JOURNAL OF
Optical Communications
and Networking

Decentralized key distribution versus on-demand relaying for QKD networks

MARÍA ÁLVAREZ ROA,[1,*] CATALINA STAN,[1] SEBASTIAN VERSCHOOR,[2] IDELFONSO TAFUR MONROY,[1] AND SIMON ROMMEL[1]

[1] *Department of Electrical Engineering, Eindhoven University of Technology, Eindhoven, The Netherlands*
[2] *Informatics Institute, University of Amsterdam, Amsterdam, The Netherlands*
**m.alvarez.roa@tue.nl*

Received 12 November 2024; revised 4 June 2025; accepted 22 June 2025; published 24 July 2025

Quantum key distribution (QKD) allows the distribution of secret keys for quantum-secure communication between two distant parties, vital in the quantum computing era in order to protect against quantum-enabled attackers. However, overcoming rate-distance limits in QKD and the establishment of quantum key distribution networks necessitate key relaying over trusted nodes. This process may be resource-intensive, consuming a substantial share of the scarce QKD key material to establish end-to-end secret keys. Hence, an efficient scheme for key relaying and the establishment of end-to-end key pools is essential for practical and extended quantum-secured networking. In this paper, we propose and compare two protocols for managing, storing, and distributing secret key material in QKD networks, addressing challenges such as the success rate of key requests, key consumption, and overhead resulting from relaying. We present an innovative, fully decentralized key distribution strategy as an alternative to the traditional hop-by-hop relaying via trusted nodes, where three experiments are considered to evaluate performance metrics under varying key demand. Our results show that the decentralized pre-flooding approach achieves higher success rates as application demands increase. This analysis highlights the strengths of each approach in enhancing QKD network performance, offering valuable insights for developing robust key distribution strategies in different scenarios.

https://doi.org/10.1364/JOCN.547793

1. INTRODUCTION

Quantum computing represents a paradigm shift in computational capabilities, introducing the potential to solve problems that are currently intractable for classical computers. One of the most significant implications of quantum computing is its ability to break widely used asymmetric cryptographic algorithms such as Diffie–Hellman (DH) and Rivest–Shamir–Adleman (RSA). These protocols are widely used for key distribution and base their security on the computational complexity of mathematical problems, which can be compromised by quantum computers capable of implementing Shor's algorithm [1]. While fully functional large-scale quantum computers capable of breaking current cryptographic systems may still be years away, sensitive data are still vulnerable. The secure communications can be collected, awaiting decryption through the potential power of quantum computing in the future. This "harvest now, decrypt later" strategy poses a serious risk, especially for data that require long-term confidentiality [2].

Quantum key distribution (QKD) enables the exchange of secret key material between two parties by providing information-theoretic security (ITS) based on the fundamental principles of quantum physics rather than using computational assumptions to establish computational security [3]. In addition, QKD allows for the detection of any eavesdropping attempts. By encoding keys into quantum states (such as the polarization of photons [3]), QKD systems can detect disturbances caused by eavesdroppers, ensuring that only the legitimate parties have access to the secure keys. Once the keys are securely exchanged, they can be used for arbitrary cryptographic purposes, including the one-time pad (OTP) for ITS encryption or the Advanced Encryption Standard (AES) for quantum-safe encryption [4,5]. This last approach leverages the strengths of both quantum key distribution and advanced cryptographic techniques, ensuring encryption that is resistant to both classical and quantum attacks.

One of the major challenges of QKD networks (QKDNs) is the key generation rate and distance limitation due to the degradation of the quantum signal over long distances through optical fibers [6]. At most, a few hundred kilometers can be reached and only between two directly connected endpoints. This rate-distance limit could be overcome by using quantum repeaters, but they are not available for practical implementation with current technologies [7]. Key relaying based on trusted nodes [8] is the most popular alternative approach to overcome rate and distance constraints and can be deployed using current technology. Larger QKD networks can be deployed using a hop-by-hop strategy, where keys are stored in QKD nodes (trusted nodes) and securely relayed to other distant QKD nodes not directly connected via a QKD link, through an uninterrupted series of intermediate nodes. During the key relaying process, secret key material is consumed along the relay path to establish end-to-end keys. Thus, key material generated by the QKD modules can be supplied to cryptographic applications or used for key relay when needed, making efficient key handling and consumption critical to the performance of the QKD network. However, the absence of standards for key relaying and key management systems (KMSs) worsens this problem. Commercial QKD modules incorporate proprietary KMS designs with unique vendor-specific interfaces and protocols for key relaying. These modules usually depend on on-demand key relaying, where QKD keys are transmitted as needed. As a result, key relaying and key management may become inefficient, impeding the scalability and interoperability required for larger and multi-vendor QKD networks.

In recent years, the optimization and management of QKD networks have garnered significant attention in the research community. The authors of [9] proposed an integer linear programming (ILP) scheduling algorithm for dynamic QKD link establishment in a switched QKD network to time-manage resources and optimize key generation and consumption rates across the network. A comparative analysis of various routing metrics using a centralized routing approach was conducted in [10] to evaluate their impact on the success rate of on-demand relaying in QKD networks. According to the results obtained, the paper recommends focusing on computationally inexpensive resource-aware routing algorithms. The authors of [11] evaluate and optimize existing routing schemes using a central network controller for demand-wise routing. Their ILP algorithm performs best but requires online knowledge of user demand and pool status, while other methods balance routing quality and management overhead. A differential-based proactive key relay that allows for calculating the key relay route by considering only the key status information is presented in [12]. The authors of [13] study the impact of static and dynamic relay weights on quantum key allocation with QoS constraints in a QKDN based on trusted relays, finding that memoryless dynamic weights reduce success rates, while static weights offer a reliable performance and user experience trade-off. A new deep reinforcement-learning (DRL) agent that routes encryption requests in a QKD network in a decentralized manner is proposed in [14], resulting in better performance than the decentralized variant of the Dijkstra algorithm or the classic hop-based algorithm.

This paper extends the work presented in [15], where the authors proposed for the first time two protocols for the management, storage, and distribution of key material in a QKD network. The solutions shown here have been simulated in the context of a QKDN with the aim of comparing results in terms of quantum key material consumption and the key request success rate—both key performance indicators for QKD networks. The first protocol uses on-demand relaying based on a hop-by-hop-strategy [16], while the second protocol adopts a decentralized pre-flooding strategy. Here, quantum keys are distributed to establish non-adjacent quantum key pools across the entire network in advance. By proactively pre-distributing key material between any pair of endpoints, this approach aims that end-to-end keys are readily available for delivery when requested by clients. This reduces delays in key delivery, which is particularly beneficial for applications with strict latency constraints. These protocols represent innovative approaches to managing secret key material within QKD networks, addressing key challenges such as latency-sensitive applications and efficient resource utilization. By simulating these solutions and comprehensively comparing their performance, we aim to enhance the design and implementation of QKD networks, ensuring robust and secure communication infrastructures for future applications.

