessary for key relay and synchronization. **Kx-2** handles authenticated information exchange and synchronization of key supply. **Mk** manages the exchange of information between the key management layer and the network management layer. **Ck** is a control interface between the controller and the KM. **Ak** allows key requests and the delivery of keys from the KM to the service layer [41]. Throughout the rest of the article, the reference points are not labeled using the specific notation provided here. However, they can still be clearly identified based on their defined functions and roles.

pace. The lengths of cryptographic keys produced at the quantum layer by the same or different QKD devices will vary. It is recommended that KM reformat keys to a specific unit length [25]. This should be done before storage and requires interaction between involved KMs, as discussed in Section 2.2.

Keys are consumed from the key storage at variable rates. The consumption process is driven by the number of cryptographic applications and their encryption preferences. The way applications access keys is defined by the ETSI key delivery interfaces, ETSI QKD 014 [42] and ETSI QKD 004 [43], which are described in Section 2.4. To meet application requirements, the KM may modify key entries (key splitting or merging) and assign unique key identifiers on supply. As a result, interaction between the KMs involved is required.

Decoupling key generation and key consumption processes addresses not only the large gap between two consecutive key generation events but also the QKD's limited key generation rates. Secure key storage allows the accumulation of larger amounts of cryptographic keys when there are low demands for consumption processes. A burst of high-consumption demands that exceed the key generation rate can then be accommodated using existing key supplies. It is currently unclear how the age of keys affects overall security levels. Since the generated keys are ITS, their value cannot be compromised without direct access to the key storage itself. Consequently, only a security incident involving the key storage would necessitate the deletion of specific keys. During idle periods, when key storage recovery takes place—meaning there is no active demand for keys—it is recommended to replace older key entries with newer ones to maintain optimal key freshness.

2.2 Key Synchronization

A QKD network as a service generates, manages, and supplies symmetric cryptographic keys. Thus, it is essential to maintain synchronization, or consistency, among the contents of key storages. Otherwise, the service is not operational. Even if the perfect correlation between two symmetric keys is proven at the quantum layer, the key management layer must verify that both KM peers receive these keys without errors. This is accomplished by exchanging message authentication codes, hash values calculated on key bits and identifiers. However, the security details of this verification are said to be outside of ITU-T recommendations [25].

As it is anticipated that keys produced by different vendors will vary in size, it is recommended that the KM resizes keys to a specific unit of length before storing them. To resize keys, KMs must agree on a unit of length (which is typically set in advance) and unique identifiers for newly produced keys.[1] Given the distinct purpose of QKD networks, which involves the production, management, and supply of ITS keys, the security measures applied for synchronization purposes between remote KMs should be implemented with equally high levels of security. Synchronization messages can be frequent, leading to high internal consumption of keys at the key management layer.

The high frequency of synchronization messages is also a result of the creation and delivery of keys to the service layer. Cryptographic applications request keys with varying requirements, as discussed in Section 2.4. Typically, one or more keys from the key storage are used to create the supply key. To accomplish this, key splitting or merging is used. The supply key is then given a unique identifier and supplied in response to the request. This modification of key entries and creation of supply key must be propagated to the peer KM. Similarly, the peer KM creates the supply key and waits until the respective cryptographic application pair requests it.

It was previously stated that keys ought to be stored in a specific unit of length, but that length was not specified. It can be predetermined or, as recommended by ITU-T [44], a machine learning mechanism can be used to learn about the application requirements and dynamically choose this specific length. The latter simplifies key supply because key transformation of available keys in storage is no longer required. Nevertheless, assigning a distinct key identifier and synchronizing it at the key management layer is still necessary. This signalization can help detect malicious requests [45, 46] quickly.

2.3 Global Key Distribution

Given the uniqueness of the QKD process, which results in point-to-point connectivity with a limited distance, global key distribution allows peers to establish keys even if a QKD link does not directly connect them. One common method of global key distribution, hop-by-hop key distribution, was covered in the introductory Section 1. Distributing a random or local key in hop-by-hop fashion poses a security risk because global keys are directly accessible to nodes along the distribution path. As a result, apart from these intuitive approaches, the ITU-T [25] recommends two other global key distribution options: distribution with XORs uniformly processed at the destination node and distribution with XORs collected at a single centralized node. A recent study proposes a modified centralized approach that relaxes the requirements of secure key storage at the intermediate nodes [47]. Acknowledging the inherent single point of failure in centralized approaches, a distributed scheme has been introduced [48]. This distributed scheme maintains relaxed trust requirements while mitigating the risks associated with a centralized approach.

The global key distribution process, like the QKD process, takes time. For example, hop-by-hop key distribution involves multiple encryptions and decryptions that are both computationally consuming and time-consuming. It also includes propagation and processing time. For a path with more nodes, global key distribution can result in significant supply delays. Thus, the global key distribution process and key consumption process are separated. This is analogous to separating key generation and key consumption processes, discussed in Section 2.1. A node may distribute enough global keys (for a given destination) in advance to meet demands.

[1] The quantum layer generates large blocks (in the order of Mbits) of truly random bits. As a result, the resize operation will generally split this large key block into smaller, easier-to-manage blocks.

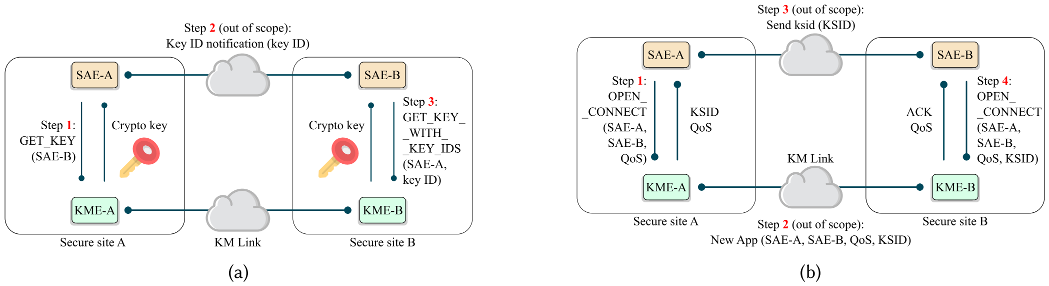


Fig. 5. (a) Use case of the ETSI QKD 014 key supply interface [42]. (b) Use case of the ETSI QKD 004 key supply interface [43]. Only key stream session establishment is shown.

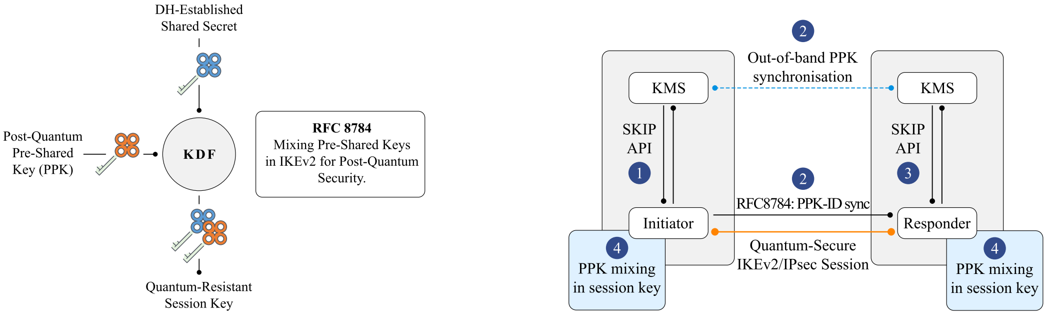
2.4 Key Supply

The key-supply interface describes access to the QKD network services by defining communication between cryptographic applications and KMs. ETSI has recently standardized two such interfaces: ETSI GS QKD 014 [42] and ETSI GS QKD 004 [43]. Through the ETSI specification, cryptographic applications are also called **Secure Application Entities (SAEs)**, whereas KMs are referred to as **Key Management Entities (KMEs)**.

2.4.1 ETSI GS QKD 014. The ETSI QKD 014 specification defines an interface based on the **HTTPS (Hypertext Transfer Protocol Secure)** protocol and the **JSON (JavaScript Object Notation)**-encoded data format of posted parameters and responses. Because of its **REST (REpresentational State Transfer)**-based nature and simplified processing logic required at KMEs, this interface is well accepted among vendors who supply quantum equipment. Three fundamental methods are used to depict the communication between SAEs and KMEs: GET_STATUS, GET_KEY, and GET_KEY_WITH_KEY_IDS. Figure 5(a) depicts the use case of the ETSI QKD 014 key-supply interface. The GET_STATUS method allows one SAE, let us call it SAE-A, to collect status information on QKD connection (which may be direct or virtual, established through trusted relays) to a specific destination SAE-B. SAE-A obtains one or more keys from the KME through the GET_KEY method. The request expresses the SAE's requirements by including the number and size of keys requested, additional destination SAEs, and other specific parameters. Provisioned keys are assigned unique key IDs that are synchronized between KMEs. After receiving key IDs through arbitrary means,[2] SAE-B uses the GET_KEY_WITH_KEY_IDS method to obtain the same set of keys.

2.4.2 ETSI GS QKD 004. The ETSI GS QKD 004 introduces the concept of sessions with **Quality of Service (QoS)** capabilities for SAEs without requiring a specific protocol. Figure 5(b) depicts the use case of the ETSI QKD 004 key-supply interface. Three primitive functions are defined: OPEN_CONNECT, GET_KEY, and CLOSE. The OPEN_CONNECT function establishes a key stream session with the expected level of service for SAE. Establishing a key stream session requires the KM to rendezvous with the designated destination KM. In the case of virtual QKD connections, the responsibility of the KM is also to discover the destination KM and QoS available in the path of multiple KMs. This, however, is outside the scope of ETSI QKD 004. Once the key stream session is established or permitted, the calling SAE, for example, SAE-A, is granted a **Key Stream IDentifier (KSID)** used in subsequent key requests. To maintain the promised level of service, the KM is responsible for managing and reserving keys for active key stream sessions.

[2] The ETSI documentation does not describe how connecting SAEs communicate key IDs.



(a) (b)

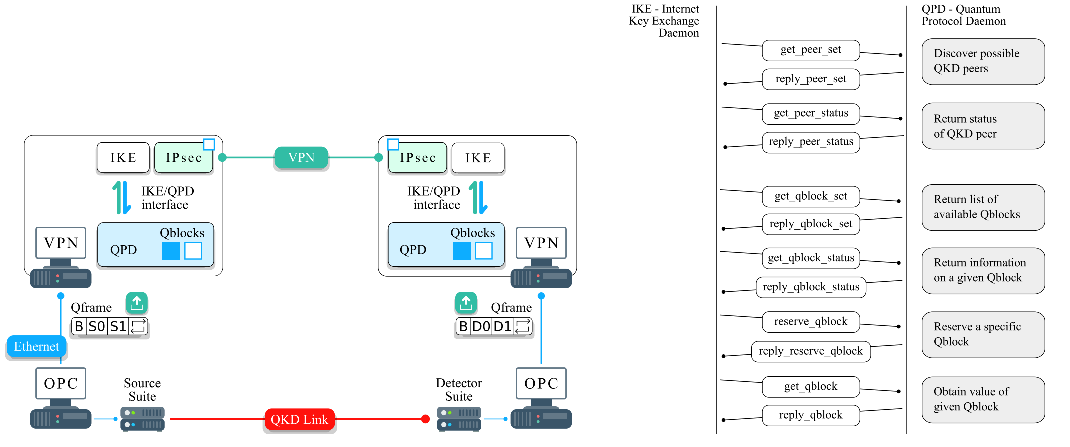
Fig. 6. (a) The IKEv2 key derivation function, a mixture of the traditional key and the PPK. (b) Quantum-Safe IKEv2 and IPsec Session Keys with a dynamic PPK as proposed by Cisco.

The QoS parameters that can be agreed upon are key size, maximum and minimum (requesting) key rate in **bits per second (bps)**, jitter of key delivery, and priority level. The GET_KEY function returns the required amount of key material requested for a specific KSID whereas the CLOSE function allows SAEs to end and terminate key stream sessions.

Another interesting notion discussed in ETSI QKD 004 is the organization of the key management layer. Each QKD module has its own key management unit, and a higher-level Key Server communicates with multiple QKD modules within the QKD node. It is emphasized that the ETSI QKD 004 interface is suitable for communication between KMs at various hierarchical levels. With multiple vendors pushing their proprietary KMs, this organization is probably the most likely to be implemented. Interoperability will also be attained through a new interface, ETSI QKD 020 [49], which is currently in draft and describes horizontal communication between two KMs within the same trusted node, allowing one KM to pass the key to the other to achieve relay through this node.

2.4.3 Cisco SKIP Protocol. Apart from the key-supply interfaces defined by the ETSI that were previously discussed, it is noteworthy to mention that certain commercial solutions are also available, such as the Cisco **Secure Key Integration Protocol (SKIP)**. The Cisco IOS-XE relies on the enhanced **Internet Key Exchange version 2 (IKEv2)** protocol (RFC 8784 [50]), which uses a mixture of traditional **Elliptic Curve Diffie-Hellman (ECDH)** cryptographic keys and **Postquantum Preshared Keys (PPK)** in the key derivation function. Figure 6(a) depicts the process of creating session keys. This feature enables quantum-safe encryption using PPKs and can be applied to all IKEv2 and IPsec VPNs, such as FlexVPN (SVTI-DVTI) and DMVPN [51]. PPKs are ingested into the router from external sources using the SKIP protocol.

