Routing Strategies for Quantum Key Distribution Networks based on Trusted Relay Nodes

Tommy van Duijn, Sebastian Verschoor, Simon Rommel, Idelfonso Tafur Monroy
Department of Electrical Engineering
Eindhoven University of Technology
Eindhoven, Netherlands
{t.c.j.v.duijn,s.r.verschoor,s.rommel,i.tafur.monroy}@tue.nl

***Abstract*—Quantum key distribution (QKD) networks enable connected parties to exchange cryptographic keys that are information-theoretically secure. In absence of quantum repeaters, key relaying between trusted nodes is the foremost method to overcome fundamental rate-distance limitations of point-to-point QKD links. This method requires consumption of keys generated by each independent QKD link in the relay path, but these resources can be depleted, resulting in failure. Hence, selecting suitable relay paths is essential for enabling successful key exchanges between each pair of communicating parties. This paper presents how the success rate of key relaying can be increased by frequently recomputing relay paths considering the availability of keys, i.e., resource-aware routing. A centralized routing approach is adopted for the key relay process, which is emulated to measure and compare the success rates achieved by different routing algorithms. It was found that using centralized resource-aware routing can increase the success rate up to 13.4 percentage point compared to selecting the shortest path without taking the availability of link keys into account. However, the differences amongst the success rates achieved by different resource-aware routing algorithms are statistically insignificant, if the centralized routing approach described in this paper is used. Therefore, this paper recommends to focus on resource-aware routing algorithms that are computationally inexpensive and suitable for large-scale implementation.**

***Index Terms*—Key relaying, network emulation, quantum key distribution, routing algorithms**

I. Introduction

The development of quantum computers threatens the security of public-key cryptography, which is widely used to enable secure communications over the internet [1]. Much research effort is being put into the development of post-quantum cryptography (PQC) that aims to be resistant against attacks using quantum computers [2]. Still, PQC depends on computational assumptions and future cryptanalysis might compromise the security it provides today.

Quantum key distribution (QKD) offers a promising method to exchange keys that complements PQC by providing information-theoretic security based on the fundamental laws of quantum physics instead of computational assumptions. QKD enables two parties to expand pre-shared keys by encoding classical bits into quantum states that can be transmitted over optical fibers or free space [3]. QKD equipment is already commercially available, but inevitable attenuation in the transmission medium imposes fundamental limitations on the maximum transmission distance and key generation rate (KGR) [4]. Briegel et al. proposed quantum repeaters to overcome rate-distance limitations by means of entanglement swapping, but this method cannot be used in practice yet because some essential technologies, like quantum memories, are still in development [5].

Key relaying is an alternative method to overcome rate-distance limitations and this method can be implemented with existing technology. Essentially, independent QKD links form a network that enables secure hop-by-hop transport of keys between any pair of nodes in the network, even when a pair of nodes is not connected by a direct QKD link.

The selection of a suitable relay path is similar to routing in classical networks, but an important difference is the fact that the status of each link in the QKD network depends on the local availability of keys. Key relaying consumes keys for every hop and is only possible with low latency if keys of the appropriate size are readily available in all nodes in the relay path. A shortage of keys results in failure, like packets being dropped in classical networks in case of congestion, and it might take an extended period of time before new keys are available. In case of on-demand relaying, the selection of relay paths directly influences the success rate of key requests by clients, which is one of the key performance indicators of QKD networks.

Although existing routing protocols and algorithms based on the shortest path can be used for QKD networks with minor modifications, this can lead to a shortage of keys in one node while QKD links connecting to other nodes are under-utilised [6]. To ensure service quality, classical networks usually have more capacity than strictly necessary. Considering the fact that QKD equipment is more expensive than classical network equipment at present, it is economically interesting for potential QKD network operators to research how a high success rate can be achieved without excessive costs resulting from such overprovisioning.

Resource-aware routing for QKD networks can increase the success rate of key requests by means of load balancing. In the context of this paper, resource-aware routing means frequently recomputing relay paths considering the availability of keys

This work was supported by the the Dutch Ministry of Economic Affairs and Climate Policy (EZK) as part of the Quantum Delta NL programme and the EU's Digital Europe Programme under project QCINed (grant no. 101091656) within the EuroQCI initiative.