By using estimated values for the secret key rate (SKR) of the decoy-state BB84 protocol based on [13,17], the simulation aims to closely model the performance of a QKDN. This approach ensures that results are applicable to real-world scenarios, providing an accurate prediction of how different strategies would perform. Moreover, for a comprehensive analysis of the network's performance under varying conditions, different simulation scenarios are considered, taking into account different rates of key consumption by applications. This allows for studying QKDN performance in "comfortable" circumstances and "under stress" resulting from high key consumption rates and insufficient resources to meet application demands.

The remainder of the article is structured as follows: Section 2 explains the QKD network model and SKR considered in this study and the key relaying protocols proposed. Section 3 explains the implementation and simulation scenarios. Section 4 analyzes the outcomes of the experiments and discusses the results. Finally, Section 5 draws relevant conclusions.

2. QKD NETWORK MODEL

QKD systems inherently face limitations in how far they can transmit secure keys, caused by losses experienced by the quantum channel due to the transmission medium, such that a longer distance implies a lower key generation rate due to the absorption and scattering of photons [18]. To address these challenges and allow quantum-safe long-distance communication, QKD networks have been developed. These networks aim to extend the reach of QKD systems beyond the constraints imposed by direct point-to-point transmission. The details of the QKDN architecture considered in this study, based on three different layers, are described below.

A. Network Architecture

The QKDN architecture shown in Fig. 1 consists of a quantum layer, a key management layer, and an application layer, according to the standardized architecture in [19]. In the quantum layer, each pair of QKD modules connected by a QKD link

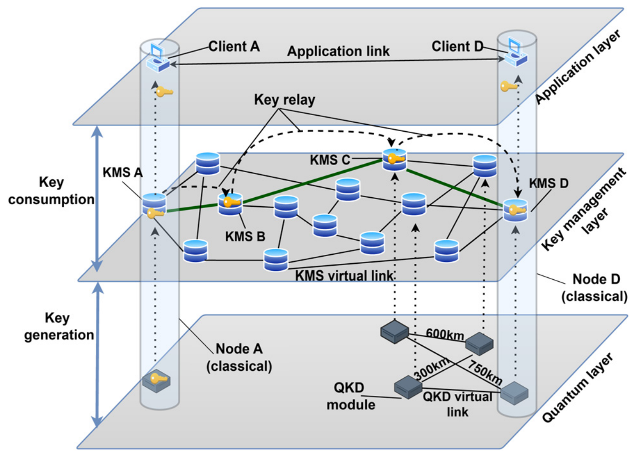

Fig. 1. Simplified QKD network architecture, showing quantum, key management, and application layers. The key management layer illustrates the NSFnet topology considered. For legibility, the application and quantum layers have been simplified, not showing all nodes.

generates and shares secure key material. The quantum links consist of an optical channel for transmitting quantum signals and a classical channel for exchanging synchronization information and key distillation [20]. Each QKD module pushes the generated key material to a KMS that is located in the key management layer in the same QKD trusted node. Note that nodes sharing the same horizontal coordinate across the layers are considered to be located in the same secure area.

The KMS is a central piece within a QKDN that allows QKD networks to be extended beyond point-to-point by establishing end-to-end keys with any node, which will subsequently be consumed by clients. Its main tasks include the storage and synchronization of key material from the QKD modules, the distribution of quantum keys across the QKDN, and the management of key requests from clients. KMSs are connected via a KMS link required to synchronize adjacent KMSs and key relaying. Every node in the network has one KMS, which can be connected to more than one co-located QKD module. Finally, clients (e.g., cryptographic applications) are located in the application layer, consuming quantum keys from the KMS via a standardized API. To model the QKDN, the classical NSFnet network topology [21], for simplicity only represented fully in the key management layer in Fig. 1, has been considered. The same classical network topology has been considered for the application layer, while that of the quantum layer is explained below.

B. Simulated Network Model

The actual length of the links in the classical network topology [21] ranges from 150 to 2400 km and significantly exceeds the useful transmission distance limit of point-to-point QKD. To convert a classical network topology to a QKDN, additional secure relay nodes are therefore required to span the physical distances. The distinction between classical nodes and trusted relay nodes is critical in network architecture. The distinction between the two is expressed as follows:

- Classical nodes are the nodes in the classical network topology, represented in Fig. 1, which have classical networking and host clients (e.g., cryptographic applications). These nodes serve two primary functions: they act as endpoints for secure key distribution and initiate secure communication sessions. Additionally, classical nodes may perform trusted relay functions when required to help establish end-to-end keys between other classical nodes for secure communication. Note that these nodes are security-wise just as trusted as the relay nodes are.
- Trusted relay nodes are used solely for quantum key relay and are not involved in hosting end-users or handling any direct application layer tasks. Their function is to act as intermediaries that facilitate the transmission of quantum keys over long distances by linking shorter QKD segments, ensuring the continuity of secure key transmission.

A critical parameter for a QKD network is the SKR of a QKD link, which measures the rate of information exchange between a pair of QKD modules. The SKR depends on fiber span parameters (length, attenuation) and device-specific factors (QKD protocol, pulse repetition rate, dark count). In this work, the protocol BB84 decoy state with a quantum channel wavelength of 1550 nm has been assumed. In addition, the SKR estimation and all relevant parameters considered are in accordance with the details provided in [17]. It is worth mentioning that these are estimations based on ideal conditions and do not account for temporal variations in the SKR that may arise in real-world deployments. Since the key relaying process requires sufficient key material for encryption and authentication, the balance between the rate of key material generation and its consumption for data encryption/decryption operations by clients significantly impacts network performance.

In order to generate a realistic simulation environment, we estimate SKR values between linked QKD devices based on the distances in the NSFnet topology. As distances between nodes in the topology exceed the useful transmission distance limit of point-to-point QKD links, intermediate relay nodes are required. Allowing for a 50% reduction in the key rate between an extrapolation of the linear region of the SKR versus loss and the actual SKR, the maximum useful distance between nodes is estimated to be 119 km, referred to as the drop distance [13]. The number of QKD spans between neighboring classical nodes is then given by the ceiling of the ratio between the link length and the drop distance. For simplicity, to ensure all QKD spans between two classical nodes exhibit similar behavior at both the quantum and key management layers, equidistant placement of intermediate nodes is assumed. This assumption guarantees the same SKR across all spans, simplifying the analysis and comparison of different network configurations. However, it is important to note that this is an idealized assumption, and in practical scenarios, factors such as node availability, geographical constraints, and network topology may affect the placement of intermediate nodes and, consequently, the SKR. Given that the SKR of any link in a real deployment will be determined by the constituent span with the smallest key rate, this simplification does not significantly alter the subsequent analysis.