The SKIP protocol operates as a restful **Application Programming Interface (API)** based on the HTTPS protocol and employs **Transport Layer Security 1.2 (TLS1.2)** with a PSK-DHE cipher suite to ensure secure communication between the KMS and the router [51]. Figure 6(b) depicts the use case of the Cisco SKIP protocol. The configuration of the SKIP clients running on both the IKEv2 initiator and responder includes the IP address and port number of the key source and the preshared key for the TLS1.2 session outlined in the RFC 5246 [52]. The PPK sources are set up with the SKIP parameters, comprising the local key source identity and the list of peer key source identities. To be SKIP compliant, an external key source must implement the protocol and use an out-of-band synchronization mechanism to deliver the same PPK between the encryption devices to both the initiator and the responder. The external key source could be a QKD device, the KMS, some software, or a cloud-based key source or service.



(a) (b)

Fig. 7. (a) The DARPA quantum network structure. Secret key material is kept in fixed-sized blocks — QBlocks within QPD. QBlocks can be reserved and supplied to the IKE daemon through the IKE/QPD interface. The IKE protocol has been modified to include obtained QKD key material in the key derivation function of phase 2. (b) DARPA's key supply interface, named *IKE/QPD interface*, allows IKE protocol, as a client application, to reserve and obtain cryptographic keys from the QKD platform [54].

3 Key Management Solutions

In this section, existing key management solutions are described in chronological order. Some of the solutions examined are more comprehensive than others, which only address one aspect of the KM functionality. To the best of our knowledge, these are all publicly reported solutions related to the key management layer functionality. Although the section contains some work about the control layer, it focuses on the problems of the key management layer with the help of the central device. Thus, it is also appropriate to include and analyze them.

3.1 DARPA Quantum Network Key Management

Since the DARPA quantum network was the first built QKD network, the responsible project team was the first to encounter difficulties in increasing the level of practical applicability of quantum technologies in modern communication systems [13, 53, 54]. As shown in Figure 7(a), a QKD link connects two QKD endpoints, each comprising an **optical process control (OPC)** computer and a VPN computer. The OPC computer oversees the optical and electronic components of the source or detector suite, facilitating the quantum transmission. The outcome, i.e., the raw key, is then forwarded to the VPN computer in a continuous series of frames known as *Qframes*.

A **quantum protocol daemon (QPD)** running on the VPN computer distills the secret keys from the raw Qframes and stores them in memory as fixed-sized blocks known as *Qblocks*. The Qblocks can be reserved or obtained by cryptographic applications using the IKE/QPD interface, depicted in Figure 7(b), where they are deployed within a key derivation function in phase 2 of the IKE protocol. Keys are always served in Qblock units of fixed size. The served Qblock is removed from QPD's memory. While limited in functionality, this is the first practical application of QKD-derived key material, demonstrating the need to store key material and propose an interface to access QKD network services. The general concepts behind the global key distribution process are outlined without delving into greater details, but the notion of managing global keys is lacking.

3.2 SECOQC QKD Network Key Management

The SECOQC project aimed to develop a global network for SEcure COmmunication based on Quantum Cryptography, which has resulted in the first European QKD network [55–59]. For the first time, it is made clear that the sole purpose of the QKD network is to generate, manage, and distribute ITS keys. This network is distinct from traditional telecommunication networks, with a completely new protocol stack running through all layers. It operates on dedicated network infrastructure or as an overlay network on conventional networks. The protocol stack is inspired by the traditional OSI[3] model but with only minor adjustments on each layer.

SECOQC's QKD network consists of **Quantum Back-Bone (QBB)** nodes and QBB links. The QBB node structure is depicted in Figure 8(a). The **Quantum Point-to-Point Protocol (Q3P)**, an extension of the traditional **Point-to-Point Protocol (PPP)**, enables two points to communicate using ITS perks. This is accomplished by utilizing ITS local keys, which are stored and managed within this module. It functions as a communication interface that applies various security profiles to ongoing traffic.

When the key is generated, it is pushed from the QKD device to the Q3P module. A communication interface between QKD devices and Q3P modules must be defined to achieve interoperability. However, the interface specification is not provided or discussed further. Delivered keys are gathered in *pickup* stores at the Q3P module that are specific to each QKD device (see Figure 8(b)). Each key has metadata assigned to it, the most important of which is an identifier — KeyID. Before periodically moving keys to the permanent and secure *common* store, the Q3P module must ensure that the same keys are present on the peer side. A STORE sub-protocol has been proposed for this purpose. It is carried out in three stages, as shown in Figure 8(c). A pair of linked Q3P modules operate in a master/slave paradigm, with the master Q3P initiating the STORE sub-protocol.

To use key material from the *common* store, the Q3P module must define key buffers, which are *in–out buffers* filled with keys from the *common* store and have a defined purpose. Keys from the *out buffers* are only used for outbound traffic to apply security services, and keys from the *in buffers* are only used for inbound traffic to process data. Both the master and the slave monitor the state of the *in buffers* and can request that new keys be moved from the *common* store, but only the master can decide and propose which keys should be moved. A LOAD sub-protocol has been proposed for this purpose. It is carried out in three stages, as shown in Figure 8(d). Because the LOAD sub-protocol is only triggered by receiver decisions, i.e., the state of *in buffers*, Q3P modules can control the transmission rate.[4]

The cryptographic applications establish a TCP connection with the QBB node, expecting services from the QKD network. As a result, the QBB node establishes a **QKD Transport Layer (QKDTL)** connection with the peer QBB node over the QKD network. The QKDTL protocol adapts the TCP protocol for the QKD network. The QKDTL handshake includes an expected key rate, and it travels hop by hop towards the destination. The SYN or SYN-ACK packet is dropped if an intermediate node cannot meet the defined key relaying throughput.[5] After establishing a QKDTL connection, a random key is generated at the desired rate and relayed through the QKD network. Unlike the traditional TCP protocol, the QKDTL protocol does not support resending. This is why congestion control is implemented differently, specifically to respond proactively rather than

[3] Open Systems Interconnection.

[4] The transmission rate in this context refers to the rate at which keys can be relayed between two nodes. The receiver QKD module can regulate the transmission rate of its peer QKD module by appropriately adjusting the activation of the LOAD sub-protocol.

[5] QKDTL communication is bidirectional, with both the sender and the receiver announcing desired key rates. As a result, the key resources for two different flows are probed by SYN and SYN-ACK packets.

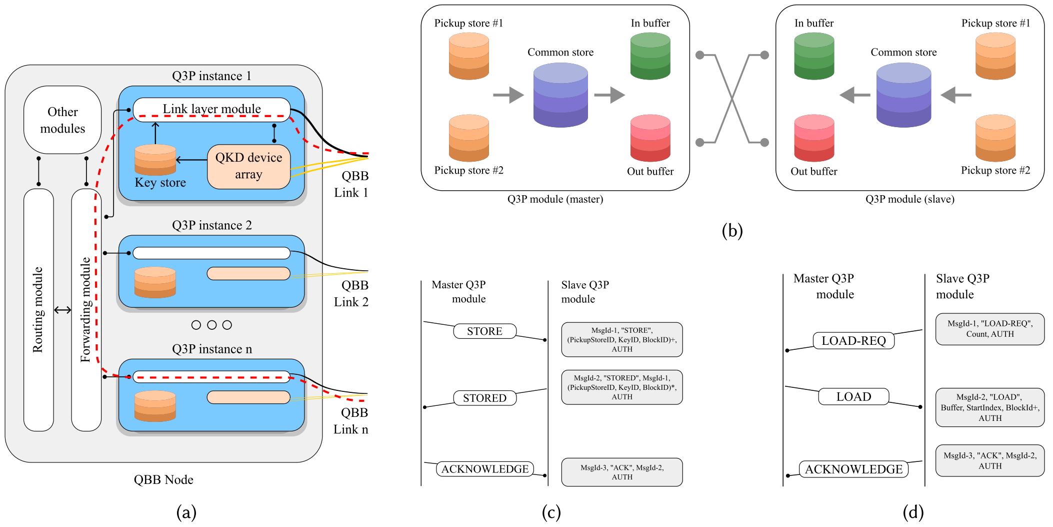


Fig. 8. (a) The structure of the QBB node. The QBB node has an equal number of Q3P instances as the number of QBB links. The QBB links are logical links comprising one or more quantum channels (*yellow solid lines*) and a classical channel *(black solid line*). The *dashed red line* shows the traffic flow (e.g., key relay) through the QBB node. The protected packets are first processed (e.g., authenticated and decrypted) in the ingress Q3P instance. The packets are inspected for the forwarding decision and handed to the appropriate egress Q3P instance. The egress Q3P instance applies desired security profiles to the ongoing packets and forwards them over a classical IP network toward the peer QBB node. (b) Key stores within SECOQC Q3P module. *Pickup* stores are unique to each QKD device. When the STORE sub-protocol is completed, keys from the *pickup* stores are moved to the *common* store. After completing the LOAD sub-protocol, keys from the *common* store are moved to *in/out buffers*. The master Q3P module dictates the execution of sub-protocols. The figure illustrates the relationship between *in/out buffers* between connecting Q3P modules. The *in buffer* of the master Q3P module is in sync with the *out buffer* of the slave Q3P module and vice versa. (c) A STORE sub-protocol. It is initiated periodically by the master Q3P to transfer keys accumulated in *pickup* stores to the *common* store. Each message is authenticated. (d) A LOAD sub-protocol. It is initiated by the master with the “LOAD” message or by the slave Q3P with the “LOAD-REQUEST” message. It is used to transfer keys from the *common* store to a target *in/out buffer* in a synchronized manner. Each message is authenticated [59].

reactively. If the intermediate node notices that the supply of keys is running low, it will set the CON flag within the QKDTL packet. When such a packet arrives at the destination, the destination will prolong sending the ACK, resulting in a timeout on the sender side and thus halving the congestion window. This reduces the transmission rate of the sender. Global keys do not need special treatment at the key management layer because they are immediately supplied to cryptographic applications.

Scheduling and load balancing are two additional concepts that can be considered within the key management domain. The forwarding module monitors and balances the consumption load across multiple Q3P modules within the node. It is supported by the routing module, which provides multiple paths in ascending order of weights equal to the number of Q3P modules. If the load on the Q3P module corresponding to the shortest path exceeds some threshold value, the Q3P module on the ascending shortest path is chosen. Then, the packets are queued for scheduling once the forwarding decision is made and the Q3P module is chosen.

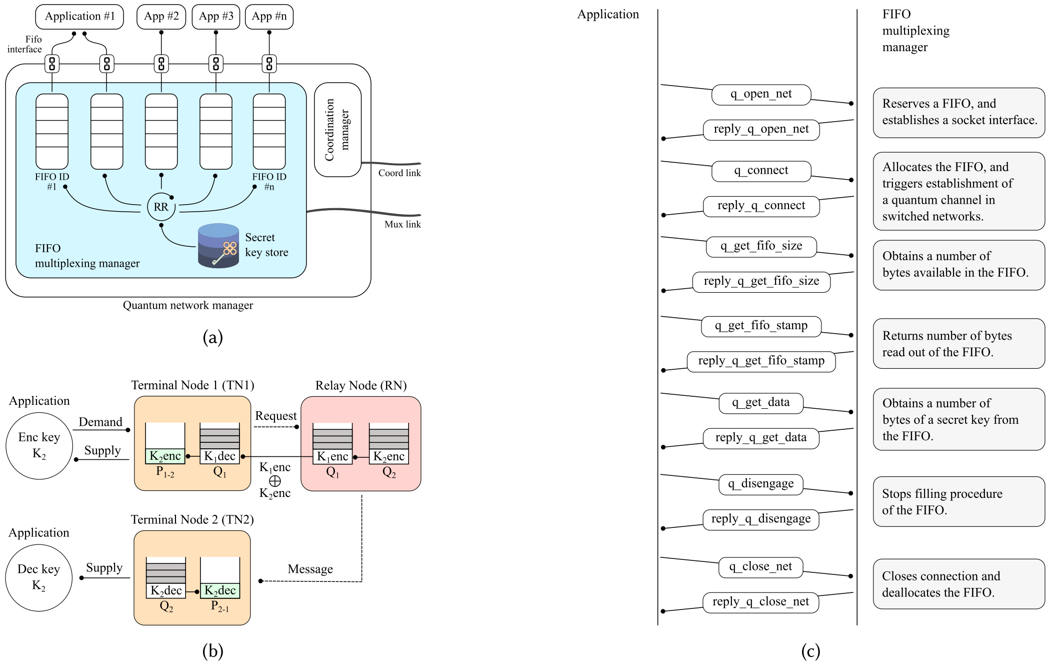


Fig. 9. (a) The structure of the NIST quantum network manager. The coordination manager is in charge of switching, polarization recovery and compensation, and channel timing alignment. The FIFO multiplexing manager creates and maintains an independent key stream for each connected application. It spawns the FIFO interface for each FIFO queue so that the application can access the stream of bits [60]. (b) The NEC on-demand key management. On request from TN1, RN encrypts key K_2 using key K_1 and sends it to TN1. Simultaneously, RN requests that node TN2 set key K_2 as a global decryption key shared with TN1. TN1 decrypts key K_2 using local key K_1 and sets it as a global encryption key shared with TN2. In this manner, TN1 and TN2 share symmetric key K_2 defined for different purposes and can supply requesting applications [61]. (c) Application programmable interface functions to define communication between applications and NIST's FIFO multiplexing manager [60].