978-3-903176-62-1 © 2024 IFIP

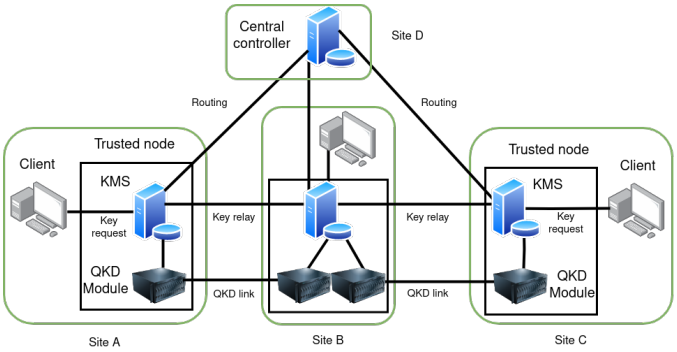

Fig. 1. Model of the QKD network. Adjacent nodes exchange key material using pairs of QKD modules that are connected by a QKD link. The quantum channel and classical channel between QKD modules are shown as one link. The central controller returns a suitable relay path upon request by a KMS.

per QKD link. Tanizawa et al. [7] proposed an adapted version of Open Shortest Path First Version 2 (OSPF-v2) [8] and achieved load balancing by sending a link state update with an increased link cost when keys are depleted. Yang et al. [9] improved load balancing by considering the number of available keys and disregarding paths with insufficient keys, but the number of available keys fluctuates which makes it challenging to keep the routing information up to date on all nodes. In contrast to the aforementioned distributed routing protocols, Chen et al. proposed to collect all routing information in a central controller that selects a suitable relay path for each individual relay request [10]. The proposal of recomputing relay paths for every individual key request raises questions about the scalability of this approach. Increasingly complex protocols and algorithms for selecting relay paths and load balancing are being proposed and potential QKD network operators will have to make a trade-off between possible benefits of resource-aware routing, computational complexity, and operational implications. Yet, it is not clear which routing protocol achieves the highest success rate because a different QKD network topology and KGR were used in each of the discussed papers and a fair comparison is lacking.

This paper aims to quantify advantages of resource-aware routing over selecting the path with the least number of hops, in terms of the achieved success rate of on-demand relaying. Existing proposals for routing metrics from related works are compared by means of emulation and a centralized routing approach. The next section explains the QKD network model, security assumptions, and key relay procedure. Section III explains the implementation and methodology of the experiments. Section IV analyzes the outcomes of the experiments and discusses the results.

II. QKD Network Model

A. Functionality of QKD nodes

The QKD network model is illustrated in Fig. 1. Every node in the QKD network has one key management server (KMS) and at least one QKD module. Every pair of QKD modules connected by a QKD link continuously runs a QKD protocol and each QKD module periodically delivers bytes to the local KMS, i.e., the KMS located at the same site as the QKD module. The output of the QKD modules will be referred to as key material in the remainder of this paper. The KMS buffers key material for key relaying and handles key requests by clients. To avoid confusion about terminology, a key to be delivered to clients is called a client key and key material used for authenticating a relay message or one-time pad (OTP) encryption of a client key is called a link key.

The KMS should be compatible with QKD equipment from all vendors to avoid vendor lock-in and allow for an hardware-independent network design. In absence of standards for key relaying (standardization ongoing), commercially available QKD modules often have an integrated KMS that works with proprietary interfaces and vendor-specific protocols. Therefore, key relaying between QKD equipment of different vendors is currently not implemented and potential QKD network operators are limited to the software features provided by each vendor. This paper assumes that the QKD network operator deploys its own KMS to enable the implementation of advanced routing protocols that are hardware-independent.

Each KMS sends information about QKD links, such as availability of keys and changes in the QKD network topology, to the central controller that selects a suitable relay path upon request by a KMS. This centralized approach obsoletes the exchange of routing information between nodes.

B. Security assumptions

QKD modules ensure the confidentiality and authenticity of key material exchanged between a pair of QKD modules. Still, a combination of physical and digital security measures is necessary for secure delivery of client keys by QKD networks. For example, adversaries must not be able to gain information of key material through (unauthorized) access to QKD equipment or the KMS. Adversaries must also not be able to eavesdrop on the communication between the client and KMS that is not protected by QKD. In this paper, attacks on nodes are out of scope and it is assumed that communication between devices within the same site is secure.