In the network model under consideration, we differentiate between the two layers of key relaying that take place. First,

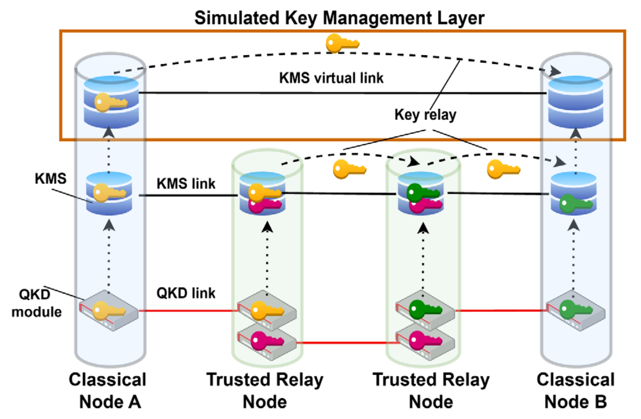

Fig. 2. Key relay scheme for the establishment of keys, shared via a QKD link, between adjacent QKD classical trusted nodes by means of QKD trusted relay nodes.

key relaying is necessary to guarantee the existence of the same keys between neighbor classical nodes that are separated by a distance greater than the maximum point-to-point useful distance for acceptable performance on a QKD link [18]. Figure 2 provides a detailed view of the existing structure in the quantum and key management layer between two classical nodes. This example shows how longer distances are bridged through trusted relay nodes, where nodes directly connected by a QKD link will share the same key material. Every time the QKD modules provide new key material to the KMS located in the same classical node, a relay hop-by-hop base process is carried out where keys are combined when they cross over to another KMS and are recovered by making use of an XOR operation. Since the SKR remains constant across all QKD spans between two classical nodes, and the key material consumption will be the same for each pool involved during the relay, the QKD intermediate trusted nodes do not impact network performance. As a result, these trusted nodes are excluded from the simulated network model, simplifying the topology as shown in the key management layer of Fig. 1, where KMSs connected by KMS virtual links are those considered in our network model. Second, key relaying is needed for the establishment of end-to-end keys between non-adjacent classical nodes to allow applications to request keys between arbitrary pairs of network nodes. The strategies presented in Section 2.C are intended to address key relay for the latter purpose.

C. Key Relaying Protocols

The trusted relay scheme typically used in existing QKD networks achieves the purpose of extending the distance between communicating parties by consuming additional key resources. However, the SKR for the generation of key material in quantum-secured communication is limited, requiring careful optimization of secret key material usage. Thus, the efficiency of the key relaying process is critical, as inefficient key consumption can lead to the depletion of available quantum keys in the key pools. A strategic approach to key management is essential to maximize the utility of generated keys and ensure effective management of key pools. This includes ensuring that there is always a sufficient reserve of keys for relaying purposes while also maintaining enough keys for immediate client use. In the context provided, we refer to a key designated for delivery to a client as a client key, while the key material utilized for tasks such as authenticating relay messages or encrypting client keys using a one-time path during relay operations is referred to as a link key. The two key relay and KMS architectures for managing, storing, and distributing key material in a QKDN compared in this study are described below and will be referred to as on-demand relaying and key flooding.

An example of a key relay for end-to-end key establishment between clients A and D is illustrated within the key management layer in Fig. 1, where client A is the sender. A quantum key, stored in KMS A and provided by a QKD module located in node A, is relayed through KMS B and KMS C via KMS links until it reaches the destination node, node D. Finally, the key is delivered to client D. Figure 3(a) shows in detail the process of relaying key material from the source KMS A to the destination KMS D within the context of an on-demand relaying solution. This diagram provides a visual representation of how the key material is transferred via two intermediate trusted nodes following the hop-by-hop strategy [16]. The process starts with a key request from a client, known as an initiator, indicating that key transmission will only occur when needed. The requested client key (or keys if multiple keys are requested) is relayed from one node to another, where a portion of the key material stored in the key pools, one link key for OTP encryption and one link key to compute an authentication tag for the relay message, is consumed at each node in order for the client key to reach its destination securely. Assuming only one client key has been requested, when a relay message reaches a trusted node, e.g., the message arriving at KMS C from KMS B, the authentication tag is verified and the client key, previously taken from the AB pool at KMS B, is decrypted using a link key previously established between KMS B and KMS C with which the key was encrypted at KMS B. Thus, the key used for encryption/decryption is taken from the BC pool on both nodes. After decrypting the client key, it is re-encrypted using a new link key specific to the current node and the next node in the relay path, in this example, a key shared by KMS C and KMS D and taken from the CD pool. The process of verification, decryption, re-encryption, and re-authentication is repeated at each subsequent node until the client key reaches its destination.

OTP encryption, denoted as $\oplus$, is used to protect client keys, and in each step, the entire message is authenticated using an ITS message authentication scheme, represented by a padlock. OTP encryption consumes one link key for each client key to be transmitted; in addition, one link key is consumed for each authenticated message. The main reason for using ITS MAC is to authenticate each transmitted message, preventing malicious parties from manipulating key transfer processes. Without authentication, an attacker could induce honest parties to transfer key bits to different pools, potentially causing the same key bits to exist in multiple locations. This could lead to double usage of key bits, compromising security. Note that the first authenticated relay message, between KMS A and KMS B in the example, only indicates the ID of the client key to be relayed, which itself does not need to be transmitted. Consequently, OTP encryption is not required for the first message. In Fig. 3(a), the messages containing the client key

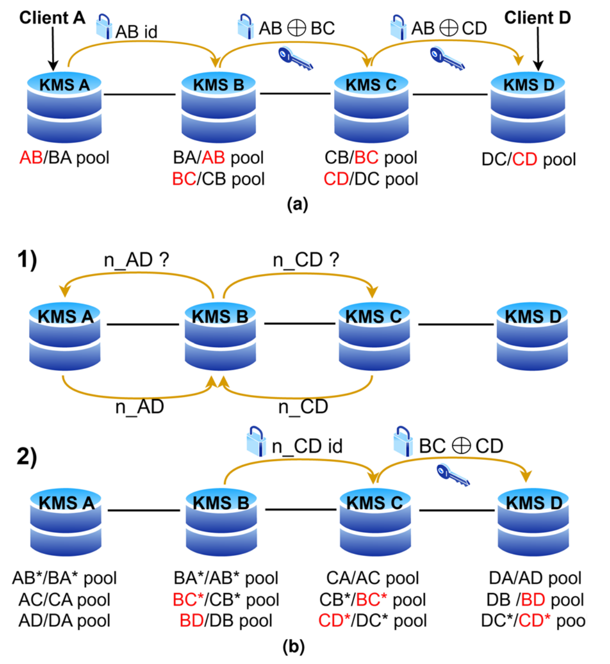

Fig. 3. Key distribution schemes for (a) on-demand relaying and (b) key flooding architectures. OTP encryption, denoted as ⊕, is used to protect client keys, and the padlock above a message indicates ITS message authentication.

that needs to be transmitted are indicated by a key symbol. To prevent the risk of keys being assigned twice due to synchronization issues, the key material shared between each pair of KMSs is divided into two parts, e.g., the key material generated between nodes A and B will be split in two and stored in the AB/BA pools. One half is designated for key relaying in one direction, while the other half is reserved for key relaying in the opposite direction. In the example, key pools from which keys were consumed during the relay process are shown in red.