3.3 NIST Quantum Network Manager

The structure of a quantum network manager proposed in a 2008 conference paper [60] is depicted in Figure 9(a). The coordination manager is responsible for control tasks and is not further discussed. The **first in first out (FIFO)** multiplexing manager creates an independent FIFO queue for a stream of synchronized bits for each application connection. A single application may open multiple threads; for example, a pair of applications may open two pairs of FIFOs for bidirectional traffic flow. Using the round-robin algorithm, the multiplexing manager fills the queues with keys from the QKD secret key store. The amount of keys assigned to each queue is set at the start of each pass. In this manner, the multiplexing manager serves multiple applications (e.g., IPsec, TLS) that may have different requirements (e.g., OTP, AES).

The authors of [60] emphasize keeping secret key stores and queues in sync across sites. A few corrupted bits do not have catastrophic consequences because a single corrupted key can be dropped and a new one used instead. On the contrary, a few dropped bits result in the loss of synchronization and the inability to use any further keys in the store or queue. This is due to the manner in which key material is stored. Each byte of generated key material is assigned a unique sequential value, a stamped ID, at the privacy amplification layer. The key material and the stamp ID of the first byte are then transferred to the secure key store of a FIFO multiplexing manager. The multiplexing manager can predict the stamp ID of incoming material based on the most recently received entries. If it deviates from the expected value, the multiplexing manager is restarted, or it looks for the last synchronization point to preserve key material. The keys are stored in the same homogeneous manner within the FIFO queue, where each byte is now assigned a new sequential stamp ID for each queue independently. Figure 9(c) depicts an interface that defines communication between the application and the multiplexing manager.

3.4 QCC Security Processor Key Manager

Within the QCC security processor, a session-based key buffer approach was proposed in 2008 to supplement the IKE/IPsec framework with QKD keys [62, 63]. The key manager is in charge of distributing generated key material into key buffers established for each application. The extended IKE protocol, i.e., the client application, negotiates key rates and key buffer capacities. The client first registers with a key manager by submitting its application ID and the ID of a peer application. The client then sends a connection request specifying the rate and size of the key buffer. Key managers operate on a master/slave basis, with the master having the lower IP address.

A refill procedure is initiated when the amount of key material in one of the buffers falls below a certain threshold. The threshold value can be calculated using Equation (1), where len_{max} is the maximum size of the key requested, $size$ is the buffer capacity, and $global_{thr}$ is some globally set threshold variable:

$$buff_{threshold} = max(len_{max}, size \cdot global_{thr}). \tag{1}$$

Unassigned key material is distributed to key buffers with levels less than the threshold value during the refill procedure. The key material is distributed following application key rates. Additional key material, if any, is assigned to the remaining active key buffers. This action must be coordinated among KMs, but it is not described in detail. The communication between KMs is authenticated with QKD keys and is carried out using a binary TCP/IP protocol.

3.5 NEC Key Management

In 2009, authors from the NEC Corporation published a paper [61] on technologies for QKD networks integrated with optical communication networks and discussed key management. The NEC architecture of the QKD network consists of four layers: a key-generation layer, a connection layer, a key management layer, and a communication layer. The key-generation layer is a collection of point-to-point QKD links that generate local keys. The connection layer performs the global key distribution. The key management layer monitors and controls the generation of local keys and the distribution of global keys. This can be accomplished in two ways: on-demand key supply and fixed key allocation. Figure 9(b) depicts on-demand key management, in which keys are relayed in response to an application request. There are two types of nodes: **terminal nodes (TNs)** that serve applications and **relay nodes (RN)** that act as trusted relay nodes and distribute keys on behalf of others. Both nodes have QKD key pools (Q) where local keys are stored. Terminal nodes have logical key pools (P) where global keys are stored. Keys are stored in fixed-size key files with one of two extensions: *enc* and *dec*. Local keys are always stored with the *dec* extension within TNs and with the *enc* extension within RNs. Figure 9(b) depicts the global key distribution.

In the second key management technique, the fixed key allocation, the relay node creates keys in advance for all terminal nodes. This approach leads to cumbersome key management in networks with many terminal nodes, as monitoring and keeping all key pools active becomes increasingly difficult. Furthermore, supporting the addition and removal of terminal nodes from the network becomes more difficult. The fixed-key allocation key management facilitates key separation in terminals and relay nodes for encryption and decryption keys.

3.6 MagiQ Technologies Key Manager

In 2010, a patent on KMs for QKD networks was published [64]. We refer to the techniques proposed as **MagiQ Technologies (MT)** KMs as the company is an assignee to the patent. MT introduces the following layers, as shown in Figure 10(a): a QKD layer, a persistent storage layer, a KM layer, a key storage layer, and an application layer. Keys are generated within the QKD layer and assigned a unique identifier as a counter value before being pushed to the persistent storage layer. Keys are kept in persistent key storage S in chronological order. The node has as many persistent storages as the number of peers that are directly connected to it. The KM layer keeps the application registration record R, which contains a list of connected applications $A_1, A_2, \dots, A_n$, references to their dedicated key storages $K_1, K_2, \dots, K_n$, and key rates $r_1, r_2, \dots, r_n$. The keys are distributed from persistent storage to application key storages based on registered application key rates. Key managers must communicate to exchange information about which keys (based on key identifiers) are distributed to which K_i. This is not discussed in detail, but it is known that communication is carried out using the TCP/IP protocol. The registered application then gains access to key storages to obtain keys, which are then removed from the key storage. The QKD devices can also register with the ,KM to obtain the keying material required for authentication.

The KM may reconfigure a registration record because application key rates can fluctuate over time. The KM monitors application key rates and adapts the distribution function dynamically to address changes. This must be coordinated among connected KMs and necessitates communication using a variant of the two-phase commit protocol. In this case, one node must act as the coordinator or the master node. Key managers may support policies that mandate all key records older than a set timestamp be deleted from storage using the same two-phase commit protocol. The MT KM includes audit and recovery functions for detecting and recovering damaged storage.

3.7 SwissQuantum QKD Network Key Management

The SwissQuantum QKD network ran for more than 1.5 years (from the end of March 2009 to the beginning of January 2011) to prove the long-term reliability of quantum technology. The results were published in 2011 [65]. The network has a three-layered structure with a quantum, key management, and application layer. Although the focus is on the quantum layer, few details describing the key management layer have been revealed. At the key management layer, the key server collects keys from QKD devices, stores them, and distributes them to applications. It includes a key redundancy concept in which two different paths (direct and via trusted relay) are used to generate keys between two sites. The keys are stored in the buffers once combined with a key shared via a **Public Key Infrastructure (PKI)** using the OTP cipher. This concept, known as *dual-key agreement*, is intended to improve service robustness by supplying only PKI keys when the QKD service fails. Each connected application has a separate key buffer. The application access to the buffer, i.e., the key supply interface, is not discussed in any way. The key relay is accomplished by sending a random key over point-to-point OTP secure tunnels.

3.8 QoS-Supported Key Manager

A service model and a supportive QoS-supported scheme were proposed for QKD in 2011 [66]. Three service classes are presented in [66]: key-guaranteed service, key-prioritized service, and key-best-effort service. The primary performance metric used to differentiate classes is Distribution Time. The Distribution Time is the total processing time required for a global key to travel from a source node to a destination node. Based on this, the key-guaranteed service

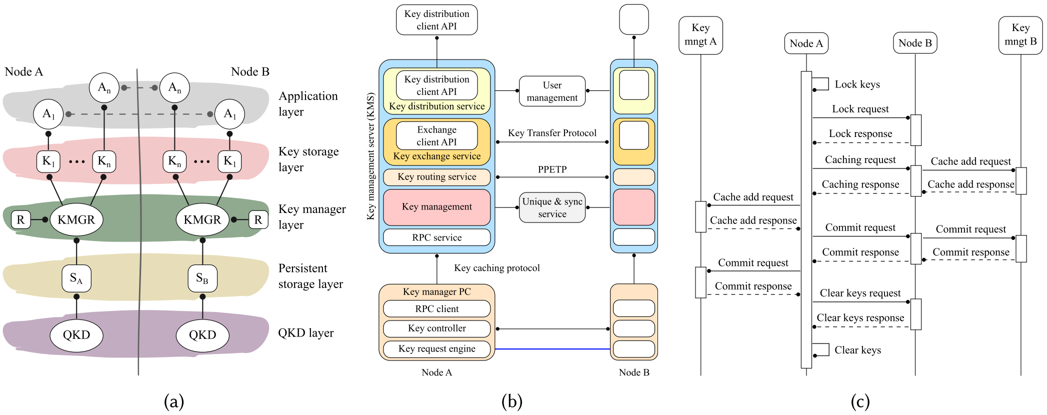


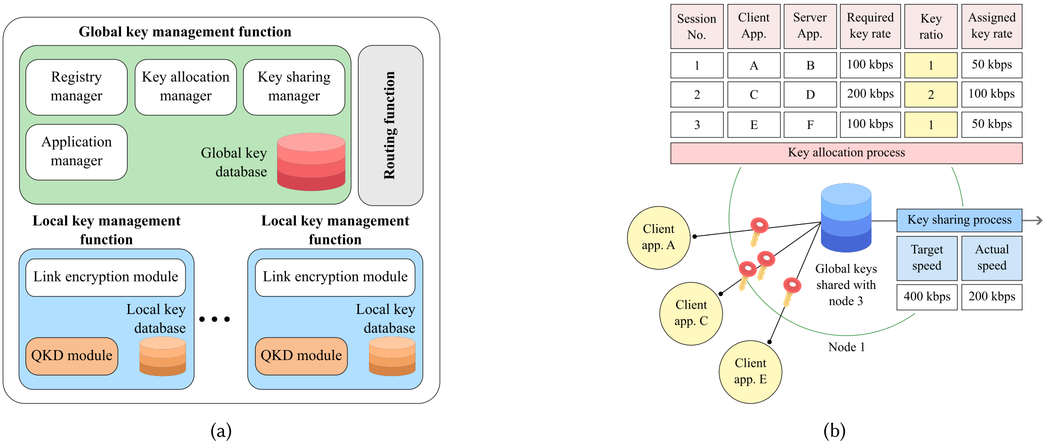
Fig. 10. (a) MagiQ Technologies layered structure [64]. (b) Structure, functional elements, and protocol of the NECTEC key management approach. (c) The NECTEC Key Caching Protocol defines key delivery from KMPC, where generated keys accumulate, to the KMS. Before keys are delivered to the KMS, they are verified [67].

refers to applications with the greatest demand for Distribution Time and the greatest right to occupy keys. The key-prioritized service refers to applications with flexible requirements on delay, while the key-best-effort service refers to delay-insensitive applications. A **Quantum Key Reservation Approach (QKRA)** scheme is proposed to support the key-guaranteed service class. It reserves key resources on the intermediate nodes on a relay path towards the destination. The **hop-by-hop queue approach (HHQA)** handles key prioritizing and key best-effort services. These two service classes are assigned to separate queues, with the queue serving key-prioritizing traffic receiving priority and being served first. The simulation results prove the advantage of the scheme when compared with the QKD network without QoS support. The protocols and details of implementation are not discussed further.

3.9 NECTEC Key Management

An efficient key management method has been proposed for the Thailand quantum network in studies presented in [67] and [68] from 2012 and 2015, respectively. We refer to this strategy as a NECTEC key management approach because the **National Electronics and Computer Technology Center (NECTEC)** provided the funding. The network structure consists of three layers: the quantum layer, the key management layer, and the application layer. Figure 10(b) depicts elements and custom protocols at the key management layer. Established keys are first accumulated within the local **Key Manager PC (KMPC)**. They are verified between connecting KMPCs before being handed over to the **Key Management Server (KMS)**. Key Caching Protocol handles key verification and delivery to the KMS. Figure 10(c) depicts its sequence diagram.

The KMS uses the Key Transfer Protocol and the **Point-to-Point Encrypted Transfer Protocol (PPETP)** to distribute global keys. In addition to the PPETP, a Key Routing protocol is defined, which determines the key distribution paths. A randomly generated key is encrypted with a local key shared with the next hop and sent directly to the destination. Simultaneously, the local key is relayed to the destination hop-by-hop using PPETP. A destination acquires the random key by receiving both messages. By demultiplexing a sequence of ordered secure bits into separate buffers for each application, the KMS supports multiple applications. Two buffers are distinguished depending on their purpose: an In-Buffer and an Out-Buffer. A Key Distribution Protocol is defined to supply keys to the cryptographic application but without specific details.