Ideally, the communication between the central controller and KMS would be information-theoretically secure (ITS). However, it is assumed that PQC provides integrity and mutual authentication for communication between the central controller and KMS. This is acceptable because the central controller and KMS do not exchange long-term secrets.

C. Key relay procedure

The sequence diagram in Fig. 2 shows the interactions between different entities that are required for a successful key exchange between two clients that do not belong to adjacent nodes. Clients request keys from their local KMS using the ETSI GS QKD 014 protocol [11]. The client that initiates the key exchange between a pair of clients will be referred to as the initiator and connects to the source KMS. The responder connects to the destination KMS. The key relay procedure allows the source and destination KMSs to exchange keys in a secure way, such that both clients in the pair can obtain a

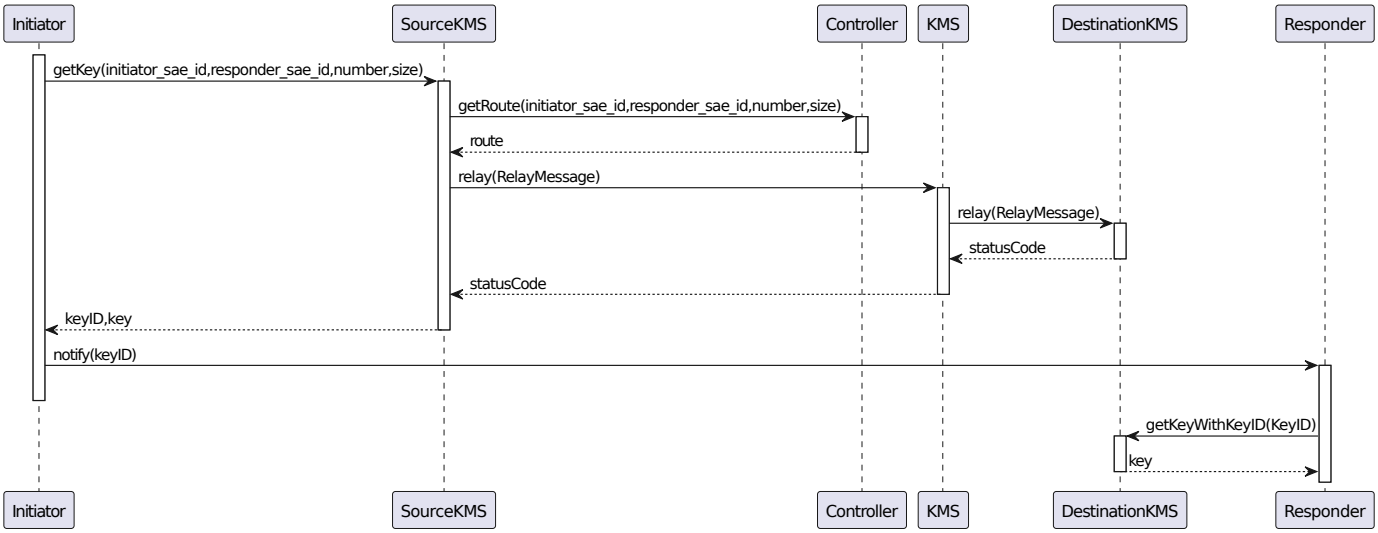

Fig. 2. Sequence diagram of interactions between different entities that are required for a successful key exchange between two clients. The keys in the relay message are protected with OTP encryption and the entire relay message is authenticated using an ITS message authentication code. The relay message is processed by each KMS and sent to the next KMS in the relay path until the destination KMS is reached.

shared key. It is assumed that the central controller is able to associate clients with their KMS. For the sake of completeness, the end-to-end exchange of key IDs between clients is included in Fig. 2, but details of how this should be implemented in the application protocol are out of scope.

III. Implementation and Methodology

A. Containernet

The work is based on the Containernet project [12]. All QKD modules, servers and clients were implemented using Docker containers that served as hosts in an emulated network topology. This approach allowed for flexible testing of software on a single computer without placing equipment in the field.