A second solution, the key flooding strategy [15], is designed to minimize the service quality degradation of quantum-secured networking; it involves the proactive distribution of key material from adjacent key pools to non-adjacent key pools anywhere in the network in a decentralized manner. By establishing key pools between any pair of endpoints ahead of time, the network aims to establish pre-distributed keys that are ready to be delivered when they are requested by clients, thereby reducing delays in key delivery. Each node in the QKDN independently manages its key pool and coordinates with adjacent nodes to efficiently distribute keys across the network. This eliminates the need for a traditional routing scheme for relay path calculation.

To ensure that all key pools in the network are adequately filled, each QKD node periodically checks the status of its non-adjacent key pools to determine which ones need additional keys. Once the QKD node identifies the end-to-end key pools that require replenishment (the amount of key material is below a certain minimum), pools with a lower number of hops between nodes will be selected to be filled first. Figure 3(b) shows the process required to fill non-adjacent key pools in the QKDN, where key pools between adjacent nodes are represented by an asterisk (*) in the names of the pool. In the example, the pool to fill is BD with KMS B acting as the initiator and KMS D as the target. During the first step of the filling process, Fig. 3(a)(1), KMS B requests from each adjacent KMS the number of keys available to reach target node D and selects the one that can reach the target with the lowest number of hops, selecting KMS C in the example. The number of keys to be relayed is set to the minimum of the available keys reported, n_CD, and the keys available for relaying in the matching neighbor key pool, BC*. In the example, we assume the number of available keys reported, n_CD, is the minimum. In Fig. 3(b)(2), KMS B transfers n keys from the adjacent key pool BC* to the far-end key pool BD. KMS C, which performs the relaying, receives from KMS B the IDs of these n keys, which identify the keys to be transmitted to KMS D. Keys are taken from pool BC* and relayed to D, where they will be stored in pool BD. An ITS authentication scheme, which consumes link keys, is used over the entire message marked with a padlock. In this approach, the only message containing OTP-encrypted data, consuming also link keys, is the one that contains the key material. In this particular example, this message will be from KMS C to KMS D, where the keys used for encryption and authentication will be taken from the key pool CD*.

Both architectures are evaluated under simulated conditions to compare their performance. Key material consumption assesses the efficiency of key usage and management strategies. Additionally, the success rate of key requests by clients measures the reliability and responsiveness of each architecture in meeting communication needs.

3. QKD NETWORK SIMULATION DESIGN

A. Simulation Setup

The aim of the simulation carried out is to evaluate the performance and feasibility of QKD networks before hardware deployment in real networks. This allows operators to test and refine network configurations, assess potential performance metrics, and identify any issues or limitations in a controlled environment, improving the efficiency of the deployment process. To model the QKD network, the NSFnet topology [21], shown in the key management layer of Fig. 1, is employed as the framework. Mininet 2.3.0 [22] has been used to recreate the QKDN environment and evaluate the behavior of the two KMS architectures proposed. Each KMS operates as a virtual host within an emulated network topology, with an ethernet switch facilitating the necessary connectivity between the hosts. These hosts can execute any required application directly on the Mininet nodes, which run as processes on the host machine. Thus, each node in the simulation runs two instances of the Uvicorn web server, enabling the simultaneous hosting of two separate applications implementing the proposed architectures. The application programming interfaces were developed using FastAPI. The functionality of the KMS is fully implemented, including all necessary messaging between hosts to support key management operations. However, actual key storage is not implemented; instead, each key pool is represented as a counter tracking the availability of key material on each node. This setup facilitates the comparative analysis of different KMS strategies in terms of key management efficiency within the same network scenario. The virtual hosts in Mininet can be configured to connect to real networks and devices, but this is out of scope. Thus, in order to model all the layers of the network architecture presented in Fig. 1, the simulation uses asynchronous functions to periodically emulate the following tasks:

- Key requests: clients asynchronously request a certain amount of keys from the KMS. This function represents client communication in the application layer. The inter-arrival time of key requests from each client is exponentially distributed.
- Key material upload: QKD equipment generates key material and periodically uploads it at each node to replenish key pools with adjacent nodes.
- Key pool filling: in scenarios involving key flooding, asynchronous functions manage the distribution of keys to ensure that key pools across the network remain filled.

In addition, network messages, including key requests and key material exchanges, are implemented as HTTP requests, and all required code is written in Python. InfluxDB, a time-series database, is utilized for monitoring network activity and storing measurement data, which can be analyzed to understand the effectiveness of different KMS strategies.

NetworkX [23] is utilized to determine the shortest path for on-demand relaying by evaluating the number of hops between the source and destination. Minimizing the number of nodes is necessary because involving longer paths requires a higher consumption of key material. Moreover, key pool fill levels have been considered at the source node for the selection of the route when there are multiple shortest paths available between two nodes.

B. Simulation Scenarios

In analyzing the performance of QKD networks, a primary focus is on understanding how key material is consumed and the success rate of key requests from clients. This involves examining how efficiently the network generates, distributes, and utilizes quantum keys for client requests and during the key relaying process. Each key pool is implemented as a counter of key material to track the availability of keys on each node. This counter is incremented when new keys are received and decremented when keys are consumed or relayed. To facilitate this analysis, client and QKD protocols are emulated only with respect to key delivery and consumption.

Simulation parameters are included in Table 1, where some parameters were fixed throughout, while others were varied for evaluation. In terms of key consumption from clients, three experiments were performed. In the first scenario, the demand for cryptographic keys is moderate and within the network's capacity to generate and distribute keys. The network operates smoothly, with sufficient resources to meet the key requests from clients. The second and third scenarios simulate conditions where the key consumption rate is high, potentially exceeding the network's capacity to generate or distribute keys effectively. The network is expected to struggle to meet the demand due to the shortage of keys, reflected in a lower success rate of client key requests. To model the scenarios mentioned, the number of average requested bytes by clients has been

Table 1. Simulation Parameters

Parameter [Unit]	Values
Average inter-request interval [s]	2
Key size [bytes]	32
Inter-arrival time of key blocks [s]	40
Average bytes requested [bytes]	224/416/640
Max capacity [kbytes]	10/50/80/100
Client key reserve [bytes]	500/1000/1500/2500
Flooding trigger interval [s]	40/30/20

varied. In the first experiment mentioned above, an average of 224 bytes is requested, while for the second and third cases, the averages are 416 and 640 bytes, respectively. Considering a key size of 32 bytes, the average number of requested keys is 7 for the first case, 13 for the second, and 20 for the third. While we only vary the key request size and maintain a constant inter-arrival time of requests in these experiments, similar behavior is expected if the inter-arrival time is varied with a fixed request size, or if both factors are varied simultaneously.

The timing of key requests from clients is modeled using an exponential distribution, with an average inter-arrival interval of 2 s. This means that, on average, each node received a request on average every two seconds (based on stating that clients are simplified in a single request process, as suggested above), but the exact timing varies, creating a realistic pattern of network demand representing the random nature of request arrivals. In addition, the destination QKD node of each request is selected randomly with each node, excluding the source node, having an equal probability of being chosen (uniform probability).