Session No.	Client App.	Server App.	Required key rate	Key ratio	Assigned key rate
1	A	B	100 kbps	1	50 kbps
2	C	D	200 kbps	2	100 kbps
3	E	F	100 kbps	1	50 kbps

Target speed	Actual speed
400 kbps	200 kbps

(a)

(b)

Fig. 11. (a) Structure of the Toshiba QKD service node. The local key management function corresponds to the SECOQC Q3P module. It manages local keys and acts as an interface for service node traffic to other nodes. The routing function selects the suitable local key management function (Q3P module) to forward keys to the next hop node. The global key management function manages global keys, including generation, distribution, and supply. This function is the main contribution of the Toshiba key management solution to the existing SECOQC approach. (b) Key allocation process within Toshiba key management. The required rate for global key sharing (400 kbps) is the sum of the client application's required key rates (100, 200, and 100 kbps). However, the achievable global key sharing is lower (200 kbps) due to the limited generation rates of QKD links in a relay path. During the allocation process, the key ratio for connected clients is calculated and key rates are assigned accordingly. The sum of key rates assigned does not exceed the global key sharing speed [69].

3.10 Toshiba Key Management

The authors from Toshiba Corporation (at the time) presented a research paper [69] in 2016 to encourage widespread use of QKD, hence, the name Toshiba key management, focusing on two barriers: applicability and cost, which are usually overlooked as attention is focused elsewhere (most commonly on improving the design and implementation of QKD protocols to overcome the rate and distance limitations of technology). The presented approach was emulated, and the results show that a single QKD network can host multiple applications concurrently, fairly, and effectively. The network architecture is based on the SECOQC QKD network described in Section 3.2 and is illustrated in Figure 11(a). The local key management function corresponds to a Q3P module in the SECOQC design, and Toshiba introduces global key management, which was partially missing in the SECOQC approach. The network architecture is enhanced with the following functions: an application directory, a key sharing and allocation mechanism, and a cryptography communication API.

The application directory function enables QKD service nodes to translate application IP addresses to the service nodes they connect.[6] This function is implemented within the registry manager of a global key management function. An application directory record is created and stored on a server running on a specific service node. The remaining service nodes within the QKD network contain clients that submit and request translations from that server.

[6] Typically, the application key request is submitted to the local QKD node by specifying the network target application. As a result, for key relay, for example, the source node must be aware of the destination node in the entire network that serves the target application.

The key sharing and allocation mechanism allows for a fair and effective supply of keys to multiple applications. As a result, two functional elements are implemented within the global key management function: a key-sharing manager and a key-allocation manager. The key-sharing manager generates random bit sequences — global keys, which are then relayed to the proper destination node. The rate at which global keys are generated and distributed is determined by the sum of application demands for the same destination node. To avoid congestion, the global key generation rate must be adjusted to match the actual throughput of a chain of QKD links on a relay path (see Section 3.2). As a result, multiple applications are competing for access to global keys. The key-allocation manager calculates the required global key ratio for each application. An example of a key allocation rule for three competing applications is depicted in Figure 11(b).

To extend this work, a high-speed key management method was proposed in 2019 [70]. To support high-speed global key distribution, an emphasis is placed on the OTP encryptor within the local key management functions. The encryptor is in the spotlight because it imposes additional computational and time costs, such as encryption, decryption, local key reading, and key removal. To improve overall throughput, the authors proposed a key removal strategy. In contrast to the conventional approach, which removes each key as soon as it is used, the proposed strategy does not remove keys until a certain number of keys are used. Keys are then removed in larger units —the results of the evaluation show that the system is adequate for the assumed high-speed QKD system. In addition, it is revealed that the local key management function uses the **Secure Shell (SSH)** protocol to read keys from QKD devices, and a cryptography communication API incorporates a REST-based API that uses the HTTPS protocol. The REST-based API defines communication between applications and the global key management function and performs the following functions: providing key status, encryption key provisioning, and decryption key provisioning. It is thus equivalent to the well-established ETSI QKD 014 interface.

3.11 NICT QKD Platform

A project funded by the **National Institute of Information and Communication Technology (NICT)** resulted, in 2017, in a QKD platform (QKDPF) that supports multiple applications [71]. Given that this article builds on the previous research from 2011 [72], which describes the realization of the Tokyo QKD network, it is logical to begin there. Although this former paper should have been considered in the previous sections (to keep the approaches in chronological order), its cursory descriptions of the key management layer have resulted in its inclusion here with the introduction of QKDPF. The Tokyo QKD network, reported in 2011 [72], consists of three layers: a quantum layer, a key management layer, and an application layer. The key management layer hosts KMAs. They are responsible for collecting keys from QKD devices, reshaping keys, assigning identifiers, and storing keys in numerical order for encryption and decryption. Once again, the interface between QKD devices from various vendors and key management is acknowledged but not discussed in detail. It is unclear how KMAs store, manage, and use keys, but it is stated that user data is given to the KMA for encryption. This goes against the now-well-established definition and purpose of QKD networks: to generate and distribute ITS cryptographic keys rather than securely transfer user data. Furthermore, the term *Key Management Server* refers to a centralized entity that assists KMA by performing key lifecycle management and relay path provisioning.

The QKDPF, proposed in 2017 [71], extended the layered structure of the Tokyo QKD network with a key supply layer, thereby aligning it with the definition and general purpose of QKD networks. This newly introduced layer sits between the key management and application layers, as shown in Figure 12(a). It hosts KSAs, enabling the secure supply of independent cryptographic keys to multiple applications. Unlike the original Tokyo QKD network design, the KMAs now transmit random data (cryptographic keys) hop by hop instead of user data. Figure 12(b) depicts

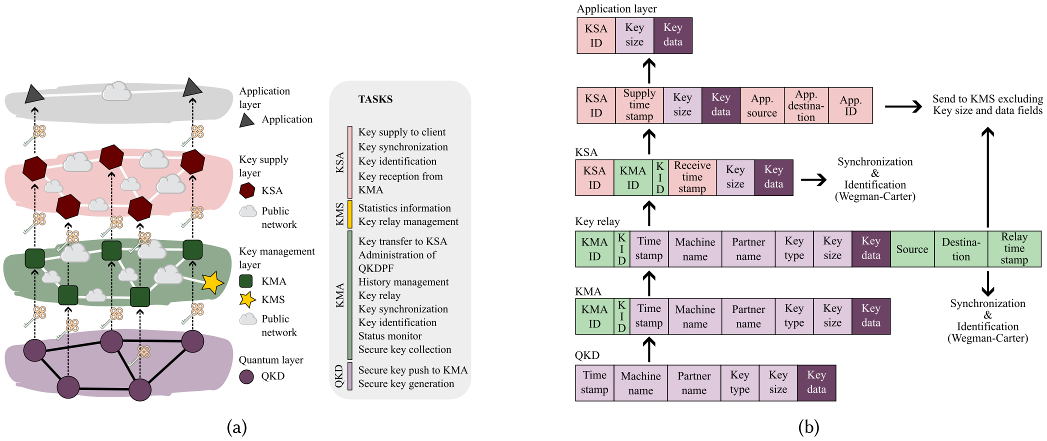


Fig. 12. (a) NICT QKD Platform structure and functional requirements of each layer. (b) NICT's key lifecycle. The key format is explained as follows: *timestamp*, timestamp of key generation; *machine name*, ID of QKD equipment at local site; *partner name*, ID of QKD equipment at opposite site; *key type*, identifier of encoding key or decoding key; *key size*, key size; *key data*, key data; *KMA ID*, ID of KMA; *KID*, ID of received quantum key; *relay timestamp* –, timestamp of key relay; *source*, relay source; *destination*, relay destination; *KSA ID*, ID of KSA; *receive timestamp*, timestamp of key reception; *supply timestamp*, timestamp of key supply; *app. source*, source ID application; *app. destination*, destination ID application; *application ID*, application ID [71].

the key lifecycle and general information associated with keys at various layers. The ITU-T Y series recommendations now advocate for nearly identical structure, elements, functions, and key formats to one proposed within QKDPF.

3.12 Quantum Canada Key Management

A project funded by Quantum Canada resulted in 2018 in general guidelines for designing a QKD network structure suitable for deployment in enterprise environments [73]. The network structure comprises four layers: the QKD link layer, the network layer, the key management layer, and the application layer. QKD is performed on individual point-to-point links at the QKD link layer, generating keys between neighboring nodes. The keys are then passed to the network layer, which manages these local keys and allows key distribution between arbitrary network nodes. Key distribution occurs at the request of the KMS layer, which estimates the load based on end-user demand for keys.

In terms of key management, Quantum Canada's approach is broadly similar to Toshiba's approach (Section 3.10) and, thus, the SECOQC project (Section 3.2). The network layer manages local keys and performs key relaying. It is divided logically into two planes: control and data planes. Based on demands generated by KMS, which can be either continuous or one-time, the control plane manages global key generation between sites and determines the best relaying paths. The continuous mode requires a specific key generation rate with a remote site, whereas the one-time mode requires a particular number of keys generated with a remote site. The data plane carries out key relaying, which uses local keys from the temporary key pool management function. Because multiple relaying flows for distinct remote sites share key resources of a QKD link layer, a scheduling algorithm is implemented to ensure fairness. A deficit-weighted round-robin algorithm is used for continuous mode demand and a simple FIFO queuing mechanism is used for one-time demand. Local keys are classified into three types an, thus, assigned to a specific purpose. A portion of the local keys is reserved for distribution to hosts on directly connected sites (no relaying is required). Others are used for relaying purposes, of which there are two types. The first type includes local keys that will be transformed into global keys as a result of relaying between the local and remote sites. The second type includes local keys, used within the local site to relay keys on behalf of others.

Keys managed by the network layer are eventually passed to the KMS layer on demand. The KMS collects, stores, and maintains synchronization of keys. Multiple connected clients within a secure site share access to the key storage. The authors emphasize the issue of key access collisions, which result in key material waste. The key access collision occurs when a client application within site A is served with key K while the KMS in remote site B serves the same key K for a different purpose. Several solutions to this problem are proposed. One solution is for KMSs to serve keys from different ends of the database, such as the KMS at site A serving keys from the beginning and the KMS at remote site B serving keys from the end. This solution, however, is not recommended because it may result in practical inefficiency due to key pool fragmentation. Another way to solve key access collisions is to assign a small number of keys from the local key pool to a working set and have KMSs serve keys from that working set only in a previously described manner.

Client requests are monitored to collect and analyze the demand statistics, which are used to control global key generation at the network layer. If the available keys are deemed insufficient, a policy engine function may provide a fall-back method, such as producing keys based on a key derivation function. The policy engine function defines policy rules governing key use and includes client expectations (e.g., size and lifetime). Client expectations are classified into five categories based on security requirements, which help guide the key-generation process at the network layer.

3.13 NSFC SDQaaS Framework

This subsection summarizes the research conducted through several works based on the same concept but introducing various optimizations. The concept will be referred to as the NSFC solution because the **National Natural Science Foundation of China (NSFC)** provided ongoing support. First, an SDQaaS framework is introduced in 2019 [74] (skipping the preliminary work from 2017 [75]). **SDQaaS** stands for **SDN for QKD as a service**. The **QKD as a service (QaaS)** concept shares the QKD network infrastructure among multiple clients. Furthermore, in SDQaaS, the QaaS is implemented in a centralized SDN controller to provide efficient and flexible key allocation. The network structure consists of three planes: infrastructure, control, and application planes.

QKD nodes, placed in the infrastructure plane, are equipped with **OpenFlow Agents (OFA),** which communicate and share relevant information with the SDN controller via the **OpenFlow Protocol (OFP)**. In a centralized control layer, a topology module collects and stores information about the QKD network topology and nodes, whereas a resource module deals with more dynamic data, gathering and storing real-time secret key rates of QKD links. Multiple clients request **secret-key rate (SKR)** settings from the QKD infrastructure via the northbound interface, which is realized as a REST-based API using the HTTP. The interface provides three simple methods for creating, modifying, and deleting services. According to the requirements, a centralized control layer allocates available SKR on each QKD link along a path to fulfill service requests from multiple clients.

The same authors proposed the **Multi-Tenant Key Algorithm (MTKA)** in a 2019 study [76]. It adheres to the same principles as SDQaaS in providing a centralized and detailed view of the QKD network infrastructure and its resources. Efficient secret key resource usage can be achieved by maximizing the **Matching Degree (MD)** function, a sum of success probability, and key resource utilization multiplied by weighting factors α and β. The success probability is defined as a ratio of admitted client requests to total requests, while key resource utilization is defined as a ratio of assigned SKR slots to total SKR slots. The MTKA assumes that the set of client requests, network topology, and SKRs on each QKD link are known in advance. It returns a list of admitted requests and the number of occupied key resources in a network. The MD function should then be optimized to maximize key resource usage.

3.14 NKPs DDKA-QKDN Scheme

In 2022, a **Dynamic-On-Demand Key Allocation (DDKA)** scheme for **QKD networks (QKDNs)** has been proposed [77]. To efficiently use key resources in the **Quantum Internet of Things (Q-IoT)** scenario, the DDKA-QKDN scheme dynamically allocates key resources per the application request. The scheme simultaneously addresses the key supplement of QKPs as well. Again, the concept is realized using SDN technology.