Each QKD module, KMS and the central controller hosted a web application in its own container using Uvicorn to enable key requests and key relaying. The application programming interfaces were developed using FastAPI and all required code was written in Python. For testing purposes and extraction of measurement data, all containers had a shared drive to write log files that could be read by the host computer.

Relay messages were implemented as HTTPS requests. The body of the request contained OTP encrypted client keys and metadata, such as a list of all hops in the relay path selected by the central controller. All information in the body of the request was authenticated using Poly1305 [13], which is an ITS message authentication code (MAC) if used properly like in this research. The use of OTP encryption and an ITS MAC required consumption of two link keys per hop for each relay message and limits the success rate of key requests compared to when an computationally secure MAC would be used. However, the required key size does not increase linearly with the size of the client key, unlike the link keys for OTP encryption. In fact, the required size of the link key for authenticating the relay message is approximately constant for such relatively small amounts of data. Hence, it is efficient to relay large keys, or equivalently, multiple keys in a single relay message.

B. Classical network

The classical network emulation was kept as simple as possible because only basic connectivity was required. Every KMS and the central controller were connected to a single switch that enabled the necessary connectivity. This avoided the necessity of including routers in the classical network emulation and reduced the computational overhead. There was no specific limit applied to the capacity of classical links in Containernet because it is expected that the capacity of classical channels will not limit the performance of QKD networks. The round-trip time of network packets sent over the internet (delay) was set to 10 ms in Containernet to account for transmission distances.

C. QKD modules and links

Because the focus of this research was on the selection of relay paths and not on the physics of QKD, the actual exchange of quantum states and related post-processing were not emulated. Instead, QKD links were simulated by periodically sending blocks of multiple 256 bit keys and corresponding key IDs from QKD transmitter modules to corresponding QKD receiver modules in plaintext over dedicated classical channels.

A model of the USNET topology depicted in Fig. 3 was used for the experiments. In reality the distances between the shown nodes are large and additional relay nodes would be required for implementation, but this is not relevant for routing since such nodes do not introduce alternative paths. Therefore, the shown links can be considered a concatenation of multiple relay links where the overall KGR is limited by the slowest link in the chain. Each simulated QKD link was assigned a fixed KGR between 500 bit/s and 2000 bit/s. These values were integers sampled from a discrete uniform distribution and

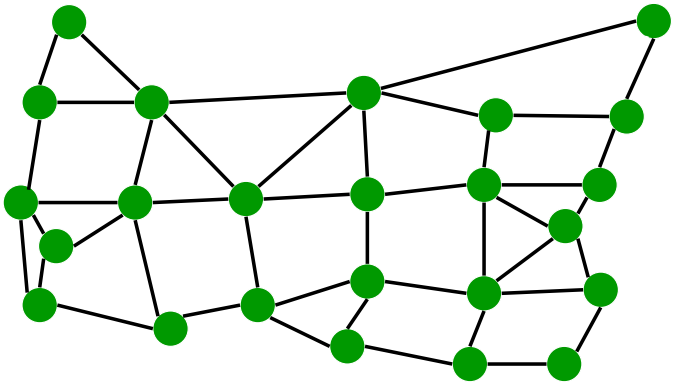

Fig. 3. QKD network topology based on a model of the USNET.

constant throughout all experiments. The block size was fixed at 10^5 bit, meaning that each pair of QKD modules generated approximately 390 link keys with a length of 256 bit after each key generation interval. Although larger block sizes may be used in practice, the block size of 10^5 bit kept the key generation interval short, allowing for experiments of limited duration.

D. Key management servers

Every KMS periodically pulled keys from its local QKD module(s) according to ETSI GS QKD 014. To prevent that keys could be allocated twice due to synchronization issues, each set of keys shared between a pair of KMSs was split in half. One half was used for key relaying in one direction and the other half for key relaying in the other direction. After the KMS pulled the keys from the QKD module, the KMS sent a link state update to the central controller. This link state update included the total number of available link keys for this specific QKD link and the KGR.

E. Central controller

The central controller used the NetworkX library [14] to create a directed graph of all QKD nodes and links based on received link state updates. Upon request by a KMS, the central controller selected a suitable relay path to the destination KMS. Different methods for selecting relay paths, referred to as routing strategies, were implemented as explained below.