In order to model the QKD protocol, the calculated SKR values have been considered, resulting in different SKR for each pair of QKD modules depending on the span length between intermediate trusted nodes on the relevant link. For this simulation, each QKD link in the network, at regular intervals of 40 s (inter-arrival time of key blocks), uploads the amount of key material generated according to the estimated SKR, to the corresponding KMS, where the key material is split equally over the two directed pools. Thus, the amount of keys added per interval in KMSs between adjacent nodes is proportional to the secret key rate between QKD modules connected by QKD links. This periodic key generation models the steady supply of keys produced by actual QKD systems, ensuring that the network consistently and often enough has new key material available for distribution and use, preventing the success rate from being affected by the key generation interval. Note that the QKD modules considered are asynchronous, we introduce an offset per link at the beginning of the simulation to start adding key material to the key pools.

Moreover, Table 1 summarizes other important parameters considered that directly affect the performance obtained. To simplify, we have assumed that all key pools in both strategies have the same maximum capacity, i.e., the maximum amount of bytes that can be stored in each key pool. Furthermore, regarding the key flooding strategy, the flooding trigger interval and the client key reserve have been varied in order to compare the impact caused on the metrics considered. The flooding trigger interval determines how often the non-adjacent key

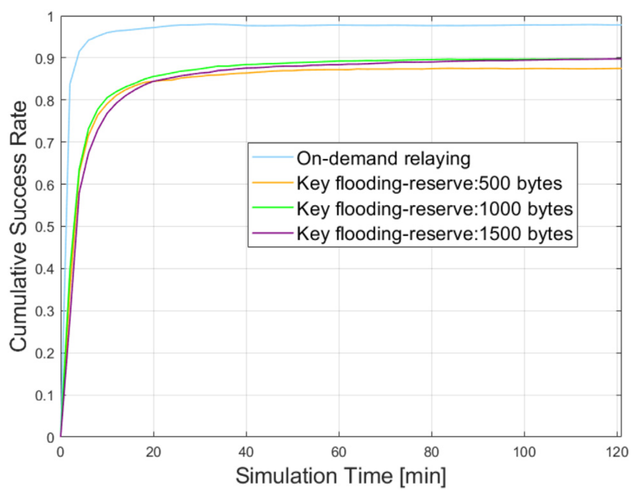

Fig. 4. Cumulative success rate of the network over the simulation time of on-demand relaying and key flooding protocols for different client key reserves when an average of 224 bytes is requested by clients.

pool is refilled. This process is triggered asynchronously across nodes, with the same frequency for each one. The client key reserve refers to the minimum target number of client keys allocated to each key pool. Once this minimum target is achieved, surplus keys can be used as link keys for OTP encryption and authentication for relaying.

In order to obtain a more accurate estimation of the system behavior by minimizing the influence of the random processes and improve the confidence of our results, we run several random simulations of two hours for each comparison type, where the data presented show the average of the results recorded. We ensured that the number of simulation runs was sufficient to reduce the standard deviation of the randomness to below 2%, guaranteeing that the variability introduced by random parameters is minimal.

4. PERFORMANCE EVALUATION RESULTS

The success rate is an essential metric indicating the proportion of key requests that are successfully fulfilled. A key request is successful if a client key is available to be allocated to the initiator, i.e., in the case of key flooding if a key is available in the end-to-end pool between the initiator and the target, and in the case of on-demand relaying if the hop-by-hop relaying is successfully completed. Figure 4 shows the variation of the cumulative success rate across the entire network during the simulation time for a specific case when clients ask for 224 bytes on average per request. Results for on-demand relaying and key flooding considering different client key reserves are shown. Starting the simulation with empty pools, we estimate that the simulation enters a stationary condition after approximately 30 min for all cases. Thus, to avoid distorted results from the fact that all key requests fail due to a shortage of link keys during the initialization of the QKD network, we discard the measured data corresponding to the first 30 min of the simulation for the calculation of the average success rate after this point.

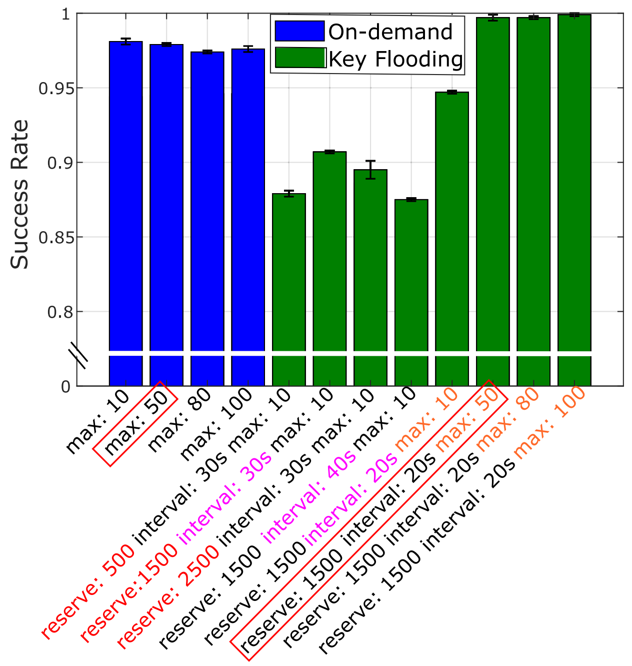

Fig. 5. Comparison of the overall success rate of on-demand relaying (blue) and key flooding (green) for different simulation scenarios when an average of 224 bytes is requested in each request.

Figure 5 compares the average value of the overall success rate across the entire network for the two proposed protocols and different scenarios where the parameters in Table 1 are used. In addition, the standard deviation is indicated with error bars. The network operates with client key requests averaging 224 bytes, allowing it to have sufficient resources to meet client demands effectively. Thus, the success rate values observed, based on the simulation, are in the range between 0.875 and 0.999, indicating that most links overproduce keys compared to the demand for keys from clients. As can be seen, increasing the maximum capacity of the pools does not improve the success rate for on-demand relaying (blue bars—variation less than 1%), while an increase from 10 to 50 kbytes shows a substantial improvement in the key flooding strategy (green bars). Larger maximum capacities allow storing more bytes that are used for the transmission of bigger blocks of key material to fill end-to-end pools, allowing the amount of keys in the pools to be sufficient to successfully handle client key requests, thus increasing the key request success rate. However, a further increase in maximum capacity yields little return. It should be noted that, for small maximum capacity values, on-demand relaying outperforms key flooding, regardless of the ratio or client key reserve considered. However, the increase in maximum capacity significantly improves performance in the case of key flooding.