The primary criterion used to process application requests is request arrival time. However, owing to the large number of requests that can arrive simultaneously, requests are queued for processing. A new strategy that prioritizes queues had to be developed to decrease queuing delays. There are two main request requirements: request key quantity K_{qua} and security K_{sec}. These two requirements are directly related, as the greater key quantity implies greater key security and vice versa. Security levels Sec are different for different applications, ranging from low-security levels ($Sec = 0$), where data is transmitted in plaintext, to high-security levels ($Sec > 0$), where different-sized keys are used to provide data confidentiality. Keys with the following sizes are available based on security level: 128 bits, 256 bits, 512 bits, 1024 bits, and 2048 bits. Smaller key quantity requests are given higher priority regarding system efficiency when providing keys. As a result, storages are consumed slowly, and the likelihood of serving subsequent requests increases. When it comes to system security, however, higher key security requests are prioritized. As a result, when calculating response weight value $est(K_i)$ the DDKA-QKDN scheme accounts for the trade-off between the two using $\omega \in [0, 1]$, as shown by Equation (2). When multiple requests arrive simultaneously, they are sorted in ascending order of response weights.

$$est(K_i) = (1 - \omega)\ln K_{qua} + \omega \ln(10 - K_{sec}) \tag{2}$$

If the request cannot be fulfilled due to a lack of keys, a request is made to supplement the pool with new keys. Key supplement requests are handled in the same manner as application key requests. The primary criterion is request arrival time, and the response weight value is used to process requests that arrive at the same time in an orderly manner. Since the key supplement process can take a long time to complete (might require a key relay process), thereby prolonging the waiting time of application requests, storage thresholds are introduced to manage key supplements dynamically. As a result, storages are promptly supplemented to prevent exhaustion. On the other hand, a higher threshold value is set to prevent overloading storages with key material that might not be consumed anytime soon and could reduce key security.

3.15 KISTI Key Management

Korea Institute of Science and Technology Information (KISTI) researchers presented a QKMS design plan in 2022 to ensure the physical layer security of the next-generation KREONET[7] [78, 79]. The architecture is layered, with transport and quantum planes decoupled. This article only describes the high-level design of functional blocks, and there is not much to reflect on in implementation. The structure is identical to that of the NICT QKD platform, including the quantum, KMA, and KSA layers. Every node in the network creates and keeps keys with every other node. As a result, two key pools are distinguished: those that store key material between adjacent nodes via quantum connectivity and those that store key material generated via key relay.

[7] KREONET is a national research and development network managed by KISTI.

Keys supplied from the QKD devices are intercepted in the QKD protocol abstraction layer and converted to the ETSI QKD 014 standard format before being delivered to the KMA. Like Quantum Canada's approach, the scheme defines fallback methods to deal with scarce key resources efficiently. The first option provides key relay even when the key material available on the link is insufficient to meet the request. The encryption key is derived from the QKD key using the **Hash-based Key Derivation Function (HKDF)**, which is then used to OTP encrypt the relay key. The second method employs HKDF on the QKD key to generate supply keys. The authors conclude that because the HKDF is used to generate supply keys, quantum keys can be kept in key pools in fixed sizes. Because the application requests different key sizes, this requirement is filled using the HKDF. However, if implemented in this manner, the QKD network will be unable to serve true ITS keys. This is because classical key expansion methods do not produce true random output.

3.16 AIT Key Manager

In 2023, authors from the **Austrian Institute of Technology (AIT)** in Vienna published a report on a KM being developed within the EuroQCI framework [80]. Their paper describes a KMS prototype that adheres to ETSI QKD 004, 014, and 015 standards and considers ITU-T Y series recommendations. The main contributions of the AIT are recommendations for KMS-to-KMS and KMS-to-SDN agent interface methods. In addition, the ETSI QKD 004 interface has been enhanced with a push mode to support communication between QKD devices and KMS.

The ETSI QKD 004 interface specification advocates using an interface between KMSs of different hierarchical levels within the same node and between QKD devices and KMS. However, the authors highlight differences between the key–supply interface and the QKD device–KMS interfaces. In general, applications want to pull keys from the KMS on demand, whereas QKD devices wish to push keys to the KMS once generated. The authors propose a push mode for the ETSI QKD 004 interface for these reasons. Figure 13(a) illustrates the pull-and-push mode.

The AIT KM prototype proposes methods for key modifications, applications, key streams, and peer availability and, finally, defines a concrete protocol to carry out these methods. The protocol of choice is a CoAP protocol. It is a REST-based, specialized web transfer protocol designed for limited-capability devices. For key modification, the following methods are introduced.

- **new_key_batch** synchronizes key material obtained from QKD devices. To reduce internal key consumption, keys are synchronized in batches. The message contains key IDs and a **Message Authentication Code (MAC)** calculated from the message and key data transmitted.
- **forward_keys** performs key relay and includes encrypted keys, key IDs, and a destination ID that guides subsequent hops.
- **split_key** splits key into smaller key blocks. It contains a key ID, a list of new key IDs, and their corresponding new lengths.
- **merge_keys** merges smaller keys into one larger key. It contains a list of key IDs, a new key ID, and the corresponding new length.
- **delete_keys** deletes keys identified with key IDs.
- **make_keys_internal** reserves a group of keys identified with given key IDs for internal use. This includes authentication and encryption keys for synchronization and relay purposes.
- **make_keys_external** reserves a group of keys identified with given key IDs for peer applications. It contains a list of key IDs and a key stream ID.

When registering a new application key stream, a peer KMS is notified using the **new_app** method. This method includes an application ID, the source and destination address, a key stream ID, and a QoS. Closing of the application key streams is synchronized using **key_stream_closed**

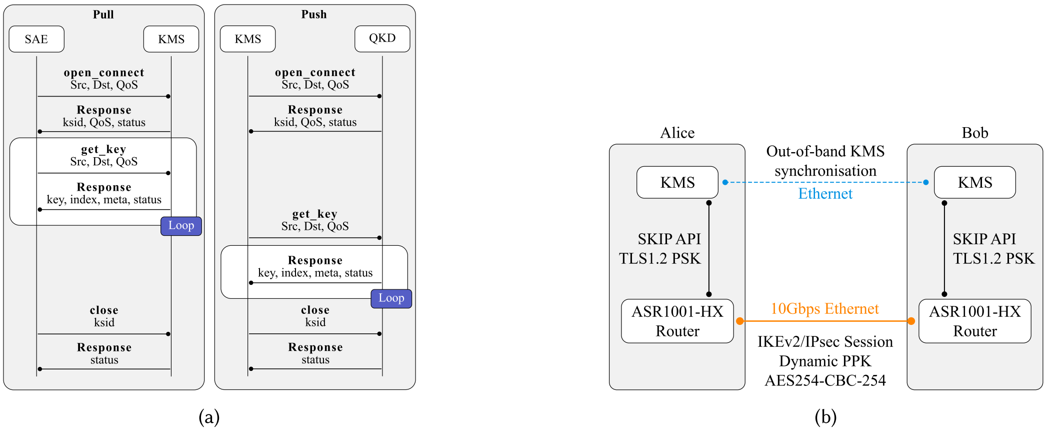


Fig. 13. (a) The enhancement of the ETSI 004 interface with a push mode defined by the AIT. It is used to supply generated keys from the QKD devices to the KMS [80]. (b) Cisco Lab setup.

notification. It includes a key stream ID. The KMSs can obtain or inform peers about their status using **get_status** and **post_status** methods. An interface is also defined between the KMS and the SDN agent, located within the same security boundary. In addition, the authors discuss PQC hybridization techniques at the KMS level. This is how the dual secret key agreement is carried out.

3.17 CISCO Key Management System

One of the Cisco testbeds is illustrated in Figure 13(b). It consists of the KMS and a Cisco Router ASR1001-HX. The KMS collects keys from the various vendors' QKD devices and makes them accessible to the network routers through the SKIP protocol. Additional experiments were carried out to transfer keys over complex and arbitrary topologies using trusted-relay nodes. The KMS is constructed utilizing the Python Flask framework as its API gateway. A proxy server fronts the KMS to safeguard the API gateway from possible attack vectors [51]. Further security is provided through the implementation of the TLS1.2 PSK authentication between the routers and the KMS.

The KMS acquires keys in bulk by reading binary files from the QKD device. The sequence of file retrieval varies depending on the QKD vendor's specifications. The binary file is converted into a hex format at the KMS to minimize storage requirements. The hex-formatted keys are stored in 64-bit blocks as ASCII text in the internal SQLite database. The availability of keys in the database is continuously monitored, and when a certain threshold is reached, a process is initiated to replenish the database on time.

The keys are supplied to the routers via the Cisco SKIP protocol. As previously mentioned, this connection is secured with the TLS to ensure client authenticity. Each router first registers itself using a unique ID to ensure traceability and identification of local and remote devices. The first requester is expected to initiate a key exchange with the remote key storage device. The two key storages will jointly formulate a Key and a KeyID. The second GET request from the remote system is made after the Key Storage has been negotiated with its counterpart. The key storage negotiates a key with the given KeyID and transfers its copy of the Key and KeyID to the router in the query response.

4 Discussion

The QKD key manager market is a prospective field with an estimated multi-million USD value in the coming years. Several KMs are commercially available as dedicated hardware solutions, offered by companies such as Toshiba, IDQ Solteris, QuantumCTek, and others [81]. However, detailed information about their deployment and implementation are limited. In contrast, Section 3 provides a comprehensive overview of the evolution of KMs through a chronological analysis of their development. Considering the timeline spanning from their earliest iterations aimed at achieving basic functionality to the present-day versions boasting enhanced features and optimizations, it becomes challenging to individually acknowledge each solution while giving due credit. By considering the functional requirements outlined in the ITU-T recommendations for the key management layer [40], we can assess how various solutions address these requirements, if at all. In the following subsections, we consider several key requirements and functionalities of KMs and conduct a comparative analysis of existing solutions. By conducting this analysis, we aim to pinpoint the current gaps and challenges in addressing the key management problem within QKD networks.

4.1 Compatibility with Various Types of QKD Modules

The KMs are required to collect keys from QKD modules through the appropriate interface and to be compatible with various kinds of QKD modules that implement different protocols. While this requirement appears straightforward to achieve, most commercial KMs are packaged with QKD modules. As a result, they use proprietary methods of communication between QKD modules and KMAs. To facilitate fair representation of all QKD equipment manufacturers in the market, it is imperative to establish clear guidelines for communication and interoperability between various QKD module manufacturers and KMs. The ETSI 004 application interface is designed to facilitate communication among KMs at various hierarchical levels. As a result, it is possible to keep a collection of proprietary QKD modules and KMs while installing a hierarchically superior KM from any manufacturer that collects keys through the interface. However, the ETSI 004 application interface is not widely used in practice owing to QoS features that are difficult to support. In addition, a QKD module manufacturer may not implement its own KM solution. As a result, the standardized interface for delivering keys from QKD modules to KMs is critical to achieving interoperability.

This requirement was first highlighted during the SECOQC project (see Section 3.2), when it was critical to allow different implementations of QKD protocols to deliver keys to the Q3P modules. However, the interface was never explained in detail and its implementation is unknown (at least to the general public). Although the interface between QKD modules and KMs is frequently mentioned, it was not previously described until recently, when ETSI interfaces were modified for this purpose. The KISTI key management solution (see Section 3.15) addresses the issue via a QKD protocol abstraction layer. The abstraction layer collects keys generated by heterogeneous QKD modules, converts them to the ETSI 014 standard format, and delivers them to the KM for storage. However, it is unclear how keys are transferred to the abstraction layer and what format the message takes. The AIT KM design proposes a more comprehensive solution, which includes an extension of the ETSI 004 interface (see Section 3.16). It defines a push mode for the GET_KEY method, which delivers keys in standard-defined format. This approach supports the capability to develop QKD modules independently of a KM, as keys can be transmitted immediately after generation.

4.2 Key Supply to the User Network

The KM is required to provide the requested number of keys to cryptographic applications via a key supply interface with security capabilities. In addition, the KM is required to apply the key management policies. In this context, we examine several factors regarding access to QKD network services: the key-supply interface, the approach to resource sharing, fallback methods, and application priorities. Table 2 provides a summary of these features for the current approaches, where “N” indicates that the feature is not supported, “Y” signifies support, and “–” denotes that

Table 2. Comparison of the Key Managers in Terms of Key-Supply Interface Functionalities

Features Solutions	Year	Key–Supply Interface	Resource Sharing	Request Priority	Fallback Method
DARPA	2002–07	List available Qblocks; reserve a certain Qblock; obtain Qblock.	N	N	N
SECOQC	2007–09	A TCP connection with specified key-supply rate.	Reservation based	N	N
NIST	2008	Establish a session; get number of available bytes; obtain a number of bytes; disengage; close the session.	Sharing based on registered application key rates	N	N
QCC	2008	Establish a session with defined key rate and key buffer size.	Sharing based on registered application key rates	N	N
NEC	2009	–	N	N	N
MagiQ Technologies	2010	–	Sharing based on registered application key rates	N	N
SwissQuantum	2009–11	–	N	N	Y
QoS–supported KM	2011	–	Sharing based on priority class and reservation based	Y	N
NECTEC	2012	–	N	N	N

Toshiba	2016	Providing status;
		encryption key provisioning;
		decryption key provisioning
		(evolved in ETSI GS QKD 014 standard)
			Sharing based on registered application key rates	N	N

NICT QKD platform	2017	–	N	N	N
Quantum Canada	2018	–	N	Y	Y
NSFC SDQaaS	2019	–	Reservation based	N	N
NKPs DDKA-QKDN	2022	–	N	Y	N
KISTI	2022	ETSI GS QKD 014	N	N	Y
AIT	2023	ETSI GS QKD 014; ETSI GS QKD 004	N	N	N
Cisco	2023	SKIP protocol	N	N	N

the feature is not explicitly discussed, making its support and detail unclear based on available information.

While two standard interfaces have been established to regulate the communication between cryptographic applications and KMs, it is still valuable to analyze earlier proposed interfaces for comprehensive understanding. The earliest form of this interface was suggested within the DARPA quantum network framework (see Section 3.1). Cryptographic applications can reserve and retrieve keys from the QKD network using this straightforward interface. However, it is the applications' responsibility to negotiate and reserve keys before they can be retrieved. The interface does not allow for any additional requirements, such as requested key sizes. The functionalities and approach of this interface are outdated, rendering it unsuitable for application in today's implementations. After several years, Toshiba defined a similar interface, which evolved into the current ETSI QKD 014 standard. It is used in the newer KISTI and AIT key management solutions. The NIST and QCC interfaces can be linked to ETSI QKD 004 key session establishment logic. Both require the establishment of key sessions with defined application key rates. However, these implementations lack support for QoS, which is why the ETSI QKD 004 standard exists. Both approaches ensure a fair sharing of resources, but the supply of keys according to the application's QoS requirements is not guaranteed. Finally, Cisco defined its key-supply interface, the SKIP protocol. However, according to the publicly available description provided (see Section 2.4.3), the underlying methods of the protocol are very similar to the ETSI QKD 014. From this discussion, it is evident that two prevailing approaches to implementing the interface exist: one is simplistic, enabling key retrieval on demand, whereas the other entails establishing a session with a desirable key rate. The ETSI interfaces effectively encompass the functionalities of all previous approaches.

When considering sharing available key resources, most solutions rely on application key rates as a metric. Each application receives only a proportional share of resources based on its requirements. In the case of limited resources and a large number of applications, the share that each application receives will be significantly lower than its requirements. In contrast, the SECOQC solution confirms the availability of resources before granting application access. However, in some cases, the provided key rate is adjusted to reflect the state of the network and the service is not guaranteed. The NSFC SDQaaS approach considers only the SKR of QKD links, not the number of available keys in storage. Each application is guaranteed the key rate it requires, allowing for the provision of a service of guaranteed quality, assuming that the performance of the QKD links is stable. However, not all applications, regardless of key rate requirements, should have equal priority in accessing resources. It is reasonable to assume that some applications are of greater critical importance and should thus receive higher priority [82]. Priorities of application requirements were rarely considered in this context. The QoS-supported key manager distinguishes three service classes. Services are differentiated according to the time (delay) requirement. The highest priority class, the key-guaranteed service, is based on a resource reservation approach, whereas the other two classes use a queuing method. The Quantum Canada approach proposes classifying applications based on security requirements and identifies five categories. However, it appears that these classes are only used to identify fallback methods for each class rather than to prioritize requests. The NKPs DDKA-QKDN scheme takes a slightly different approach, prioritizing requests from the QKD network perspective. The priority of the request is determined by balancing security and the key quantity requirements. Based on this analysis, it can be concluded that there is still no established approach to resource sharing and prioritizing requests to ensure varying levels of service for applications of different purposes. These functionalities are crucial, highlighting the current deficiencies in key manager capabilities, especially considering the growing trend of integrating QKD networks as enterprise services.

The debate over whether fallback methods should be supported within QKD networks is ongoing. Transitioning to new cryptographic solutions typically takes time, making hybrid approaches a preferred strategy [83]. These approaches are primarily adopted to ensure redundancy, maintaining secure communication as long as at least one cryptographic primitive in the hybrid system remains secure. There is a growing interest in integrating PQC methods into key management systems. This approach allows for the utilization of both PQC and QKD methods for dual key agreement, providing redundancy and a straightforward fallback option if QKD key resources become unavailable. However, it is crucial to assess the security requirements of the application and determine whether they can be met with less secure cryptographic keys. The SwissQuantum solution adopts this approach, which is also listed in the ITU-T functional requirements for the key management layer. Quantum Canada's approach differs somewhat and entails significantly more complex key lifecycle management depending on the scenario. This raises the question: why should a QKD network concern itself with whether a key will be used to derive multiple session keys? Such discussions might be better suited for the service layer rather than the network layer. However, the PQC could be valuable within the QKD network for ensuring robust end-to-end authenticity of KMs [84, 85] and for initial authentication at the quantum layer.

4.3 Secure Key Storage and Key Formatting

The KM is required to securely store and format keys where necessary for internal purposes or for key supply or key relay, including combining or splitting where lengths are not appropriate. The necessity for secure key storage has been acknowledged from the outset owing to the unique characteristics of the QKD process. Thus, the existing approaches studied in this article include this fundamental requirement of the KM. Merely stating that keys will be stored is insufficient; the manner in which they are stored is equally crucial, as it greatly influences the effectiveness of key servicing capabilities. Table 3 provides a tabular comparison of existing KMs from the perspective of key storage realization and closely related techniques associated with this functionality, where

Table 3. Comparison of the Key Managers in Terms of Key Storage and Related Functionalities

Features Solutions	Year	Local Key Storage	Global Key Storage	Storage Method	Reformat Feature	Logical Key Division	Storage Thresholds
DARPA	2002–07	Y	N	Fixed–sized blocks	N	N	N
SECOQC	2007–09	Y	N	Homogeneous blocks	Y – On reception; N – On supply	Y Encryption & Decryption Keys	Y
NIST	2008	Y	N	Homogeneous bytes	Y Supply in multiple of bytes	Y Session–based	N
QCC	2008	Y	N	–	–	Y Session–based	Y
NEC	2009	Y	Y	Fixed–size blocks	N	Y Encryption and decryption Keys	N
MagiQ Technologies	2010	Y	N	Fixed–sized blocks	N	Y Session–based	N
SwissQuantum	2009–11	Y	Y	–	–	Y Session–based	–
QoS–supported KM	2011	Y	–	–	–	–	–
NECTEC	2012	Y	Y	–	–	Y Session–based	N
Toshiba	2016	Y	Y	Homogeneous blocks	Y – On reception N – On supply	Y Encryption and decryption keys; Session–based	N
NICT QKD Platform	2017	Y	Y	–	–	–	N
Quantum Canada	2018	Y	Y	–	–	Y Local and Global keys; encryption and decryption keys	N
NSFC SDQaaS	2019	Y	Y	–	–	Y Session–based	N
NKPs DDKA–QKDN	2022	Y	Y	Five predetermined fixed–sized blocks	Y – On reception N – On supply	N	Y
KISTI	2022	Y	Y	–	Y HKDF	–	N
AIT	2023	Y	–	–	Y Split and merge	–	N
Cisco	2023	Y	N	Fixed-sized blocks (64 bits)	Y	–	Y

"N" indicates the feature is not supported, "Y" signifies support, and "–" denotes that the feature is not explicitly discussed, making its support and detail unclear based on available information.

An in-depth analysis of existing approaches has uncovered several prevailing key storage designs alongside numerous shortcomings that require attention and resolution. To address the inefficiencies in key delivery identified within the DARPA quantum network, the SECOQC approach (see Section 3.2) suggests categorizing keys based on their intended purpose, separating them into encryption and decryption keys. This approach allows KMs to facilitate the seamless utilization of keys for delivery or key relaying, eliminating concerns about disagreements or collisions in key access. Cryptographic applications, as well as internal processes such as key relaying, have shared access to the singular encryption key storage. The decryption key storage, however, is accessed only upon instruction from the peer KM or when a slave cryptographic application requests the service. Given the significance of the SECOQC architecture, this key management approach is adopted in Toshiba, and it is highly probable that it is also utilized in the NICT QKD platform. This assumption is based on the fact that the NICT QKD platform is built upon the work developed for the Tokyo QKD network, which employed the same architecture as SECOQC. Similarly, the following solutions define singular encryption and decryption key storages: NEC (see Section 3.5) and Quantum Canada (see Section 3.12). The encryption and decryption storages in SECOQC (and, thus, Toshiba) contain predefined-size key blocks that are sequentially stored. This approach makes it straightforward to reformat keys that arrive from QKD modules in large and varying sizes into blocks of predefined sizes. However, it also implies that keys are suitable for use/supply in an ordered manner and in a single predefined size — the block size. The block size, however, has not been discussed. Similarly, NEC's key management solution defines keys as fixed-size files.

NIST (see Section 3.3) introduced a session-based key storage approach that is used in several solutions, including the QCC security processor (see Section 3.4), MagiQ Technologies (see Section 3.6), SwissQuantum (see Section 3.7), NECTEC (see Section 3.9), and NSFC SDQaaS framework (see Section 3.13). Each cryptographic application is assigned its own key storage and a subset of the available keys from the common storage. Each application typically registers with a desired key rate, which serves as a guideline for assigning available keys to multiple buffers. One of the packet scheduling algorithms can be used to ensure that keys are assigned fairly across various buffers. The NIST defines session-based storage as containing keys that are reformatted to a one-byte size and stored sequentially. This method is appropriate for meeting the varying key size requirements of cryptographic applications, as any size in bits (that is a multiple of 8) can be supplied by combining multiple bytes from storage. The MagiQ Technologies key management approach specifies that keys are stored in predetermined fixed-size blocks, but the block size is unknown. Because the reformatting is not specified, the keys are most likely provided in block format.

Although the discussed approaches to key storage are primarily for local keys, the same constructions could be used for global keys as well. However, it is worth noting that the approach to global key storage is rarely discussed. The SECOQC makes it abundantly clear that global key storage and management are not supported. Upon completion of global key distribution, keys are simply provided to the requesting application. Given that global keys are distributed in larger blocks to reduce encryption overhead, the system may be inefficient because large keys are supplied to the application regardless of its size requirements. This transfers responsibility for global key management to the application that will use the obtained key. The Toshiba defines the global key management function but does not discuss the storage method.

The deficiency of KMs becomes apparent in solutions that store keys as fixed-sized blocks, whether they are in shared encryption storages or dedicated application storages. This oversight occurs solely because of neglecting key formatting when requested sizes do not align with block sizes. For example, the key management systems of DARPA and MagiQ Technologies clearly states that the keys are received from the quantum layer at a predetermined size and, thus, stored without any reformatting. Since the key formatting is not supported at supply, the system is expected to have low efficiency as large chunks of keys are delivered on any request, even requests that require tiny quantities. The NKP DDKA-QKDN scheme partially addresses this problem by storing keys in five different but predetermined sizes, namely 128, 256, 512, 1024, and 2048 bits. This means that the reformat function is executed before storage. However, it does not define the ratio of how many keys and of which size should be created at a given time. More sophisticated ways of solving this problem have been proposed by AIT and partly by KISTI by introducing the function of reformatting on supply. By having this function, it can be concluded that both are indeed store keys in fixed-sized blocks. KISTI uses one approach considered controversial in terms of QKD networks, namely, the use of the HKDF function to derive smaller keys, which are supplied on demand, from large keys. Because many smaller keys are derived from a single key, they are not eligible for the ITS profile. The AIT defines split and merge methods for reformatting available keys to desired sizes at the key management layer. This solution can apply to all solutions that lack this functionality. The Cisco key manager stores keys in fixed-sized 64-bit blocks, supporting the reformat function before storage. However, it does not address how to create the supply key of the requested size. The key may indeed be of any length–in this case, it may be a multiple of 64–for the application mentioned in Section 3.17, where it is used in the key derivation procedure.

Some of the mentioned research works discuss storage thresholds, which are important for improving efficiency and enabling continuous supply without interruption. The SECOQC key management approach monitors the available number of decryption keys and requires a refill procedure if this number is critically low. Since the keys used for decryption have encryption copies on another node, timely refilling allows for continuous transfer of sensitive data. Similarly, the QCC key manager applies thresholds to a session-based approach. Suppose that the number of keys in the application's dedicated buffer falls below a threshold value. In that case, new keys, if available, are assigned promptly so that the application does not experience interruptions in key supply. Similarly, the Cisco key manager defines a threshold value that causes new keys to be pulled from the QKD devices. Threshold values can be assigned to global key storage, as in NKP's DDKA-QKDN scheme. This allows for the distribution of a sufficient number of global keys in advance. Global key storage thresholds are more important and complex than previously discussed thresholds introduced in SECOQC and QCC solutions. Since the global key distribution time can vary, the lower threshold should be set dynamically to account for this distribution time.