Comparing the rest of the key flooding scenarios in Fig. 5, we can see that there is an optimum client key reserve value, 1500 bytes, which improves the success rate significantly. Enforcing a higher reserve, 2500 bytes, is associated with a lower success rate since the lack of link keys, due to the increase of reserved client keys, block the key distribution process across the QKD network. However, a lower reserve of 500 bytes will be insufficient to satisfy requests for keys from clients. Small

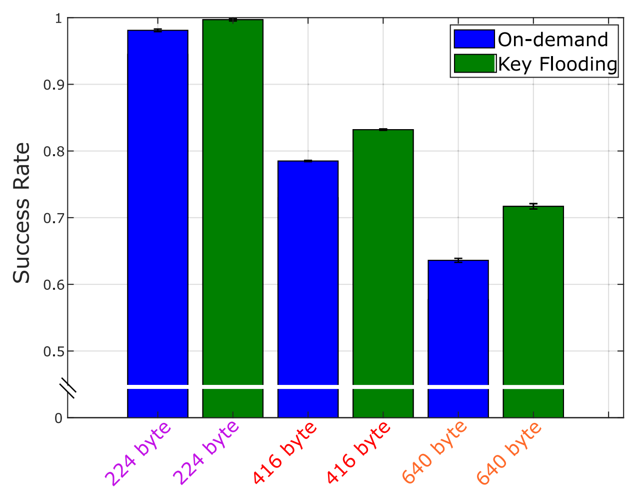

Fig. 6. Comparison of the overall success rate of on-demand relaying (blue) and key flooding (green) with a maximum pool capacity of 50 kbytes, a flooding trigger interval of 20 s, and a client key reserve of 1500 bytes when an average of 224, 416, and 640 bytes are requested in each request.

client key reserve will degrade the performance of the network due to the lack of client keys. By reserving fewer client keys, end-to-end pools will deplete more rapidly, and the frequency of key replenishment will not be sufficient to keep up with demand. As a result, it will not be possible to handle client key requests while waiting for the key-filling process to trigger. Therefore, reducing the flooding trigger interval can significantly improve network performance. Considering a reserve of 1500 bytes and a maximum capacity of 10 kbytes, Fig. 5 shows success rates of 0.875, 0.907, and 0.947 for intervals of 40 s, 30 s, and 20 s, respectively. By triggering the flooding of non-neighboring pools with keys more frequently, the approach aims to provide a more constant supply of key material across the network and thus reduces the likelihood of key depletion.

In the remainder, we discuss the results associated with the scenarios highlighted in Fig. 5. That is considering for both strategies a maximum pool capacity of 50 kbytes, a 1500 byte reserve of client keys, and a flooding trigger interval of 20 s for key flooding. Figure 6 shows the impact of increased key consumption on the success rate, where results assuming an average of 224, 416, and 640 bytes for each request are displayed. Success rates decrease as key request sizes—and hence key consumption—increase, most notably for on-demand relaying. When an average of 224 bytes is considered, we get a success rate of 0.981 and 0.997 for on-demand and key flooding, respectively. The overall success rate drops by 35% for on-demand relaying and 28% for key flooding when the average request size increases from 224 to 640 bytes, i.e., key consumption almost triples (increases by ∼285%). Notably, it can be observed that, when the network begins to struggle to meet demand due to key shortages, the routing strategy used to select the relay path for the on-demand relaying protocol has a high impact on the network performance, reducing the overall success rate.

Figure 7 shows the average success rate of key requests for (a) on-demand relaying and (b) key flooding for each QKD node across the QKD network topology when an average of

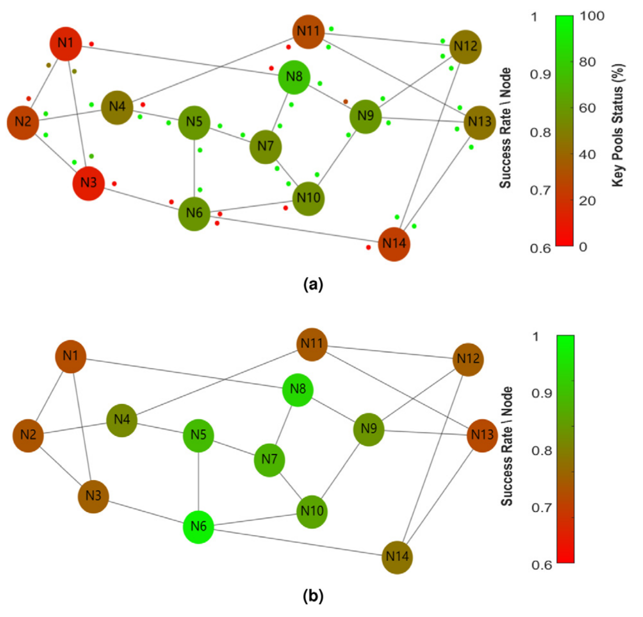

Fig. 7. QKD network topology based on a model of the NSFnet. Node colors show success rates for (a) the on-demand relaying strategy and (b) the key flooding strategy; small dots indicate neighboring key pool status.

416 bytes per request, a maximum pool capacity of 50 kbytes, a 1500 byte reserve of client keys, and a flooding trigger interval of 20 s for key flooding are considered. Node color represents the success rate, while the dots next to the nodes in Fig. 7(a) indicate the status of each key pool, i.e., the average key pool fill level in % of maximum capacity. Note that all key pools are initially in an empty state. In the case of key flooding, key pools at each node exist to all end-points of the topology; thus, the state is not shown in this case.

In Figs. 7(a) and 7(b), we can see that some specific nodes N1, N2, N3, N11, N13, and N14 show low success rate ($\leq$78%). For on-demand relaying, the need to use more transit nodes for key establishment between farther nodes makes the lack of key material during the relaying process more likely. For key flooding, due to the algorithm definition, end-to-end pools between nodes separated by a larger number of hops become more dependent on the successful distribution of key material in key pools of intermediate nodes, making the flooding process less efficient as the hop count increases. In contrast, nodes like N5, N6, N7, N8, and N9, visually located in the center of the network topology, with a lower number of hops on average with the rest of the nodes, show higher success rate results. In addition, Fig. 7(a) also shows that key usage on a link is not always symmetric (e.g., neighboring pools on N1–N2). This asymmetry arises due to the network topology and the path selection algorithm, where the selection of the relay route is based on the key pool status of the first relay hop when multiple shortest paths are available. As seen in Fig. 6, on-demand relaying shows a lower overall success rate than key flooding, which is also reflected in the average success rate of each node. On-demand relaying results show low availability of key material in certain key pools, indicating increased relay message traffic between the QKD nodes. This leads to