From this discussion, it is apparent that the methods of storing and managing keys within the key management system are still notably constrained. Furthermore, while we can identify various approaches to key storage, there is a lack of research examining the comparative effectiveness of these approaches. A system that supports both identified key storage approaches—encryption and decryption, and session based—would be capable of accommodating both ETSI interfaces for accessing services. A shared encryption key storage would cater to ETSI 014 requests, whereas the session-based approach would respond to the session-based nature of the ETSI 004 interface. Additionally, many solutions lack a critical key formatting capability, which may compromise the QKD service. Applications often receive key blocks larger in size than required, reducing the probability of serving other applications due to decreased key material in storages. The initial design of the AIT key manager indicates support for merge and split operations. However, questions persist regarding the efficiency with which these operations can be executed. Moreover, there is a significant gap in supporting a large number of applications with guaranteed service levels and ensuring fair sharing of scarce resources. Presently, if supported at all, solutions offer a ratio of available key material based on desired application rates. There is no assurance of a guaranteed level of service. For technology to be suitable for application in critical infrastructures, it is crucial to have a method of differentiating application priorities and ensuring guaranteed levels of service.

5 Key Challenges and Future Directions

This section presents a concise list of challenges and future endeavors in key management in QKD networks. Key management is the foundational task of QKD networks; without it, the key generation process would be ineffective. However, the development and optimization of the key management layer are significantly constrained due to the limited number of testbeds and real-world applications of QKD network services. These are identified key challenges to guide future directions:

- **Standardization of interfaces for communication between QKD modules and KMs:** To facilitate fair representation of all QKD equipment manufacturers in the market, it is imperative to establish clear guidelines for communication and interoperability between various QKD module manufacturers and KMs. Currently, there is no standardized interface that delineates this communication.
- **Effective key storage approaches:** The KM is required to effectively manage cryptographic keys, which includes key storage and formatting. It is critical that keys are stored in a manner that allows for efficient key addition and retrieval. Key formatting should also be supported and performed efficiently on demand because the application in critical infrastructures requires a minimum delay in key supply. There are no studies analyzing

the effectiveness of key storage designs, nor are there detailed approaches to effective key formatting.

- **Effective resource sharing and quality of service support:** With the growing push for enterprise integration of QKD networks, it is anticipated that a large number of users, i.e., cryptographic applications, will need to share limited key resources. The KM must prioritize applications of varying backgrounds and even support quality of service requirements for the most critical applications. Most approaches utilize desired key rates as a metric for allocating available resources. However, it is essential to consider the nature of the applications.
- **Standardization of interfaces for communication between remote KMs:** To achieve interoperability and ensure fair representation of different KM vendors, it is crucial to define the methods and protocols between remote KMs. This point presents a significant obstacle to developing and integrating QKD networks. As vendors endeavor to incorporate their QKD devices and KMs into a single device and rely on proprietary interfaces, the result is the imposition of a QKD network reliant on the catalog of a single vendor. The absence of a standardized interface between KMs hinders the broader integration of QKD networks, particularly in terms of key relaying, which is aimed to be addressed through the ETSI QKD 020 standard. However, there are concerns that this solution may not scale effectively and could potentially slow down the traffic within the QKD network.
- **Managing keys of different security levels:** In theory, QKD modules generate ITS keys at the quantum layer. Due to technological limitations in quantum communication processes [86], guaranteeing the ITS security level in practice remains challenging. To address this issue, an epsilon parameter was proposed [87] to quantify the security of the generated key. However, there is no publicly available research discussing a key management system capable of handling keys with different security levels. This feature would allow applications to specify the required security level of requested keys through, for example, extension fields in the key request body [42]. The same fields can also be utilized by applications to specify other requirements, such as the preferred key vendor, key age, and other relevant constraints.

6 Conclusion

The key management layer must be addressed for QKD networks to become a viable technology. It enables the realization of QKD networks by overcoming the point-to-point limitations of QKD links. It determines the QKD network's ability to provide a service with guaranteed QoS and delivery of keys to end users safely and on time. Finally, it enables the interoperability of QKD equipment by connecting different types of QKD links into a single QKD network. This article extensively reviews the evolution of KMs in trusted-relay QKD networks. It analyzes and compares existing solutions regarding key storage and service provisions. To the best of our knowledge, this is the first article to examine approaches to developing the KM component in QKD networks. The main contribution of this article is an in-depth analysis of existing approaches for developing key management systems as fundamental components of the QKD network.

References

[1] Hanif Ullah, Nithya Gopalakrishnan Nair, Adrian Moore, Chris Nugent, Paul Muschamp, and Maria Cuevas. 2019. 5G communication: An overview of vehicle-to-everything, drones, and healthcare use-cases. *IEEE Access* 7 (2019), 37251–37268.

[2] Roger A. Grimes. 2019. *Cryptography Apocalypse: Preparing for the Day when Quantum Computing Breaks Today's Crypto*. John Wiley & Sons, NJ.

[3] Peter W. Shor. 1999. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Review* 41, 2 (1999), 303–332.

[4] Duc-Thuan Dam, Thai-Ha Tran, Van-Phuc Hoang, Cong-Kha Pham, and Trong-Thuc Hoang. 2023. A survey of post-quantum cryptography: Start of a new race. *Cryptography* 7, 3 (2023), 40.

[5] Charles H. Bennett and Gilles Brassard. 1984. Quantum cryptography: Public key distribution and coin tossing. In *Proceedings of IEEE International Conference on Computers, Systems and Signal Processing*, Vol. 175. Steering Committee, 8.

[6] Peter W. Shor and John Preskill. 2000. Simple proof of security of the BB84 quantum key distribution protocol. *Physical Review Letters* 85, 2 (2000), 441.

[7] Charles H. Bennett, François Bessette, Gilles Brassard, Louis Salvail, and John Smolin. 1992. Experimental quantum cryptography. *Journal of Cryptology* 5 (1992), 3–28.

[8] Gilles Brassard and Louis Salvail. 1993. Secret-key reconciliation by public discussion. In *Workshop on the Theory and Application of of Cryptographic Techniques*. Springer, 410–423.

[9] Charles H. Bennett, Gilles Brassard, and Jean-Marc Robert. 1986. How to reduce your enemy's information. In *Advances in Cryptology–CRYPTO'85 Proceedings 5*. Springer, 468–476.

[10] Charles H. Bennett, Gilles Brassard, and Jean-Marc Robert. 1988. Privacy amplification by public discussion. *SIAM Journal on Computing* 17, 2 (1988), 210–229.

[11] Emir Dervisevic and Miralem Mehic. 2021. Overview of quantum key distribution technique within IPsec architecture. In *ISCRAM 2021 Conference Proceedings – 18th International Conference on Information Systems for Crisis Response and Management*. Virginia Tech, 391–403.

[12] Romain Alleaume, Francois Roueff, Eleni Diamanti, and N. Lütkenhaus. 2009. Topological optimization of quantum key distribution networks. *New Journal of Physics* 11, 7 (2009), 075002.

[13] Chip Elliott. 2002. Building the quantum network. *New Journal of Physics* 4, 1 (2002), 46.

[14] Juan Yin, Yuan Cao, Yu-Huai Li, Sheng-Kai Liao, Liang Zhang, Ji-Gang Ren, Wen-Qi Cai, Wei-Yue Liu, Bo Li, Hui Dai, et al. 2017. Satellite-based entanglement distribution over 1200 kilometers. *Science* 356, 6343 (2017), 1140–1144.

[15] Sheng-Kai Liao, Wen-Qi Cai, Wei-Yue Liu, Liang Zhang, Yang Li, Ji-Gang Ren, Juan Yin, Qi Shen, Yuan Cao, Zheng-Ping Li, et al. 2017. Satellite-to-ground quantum key distribution. *Nature* 549, 7670 (2017), 43–47.

[16] Shuang Wang, Zhen-Qiang Yin, De-Yong He, Wei Chen, Rui-Qiang Wang, Peng Ye, Yao Zhou, Guan-Jie Fan-Yuan, Fang-Xiang Wang, Yong-Gang Zhu, et al. 2022. Twin-field quantum key distribution over 830-km fibre. *Nature Photonics* 16, 2 (2022), 154–161.

[17] Jiu-Peng Chen, Chi Zhang, Yang Liu, Cong Jiang, Dong-Feng Zhao, Wei-Jun Zhang, Fa-Xi Chen, Hao Li, Li-Xing You, Zhen Wang, et al. 2022. Quantum key distribution over 658 km fiber with distributed vibration sensing. *Physical Review Letters* 128, 18 (2022), 180502.

[18] Sebastian Philipp Neumann, Alexander Buchner, Lukas Bulla, Martin Bohmann, and Rupert Ursin. 2022. Continuous entanglement distribution over a transnational 248 km fiber link. *Nature Communications* 13, 1 (2022), 6134.

[19] Jing-Yang Liu, Xiao Ma, Hua-Jian Ding, Chun-Hui Zhang, Xing-Yu Zhou, and Qin Wang. 2023. Experimental demonstration of five-intensity measurement-device-independent quantum key distribution over 442 km. *Physical Review A* 108, 2 (2023), 022605.

[20] Heng Wang, Yang Li, Yaodi Pi, Yan Pan, Yun Shao, Li Ma, Yichen Zhang, Jie Yang, Tao Zhang, Wei Huang, et al. 2022. Sub-Gbps key rate four-state continuous-variable quantum key distribution within metropolitan area. *Communications Physics* 5, 1 (2022), 162.

[21] Fadri Grünenfelder, Alberto Boaron, Giovanni V. Resta, Matthieu Perrenoud, Davide Rusca, Claudio Barreiro, Raphaël Houlmann, Rebecka Sax, Lorenzo Stasi, Sylvain El-Khoury, et al. 2023. Fast single-photon detectors and real-time key distillation enable high secret-key-rate quantum key distribution systems. *Nature Photonics* 17, 5 (2023), 422–426.

[22] Wei Li, Likang Zhang, Hao Tan, Yichen Lu, Sheng-Kai Liao, Jia Huang, Hao Li, Zhen Wang, Hao-Kun Mao, Bingze Yan, et al. 2023. High-rate quantum key distribution exceeding 110 Mb s−1. *Nature Photonics* 17, 5 (2023), 416–421.

[23] Miralem Mehic, Stefan Rass, Peppino Fazio, and Miroslav Voznak. 2022. *Quantum Key Distribution Networks: A Quality of Service Perspective*. Springer, Cham, Germany.

[24] Miralem Mehic, Marcin Niemiec, Stefan Rass, Jiajun Ma, Momtchil Peev, Alejandro Aguado, Vicente Martin, Stefan Schauer, Andreas Poppe, Christoph Pacher, et al. 2020. Quantum key distribution: A networking perspective. *ACM Computing Surveys (CSUR)* 53, 5 (2020), 1–41.

[25] ITU-T Y.3803. 2020. Quantum Key Distribution Networks – Key Management. (December 2020).

[26] Miralem Mehic, Libor Michalek, Emir Dervisevic, Patrik Burdiak, Matej Plakalovic, Jan Rozhon, Nerman Mahovac, Filip Richter, Enio Kaljic, Filip Lauterbach, et al. 2023. Quantum cryptography in 5g networks: A comprehensive overview. *IEEE Communications Surveys & Tutorials* 26, 1 (2023), 302 – 346.

[27] Romain Alléaume, Cyril Branciard, Jan Bouda, Thierry Debuisschert, Mehrdad Dianati, Nicolas Gisin, Mark Godfrey, Philippe Grangier, Thomas Länger, Norbert Lütkenhaus, et al. 2014. Using quantum key distribution for cryptographic purposes: A survey. *Theoretical Computer Science* 560 (2014), 62–81.

[28] Jeffrey D. Morris, Michael R. Grimaila, Douglas D. Hodson, David Jacques, and Gerald Baumgartner. 2014. A survey of quantum key distribution (QKD) technologies. In *Emerging Trends in ICT Security*. Elsevier, Amsterdam, Netherlands, 141–152.

[29] Eleni Diamanti, Hoi-Kwong Lo, Bing Qi, and Zhiliang Yuan. 2016. Practical challenges in quantum key distribution. *npj Quantum Information* 2, 1 (2016), 1–12.

[30] Matthias Geihs, Oleg Nikiforov, Denise Demirel, Alexander Sauer, Denis Butin, Felix Günther, Gernot Alber, Thomas Walther, and Johannes Buchmann. 2019. The status of quantum-key-distribution-based long-term secure internet communication. *IEEE Transactions on Sustainable Computing* 6, 1 (2019), 19–29.

[31] Feihu Xu, Xiongfeng Ma, Qiang Zhang, Hoi-Kwong Lo, and Jian-Wei Pan. 2020. Secure quantum key distribution with realistic devices. *Reviews of Modern Physics* 92, 2 (2020), 025002.

[32] Fabio Cavaliere, Enrico Prati, Luca Poti, Imran Muhammad, and Tommaso Catuogno. 2020. Secure quantum communication technologies and systems: From labs to markets. *Quantum Reports* 2, 1 (2020), 80–106.

[33] Purva Sharma, Anuj Agrawal, Vimal Bhatia, Shashi Prakash, and Amit Kumar Mishra. 2021. Quantum key distribution secured optical networks: A survey. *IEEE Open Journal of the Communications Society* 2 (2021), 2049–2083.

[34] Omar Amer, Vaibhav Garg, and Walter O. Krawec. 2021. An introduction to practical quantum key distribution. *IEEE Aerospace and Electronic Systems Magazine* 36, 3 (2021), 30–55.