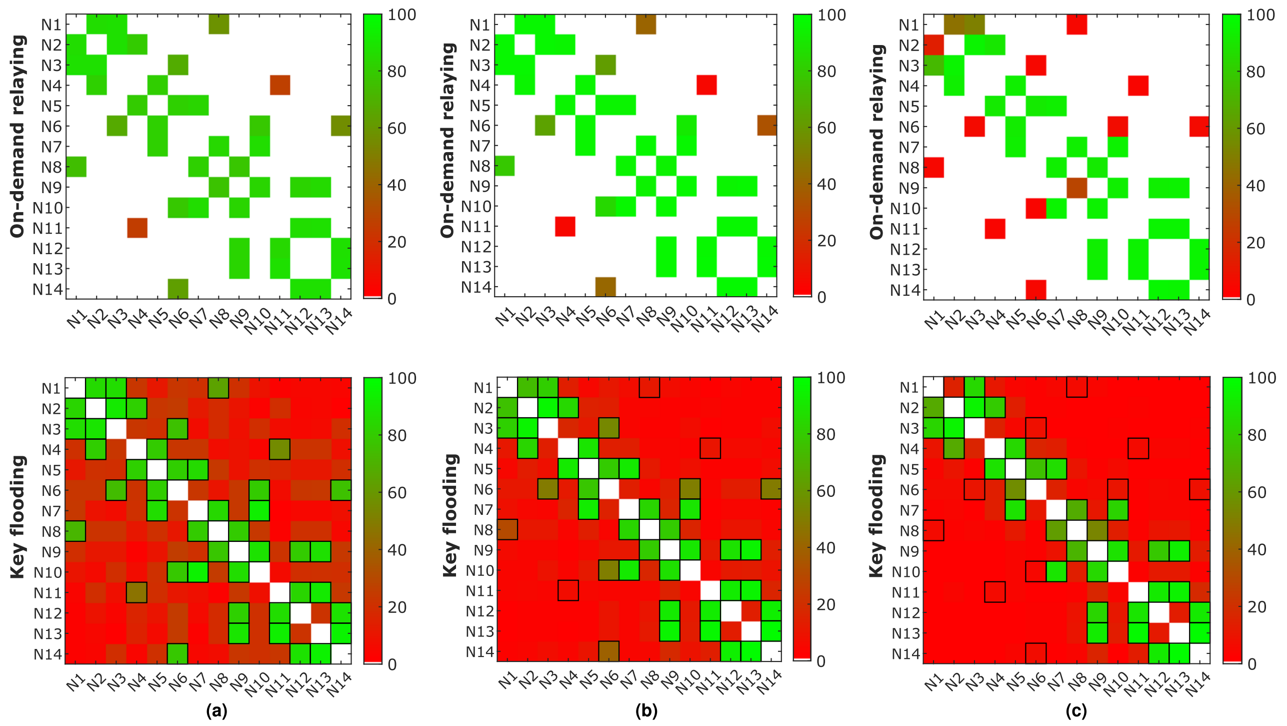

Fig. 8. Average key pool fill level of on-demand relaying and key flooding protocols for different average bytes requested and different maximum pool capacities: (a) 224 bytes and 10 kbytes, (b) 224 bytes and 50 kbytes, and (c) 416 bytes and 50 kbytes.

significant variations in key material consumption across different links. The shortage of key material in some pools could lead to a higher number of unsuccessful key requests, as the system may be unable to complete key relays.

To optimize QKD deployment, particularly on the most utilized links with the highest traffic load, it is essential to have a comprehensive understanding of the status of each key pool. Thus, Fig. 8 shows the average key pool level of the two proposed protocols for three scenarios where different values of the average bytes requested and maximum pool capacity have been considered. The blank squares in the case of on-demand relaying indicate that the corresponding key pool does not exist, as the respective nodes are not neighbors.

The key pool status shown in Fig. 8(c) associated with the key flooding protocol extends the information included in Fig. 7(b), where key pools exist between any pair of nodes in the network. The quantity of key material available in the key pools associated with directly linked nodes, which are periodically replenished by the QKD modules, tends to be significantly higher compared to the key pools between non-adjacent nodes. Key pools between adjacent nodes, pools existent in both relay approaches, are highlighted with a black square. The higher the number of hops between nodes, the more challenging it becomes to maintain adequate key levels, which may lead to key shortages and thus increase the number of failed key requests. When the demand for keys from clients is increased, Fig. 8(c), we observe an increase in the number of key pools in red, indicating that the amount of resources is below 20% of the total capacity.

Key material consumption at each node varies between the two different strategies due to differences in how routing decisions are made. The on-demand approach dynamically selects the shortest route to relay end-to-end keys, minimizing hops and considering key pool fill levels at the source node when multiple paths exist. In contrast, key flooding selects the neighbor node used for filling of the pools based on the key availability and proximity, in terms of hops, to the target KMS that needs to be replenished with keys. Thus, by examining the status of each key pool, network operators can identify potential bottlenecks where key pools become depleted, and thus, anticipate and mitigate associated key shortages. For on-demand relaying, operators can adjust the routing of key relays by checking the status of each key pool to avoid overloading specific nodes and distribute the relay burden more evenly across the network. For key flooding, operators can optimize the frequency of key floods and client key reserve values, aiming that nodes with the highest demand receive enough key material. Additionally, operators can monitor the fill levels and prioritize key replenishment of heavily used pools to ensure they remain operational even under high demand. The addition of a greater number of intermediate trusted relay nodes between classical nodes would increase the SKR and is thus a possible solution to increase the amount of key material added to each pool between neighbor classical nodes.

When comparing the number of hops per client key request, we consider the number of QKD links that consume key bits for serving a key request or, in the case of flooding, to establish an end-to-end pool. Based on the underlying network topology, the mean number of hops per client key request in on-demand relaying is 2.14 hops, which is minimal because it will always take the shortest path by definition. The mean number of hops to fill end-to-end pools for key flooding is 2.28

hops, which is slightly higher because flooding might take a detour when pools on the shortest path are exhausted.

Regarding the overall average key consumption from the time the simulation enters in stationary condition until the end of the simulation period, the on-demand protocol consumes ∼7% more key material compared to the key flooding relaying. In this context, the overall average success rates are 0.981 and 0.947 for on-demand relaying and key flooding, respectively. The simulation parameters considered are a maximum capacity of 10 kbytes, a flooding trigger interval of 20 s, and a client key reserve of 1500 bytes. The average request size is 224 bytes. Note that, for the on-demand approach, the consumption of key material does not guarantee the establishment of an end-to-end key since the lack of resources in the intermediate nodes of the relay path may cause the distribution process not to be completed, and thus, key material consumed in the relay process up to that point will be wasted. If we subtract the amount of key material consumed during the relay process that later turned out not to be able to be completed (1%) from the total consumption, we obtain that on-demand relay consumes approximately 6% more resources than key flooding for the same scenario considered above. For on-demand relaying each relay message at every hop along the relay path has to be authenticated resulting in higher consumption of link keys. For key flooding, since larger blocks of key material can be relayed to fill end-to-end pools, fewer relay messages are required, consuming less link keys for authentication. Thus, the consumption of keys for authentication purposes for on-demand is almost three times larger (larger by ∼266%) when the success rate is similar for both approaches.

When a maximum capacity of 50 kbytes is considered, the key flooding solution increases the key consumption; however, it is still lower than the total consumption in the on-demand approach by approximately 3% (2.01% without considering consumed bytes in uncompleted relay processes in on-demand). The resulting impact on the overall success rate is notable, with success rates of 0.997 for key flooding and 0.979 for on-demand relaying. If we calculate the ratio between the overall key consumption (client and link keys) and the number of successful client key requests for the scenario mentioned above, we obtain that the on-demand approach consumes on average 8.9% more key material to successfully assign a client key. When we increase the client key demand to 640 bytes on average, we observe that on-demand consumes in total 26.19% more key material, with approximately 21% of the key material being consumed by unsuccessful key requests.

A resource-aware routing for on-demand relaying would reduce overall key consumption by preventing key material from being consumed due to uncompleted relays, caused by insufficient keys at the intermediate nodes along the relay path [10,13]. However, a routing strategy that is not aware of the availability of keys in the relay path has been considered, causing a higher total consumption of key material. Key availability has been checked at the first relay hop when multiple shortest paths exist. In summary, key flooding avoids the loss of key material as observed for unsuccessful relays in the on-demand protocol and reduces the overhead for authentication tags of relay messages, hence successfully avoiding waste of key material.

Although the simulation does not accurately measure the latency between a client's key request and the delivery of the requested key, it can be stated that the average response time is faster with the key flooding approach compared to on-demand relaying, as key distribution takes place ahead of time without waiting for the arrival of key requests. However, the key flooding scheme necessitates the creation and maintenance of all possible end-to-end key pools, which introduces a substantial burden in terms of memory usage and coordination requirements. As the number of nodes in the network increases, the number of required key pools grows quadratically, meaning the memory and processing demands increase rapidly as the network expands.

5. CONCLUSION AND FUTURE WORKS

In this work, we compare and analyze two key relaying and KMS protocols for managing, storing, and distributing key material in a QKD network considering realistic SKR values. A fully decentralized key distribution strategy is proposed as an alternative to on-demand hop-by-hop relaying. By establishing end-to-end key pools in advance, the key flooding approach reduces the latency associated with key request processing. The absence of a central controller simplifies implementation and eliminates a single point of failure. While potentially less efficient under normal conditions, this design enhances the network's resilience under attack. However, a substantial memory and coordination cost is associated with the establishment of all possible end-to-end pools. Overall success rates in the network show higher values for key flooding when provided key pools are sufficiently large. Moreover, better success rate results are shown for this strategy compared to on-demand relaying when increasing client key consumption. The simulations and performance analysis results enable proactive adjustments to the QKD system for the proposed classical network topology, such as redistributing resources to more critical links. However, there are still open research questions and challenges that require further investigation.

Results presented here show that relay paths based on hop count may concentrate traffic through specific nodes or regions of the network, leading to an uneven consumption of key material. As these pools become depleted, the system faces a growing risk of being unable to fulfill key requests, resulting in a higher number of unsuccessful attempts to establish secure connections. To address these issues, it is essential to refine the routing schemes employed within the QKDN. One approach could involve the development of more adaptive routing algorithms [10,13] that consider the real-time status of key pools or other relay weights chosen from the network topology directly [13].

Furthermore, it is crucial to study the behavior of both protocols in various network topologies and to analyze how the solutions scale with network size. Evaluating the performance of these protocols in diverse configurations will help identify potential bottlenecks and inefficiencies that may arise as the network expands. This work provides a valuable tool for deployment planning and performance prediction, offering the ability to model and anticipate network behavior. By providing a detailed analysis, this tool helps optimize QKD network design, making it more efficient and scalable while avoiding unexpected behavior during deployment.

Funding. Ministerie van Economische Zaken en Klimaat (Quantum Delta NL); DIGITAL Europe Programme (101091656); H2020 Marie Skłodowska-Curie Actions (953442).

Acknowledgment. The authors would like to thank SURF, especially dr.ir. R.C.J. Smets, D. Maier, and dr. T. Strömberg for their collaboration.

REFERENCES

1. P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," SIAM J. Comput. **26**, 1484–1509 (1997).
2. N. Hassan and R. Hijazi, *Digital Privacy and Security Using Windows* (Apress, 2017).
3. C. H. Bennett and G. Brassard, "Quantum cryptography: public key distribution and coin tossing," Theor. Comput. Sci. **560**, 7–11 (2014).
4. C. Stan, C. Rubio García, B. Cimoli, *et al.*, "Securing communication with quantum key distribution: implications and impact on network performance," in *Advanced Photonics Congress* (Optica Publishing Group, 2022), paper SpW2J.2.
5. C. Rubio García, S. Rommel, S. Takarabt, *et al.*, "Quantum-resistant transport layer security," Comput. Commun. **213**, 345–358 (2024).
6. M. Dianati, R. Alléaume, M. Gagnaire, *et al.*, "Architecture and protocols of the future European quantum key distribution network," Secur. Commun. Netw. **1**, 57–74 (2008).
7. H.-J. Briegel, W. Dür, J. I. Cirac, *et al.*, "Quantum repeaters: the role of imperfect local operations in quantum communication," Phys. Rev. Lett. **81**, 5932–5935 (1998).
8. C. Elliott, "Building the quantum network*," New J. Phys. **4**, 46 (2002).
9. K. Christodoulopoulos, N. Makris, G. T. Kanellos, *et al.*, "Optimizing key consumption in switched QKD networks," in *Optical Fiber Communication Conference (OFC)* (Optica Publishing Group, 2024), paper W2A.34.
10. T. van Duijn, S. Verschoor, S. Rommel, *et al.*, "Routing strategies for quantum key distribution networks based on trusted relay nodes," in *Optical Network Design and Modeling Conference* (IEEE, 2024).
11. T. Johann, M. Wenning, D. Giemsa, *et al.*, "Comparison and optimization of different routing methods for meshed QKD networks using trusted nodes," J. Opt. Commun. Netw. **16**, 382–391 (2024).
12. C. Lee, Y. Kim, K. Shim, *et al.*, "Key-count differential-based proactive key relay algorithm for scalable quantum-secured networking," J. Opt. Commun. Netw. **15**, 282–293 (2023).
13. C. Stan, D. Verchere, J. J. Vegas Olmos, *et al.*, "Resource allocation strategies in quantum key distribution networks," in *IEEE Global Communications Conference* (2024).
14. T. Johann, S. Kühl, and S. Pachnicke, "Deep reinforcement learning based decentralized routing and load-balancing in meshed QKD-networks," in *50th European Conference on Optical Communications (ECOC)*, Frankfurt, Germany, 2024.
15. M. Álvarez Roa, S. Verschoor, and S. Rommel, "Efficiency analysis of two key relaying architectures for QKD networks," in *50th European Conference on Optical Communications (ECOC)*, Frankfurt, Germany, 2024.
16. M. Peev, C. Pacher, R. Alléaume, *et al.*, "The SECOQC quantum key distribution network in Vienna," New J. Phys. **11**, 075001 (2009).
17. C. Stan, D. Verchere, J. J. V. Olmos, *et al.*, "Dynamic-threshold-based pre-relaying for enhanced key allocation in quantum-secured networks," J. Opt. Commun. Netw. **17**, 233–248 (2025).
18. S. Pirandola, R. Laurenza, C. Ottaviani, *et al.*, "Fundamental limits of repeaterless quantum communications," Nat. Commun. **8**, 15043 (2017).
19. "Quantum key distribution networks–Key management," ITU-T Standard Y.3803 (2020).
20. G. van Assche, *Quantum Cryptography and Secret-Key Distillation* (Cambridge University, 2006).
21. X. Chen, B. Li, R. Proietti, *et al.*, "DeepRMSA: a deep reinforcement learning framework for routing, modulation and spectrum assignment in elastic optical networks," J. Lightwave Technol. **37**, 4155–4163 (2019).
22. "Mininet," http://mininet.org/.
23. A. Hagberg, P. Swart, and D. Chult, "Exploring network structure, dynamics, and function using NetworkX" (2008).