[35] Chia-Wei Tsai, Chun-Wei Yang, Jason Lin, Yao-Chung Chang, and Ruay-Shiung Chang. 2021. Quantum key distribution networks: Challenges and future research issues in security. *Applied Sciences* 11, 9 (2021), 3767.

[36] Yuan Cao, Yongli Zhao, Qin Wang, Jie Zhang, Soon Xin Ng, and Lajos Hanzo. 2022. The evolution of quantum key distribution networks: On the road to the qinternet. *IEEE Communications Surveys & Tutorials* 24, 2 (2022), 839–894.

[37] Randall Atkinson and Stephen Kent. 1995. *Security Architecture for the Internet Protocol*. Technical Report. RFC 1825, August.

[38] Charlie Kaufman, Paul Hoffman, Yoav Nir, Pasi Eronen, and Tero Kivinen. 2010. *Internet Key Exchange Protocol Version 2 (IKEv2)*. Technical Report. RFC 5996, September.

[39] ITU-T Y.3800 (2019) Corrigendum 1. 2020. Overview on Networks Supporting Quantum Key Distribution (April 2020).

[40] ITU-T Y.3801. 2020. Functional Requirements for Quantum Key Distribution Networks (April 2020).

[41] ITU-T Y.3802. 2020. Quantum Key Distribution Networks – Functional Architecture (December 2020).

[42] ETSI GS QKD 014. 2019. Quantum Key Distribution (QKD); Protocol and Data Format of REST-based Key Delivery API (2019).

[43] ETSI GS QKD 004. 2020. Quantum Key Distribution (QKD); Application Interface. (2020).

[44] ITU-T Series Y Supplement 70. 2021. Quantum Key Distribution Networks – Applications of Machine Learning (July 2021).

[45] Emir Dervisevic, Filip Lauterbach, Patrik Burdiak, Jan Rozhon, Martina Slívová, Matej Plakalovic, Mirza Hamza, Peppino Fazio, Miroslav Voznak, and Miralem Mehic. 2022. Simulations of denial of service attacks in quantum key distribution networks. In *2022 XXVIII International Conference on Information, Communication and Automation Technologies (ICAT'22)*. IEEE, 1–5.

[46] Miralem Mehic, Stefan Rass, Emir Dervisevic, and Miroslav Voznak. 2022. Tackling denial of service attacks on key management in software-defined quantum key distribution networks. *IEEE Access* 10 (2022), 110512–110520.

[47] Hua Dong, Yaqi Song, and Li Yang. 2019. Wide area key distribution network based on a quantum key distribution system. *Applied Sciences* 9, 6 (2019), 1073.

[48] Nilesh Vyas and Paulo Mendes. 2024. Relaxing trust assumptions on quantum key distribution networks. *arXiv preprint arXiv:2402.13136* (2024).

[49] ETSI GS QKD 020. 2023. Quantum Key Distribution (QKD); Protocol and Data Format of REST-based Interoperable Key Management System API (2023). Draft.

[50] Scott Fluhrer, Panos Kampanakis, David McGrew, and Valery Smyslov. 2020. Mixing Preshared Keys in the Internet Key Exchange Protocol Version 2 (IKEv2) for Post-quantum Security. RFC 8784. (June 2020). DOI: http://dx.doi.org/10.17487/RFC8784

[51] Cisco. Configuring Quantum-Safe Encryption Using Postquantum Preshared Keys ([n. d.]). Retrieved from https://www.cisco.com/c/en/us/td/docs/routers/ios/config/17-x/sec-vpn/b-security-vpn/m-sec-cfg-quantum-encryption-ppk.html

[52] T. Dierks and E. Rescorla. 2008. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (August 2008). DOI: http://dx.doi.org/10.17487/RFC5246

[53] Chip Elliott, David Pearson, and Gregory Troxel. 2003. Quantum cryptography in practice. In *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*. ACM, 227–238.

[54] Chip Elliott and Henry Yeh. 2007. *DARPA Quantum Network Testbed*. Technical Report. BBN Technologies, Cambridge MA.

[55] Mehrdad Dianati and Romain Alléaume. 2007. Transport layer protocols for the Secoqc quantum key distribution (QKD) network. In *32nd IEEE Conference on Local Computer Networks (LCN'07)*. IEEE, 1025–1034.

[56] Mehrdad Dianati and Romain Alléaume. 2007. Architecture of the Secoqc quantum key distribution network. In *2007 1st International Conference on Quantum, Nano, and Micro Technologies (ICQNM'07)*. IEEE, 13–13.

[57] Mehrdad Dianati, Romain Alléaume, Maurice Gagnaire, and Xuemin Shen. 2008. Architecture and protocols of the future European quantum key distribution network. *Security and Communication Networks* 1, 1 (2008), 57–74.

[58] Momtchil Peev, Christoph Pacher, Romain Alléaume, Claudio Barreiro, Jan Bouda, W. Boxleitner, Thierry Debuisschert, Eleni Diamanti, Mehrdad Dianati, J. F. Dynes, et al. 2009. The SECOQC quantum key distribution network in Vienna. *New Journal of Physics* 11, 7 (2009), 075001.

[59] Oliver Maurhart. 2010. QKD networks based on Q3P. In *Applied Quantum Cryptography*. Springer, Berlin, Heidelberg, 151–171.

[60] Alan Mink, Lijun Ma, Tassos Nakassis, Hai Xu, Oliver Slattery, Barry Hershman, and Xiao Tang. 2008. A quantum network manager that supports a one-time pad stream. In *2nd International Conference on Quantum, Nano and Micro Technologies (ICQNM'08)*. IEEE, 16–21.

[61] Wakako Maeda, Akihiro Tanaka, Seigo Takahashi, Akio Tajima, and Akihisa Tomita. 2009. Technologies for quantum key distribution networks integrated with optical communication networks. *IEEE Journal of Selected Topics in Quantum Electronics* 15, 6 (2009), 1591–1601.

[62] Thomas Lorunser, Edwin Querasser, Thomas Matyus, Momtchil Peev, Johannes Wolkerstorfer, Michael Hutter, Alexander Szekely, Ilse Wimberger, Christian Pfaffel-Janser, and Andreas Neppach. 2008. Security processor with quantum key distribution. In *2008 International Conference on Application-Specific Systems, Architectures and Processors*. IEEE, 37–42.

[63] Andreas Neppach, Christian Pfaffel-Janser, Ilse Wimberger, Thomas Loruenser, Michael Meyenburg, Alexander Szekely, and Johannes Wolkerstorfer. 2008. Key management of quantum generated keys in IPsec. In *SECRYPT*. INSTICC Press, 177–183.

[64] Keun Lee and Audrlus Berzanskis. U.S. Patent US20060062392A1, Jan. 2010. Key Manager for QKD Networks. (U.S. Patent US20060062392A1, Jan. 2010).

[65] Damien Stucki, Matthieu Legre, Francois Buntschu, B. Clausen, Nadine Felber, Nicolas Gisin, Luca Henzen, Pascal Junod, Gérald Litzistorf, Patrick Monbaron, et al. 2011. Long-term performance of the SwissQuantum quantum key distribution network in a field environment. *New Journal of Physics* 13, 12 (2011), 123001.

[66] Xianzhu Cheng, Yongmei Sun, and Yuefeng Ji. 2011. A QoS-supported scheme for quantum key distribution. In *2011 International Conference on Advanced Intelligence and Awareness Internet (AIAI'11)*. IET, 220–224.

[67] Montida Pattaranantakul, Aroon Janthong, Kittichai Sanguannam, Paramin Sangwongngam, and Keattisak Sripimanwat. 2012. Secure and efficient key management technique in quantum cryptography network. In *2012 4th International Conference on Ubiquitous and Future Networks (ICUFN'12)*. IEEE, 280–285.

[68] Montida Pattaranantakul, Kittichai Sanguannam, Paramin Sangwongngam, and Chalee Vorakulpipat. 2015. Efficient key management protocol for secure RTMP video streaming toward trusted quantum network. *Etri Journal* 37, 4 (2015), 696–706.

[69] Yoshimichi Tanizawa, Ririka Takahashi, Hideaki Sato, Alexander R. Dixon, and Shinichi Kawamura. 2016. A secure communication network infrastructure based on quantum key distribution technology. *IEICE Transactions on Communications* 99, 5 (2016), 1054–1069.

[70] Ririka Takahashi, Yoshimichi Tanizawa, and Alexander Dixon. 2019. A high-speed key management method for quantum key distribution network. In *2019 11th International Conference on Ubiquitous and Future Networks (ICUFN'19)*. IEEE, 437–442.

[71] A. Tajima, T. Kondoh, T. Ochi, M. Fujiwara, K. Yoshino, H. Iizuka, T. Sakamoto, A. Tomita, E. Shimamura, S. Asami, et al. 2017. Quantum key distribution network for multiple applications. *Quantum Science and Technology* 2, 3 (2017), 034003.

[72] Masahide Sasaki, Mikio Fujiwara, H. Ishizuka, W. Klaus, K. Wakui, M. Takeoka, S. Miki, T. Yamashita, Z. Wang, A. Tanaka, et al. 2011. Field test of quantum key distribution in the Tokyo QKD Network. *Optics Express* 19, 11 (2011), 10387–10409.

[73] Piotr K. Tysowski, Xinhua Ling, Norbert Lütkenhaus, and Michele Mosca. 2018. The engineering of a scalable multisite communications system utilizing quantum key distribution (QKD). *Quantum Science and Technology* 3, 2 (2018), 024001.

[74] Yuan Cao, Yongli Zhao, Jianquan Wang, Xiaosong Yu, Zhangchao Ma, and Jie Zhang. 2019. SDQaaS: Software defined networking for quantum key distribution as a service. *Optics Express* 27, 5 (2019), 6892–6909.

[75] Yuan Cao, Yongli Zhao, Carlos Colman-Meixner, Xiaosong Yu, and Jie Zhang. 2017. Key on demand (KoD) for software-defined optical networks secured by quantum key distribution (QKD). *Optics Express* 25, 22 (2017), 26453–26467.

[76] Yuan Cao, Yongli Zhao, Rui Lin, Xiaosong Yu, Jie Zhang, and Jiajia Chen. 2019. Multi-tenant secret-key assignment over quantum key distribution networks. *Optics Express* 27, 3 (2019), 2544–2561.

[77] Liquan Chen, Qianye Chen, Mengnan Zhao, Jingqi Chen, Suhui Liu, and Yongli Zhao. 2022. DDKA-QKDN: Dynamic on-demand key allocation scheme for quantum Internet of Things secured by QKD network. *Entropy* 24, 2 (2022), 149.

[78] Kyu-Seok Shim, Yong-hwan Kim, Ilkwon Sohn, Eunjoo Lee, Kwang-il Bae, and Wonhyuk Lee. 2022. Design and validation of quantum key management system for construction of KREONET quantum cryptography communication. *Journal of Web Engineering* 21, 5 (2022), 1377–1418.

[79] Kyu-Seok Shim, Wonhyuk Lee, and Yong-Hwan Kim. 2022. A design of secure communication architecture applying quantum cryptography. *Journal of Information Science Theory & Practice (JIStaP)* 10 (2022), 123–134.

[80] Paul James, Stephan Laschet, Sebastian Ramacher, and Luca Torresetti. 2023. Key management systems for large-scale quantum key distribution networks. In *Proceedings of the 18th International Conference on Availability, Reliability and Security*. ACM, 1–9.

[81] marketintellix. 2024. Quantum Key Management Machine Market: Growth Trends and Forecasts. Retrieved December 14, 2024 from https://www.marketintellix.com/report/quantum-key-management-machine-market-278467. (2024).

[82] Miralem Mehic, Peppino Fazio, Stefan Rass, Oliver Maurhart, Momtchil Peev, Andreas Poppe, Jan Rozhon, Marcin Niemiec, and Miroslav Voznak. 2019. A novel approach to quality-of-service provisioning in trusted relay quantum key distribution networks. *IEEE/ACM Transactions on Networking* 28, 1 (2019), 168–181.

[83] European Commission. 2022. Recommendation for a Coordinated Implementation of a Roadmap on the Transition to Post-Quantum Cryptography. Retrieved December 14, 2024 from https://digital-strategy.ec.europa.eu/en/library/recommendation-coordinated-implementation-roadmap-transition-post-quantum-cryptography. (2022).

[84] Sonja Bruckner, Sebastian Ramacher, and Christoph Striecks. 2023. Muckle+: End-to-end hybrid authenticated key exchanges. In *International Conference on Post-Quantum Cryptography*. Springer, 601–633.

[85] Marc Geitz, Ronny Döring, and Ralf-Peter Braun. 2023. Hybrid QKD & PQC protocols implemented in the Berlin OpenQKD testbed. In *2023 8th International Conference on Frontiers of Signal Processing (ICFSP'23)*. IEEE, 69–74.

[86] Nicolas Gisin, Grégoire Ribordy, Wolfgang Tittel, and Hugo Zbinden. 2002. Quantum cryptography. *Reviews of Modern Physics* 74, 1 (2002), 145.

[87] Renato Renner. 2008. Security of quantum key distribution. *International Journal of Quantum Information* 6, 01 (2008), 1–127.

Received 28 April 2024; revised 19 March 2025; accepted 6 April 2025