- Shortest Path First (SPF). This strategy is the simplest strategy and always selects the path with the least hops using Dijkstra's algorithm. This means all link cost are constant and simply equal to 1.
- SPF+. Similar to SPF but excludes QKD links with insufficient link keys from the graph before the shortest path is selected.
- ADA-KGR [10]. Using this strategy, the link cost of each QKD link is inversely proportional to its average KGR, i.e., $C_{\text{link}} = 1/KGR$. In contrast to SPF(+), the path cost is not simply the sum of all link costs in the relay path but includes an additional component based on the extra number of hops compared to the path with the least number of hops. The total path cost is given by

$$C_{\text{path}} = \frac{1}{3}(N - N_{\text{min}})\frac{C_{\text{Pmh}}}{N_{\text{min}}} + \frac{2}{3}\sum C_{\text{link}}, \qquad (1)$$

where N is the number of hops in the considered relay path, N_{min} is the number of hops in the path with the least hops, and C_{Pmh} is the sum of all link costs in the path with the least hops. Similar to SPF+ and all following strategies, the graph is pre-processed to exclude links with insufficient keys. Unfortunately, the NetworkX library does not have built-in functions to find the optimal path based on such complex equations. Therefore, (1) is evaluated for multiple paths and the path with the least total cost is selected for each individual key request. Reference [10] proposed to evaluate (1) for all feasible paths. This is possible for small networks, but is not realistic for large QKD networks due to computational complexity and constraints on latency. To limit the computational complexity of this routing strategy, (1) is only evaluated for the ten paths with the least number of hops. Reference [10] did not mention what should happen if multiple paths with unequal costs have the minimum number of hops. The first path returned by the shortest path function in the NetworkX library is used for calculating C_{Pmh} in this implementation.
- ADA-KEYS [10]. Similar to ADA-KGR but in this case the link cost is inversely proportional to the number of available keys, or residual key volume, per QKD link instead of the KGR, i.e., $C_{\text{link}} = 1/RKV$.
- OSKAR [15]. This strategy combines the KGR and number of available keys per QKD link in the calculation of the path cost, but does not consider the number of hops in the path.

$$C_{\text{link}} = \frac{M}{KGR} * 10^{\frac{M-S}{M}}, \qquad (2)$$

where S is the number of keys stored in the buffer for this QKD link and M is the maximum number of link keys that the KMS will store in its buffer per QKD link. For emulation purposes, M is set to 256 000 bit or, equivalently, 1000 keys of 256 bit. Since the path cost is simply the sum of all individual link costs in the path, Dijkstra's algorithm can be used without modifications.
- CUSTOM. Similar to ADA-KGR and ADA-KEYS, this strategy proposed by Yang et al. [9] includes an additional cost component for extra hops in the selected path but this strategy limits this extra cost in contrast to ADA-KGR and ADA-KEYS.

$$C_{\text{link}} = \begin{cases} 1, & \text{if } \frac{M_{\text{max}}-S}{KGR} \leq 1 \\ \frac{M_{\text{max}}-S}{KGR}, & \text{otherwise,} \end{cases} \qquad (3)$$

where M_{max} is the maximum value of M in the entire network.

$$C_{\text{path}} = f_{\text{path}} + \sum C_{\text{link}}, \qquad (4)$$

where

$$f_{\text{path}} = \begin{cases} \frac{(N-N_{\min})}{5}\overline{C_{\text{link}}}, & \text{if } (N-N_{\min}) \leq 5 \\ \overline{C_{\text{link}}}, & \text{otherwise} \end{cases} \quad (5)$$

and $\overline{C_{\text{link}}}$ is the average link cost in the considered path. Equation (4) is evaluated for a limited number of paths, following the same approach as used for the ADA strategy.

In contrast to distributed routing protocols, the central controller can continuously keep track of the exact number of available link keys per QKD link because the requested number of keys and the requested key size are included in each routing request. If the SPF strategy is used, the central controller will always return a relay path if the source and destination KMS are in the graph, regardless of whether all link keys required for key relaying are available on every KMS in the relay path or not. If one of the other strategies is used, the central controller will first check if the necessary resources are available in the relay path and will return an error message if no suitable path is found. This can be considered an enhanced version of admission control and avoids wastage of link keys due to failing key relay attempts.

F. Experiments

Two experiments were performed. During each experiment, every client requested 256 bit keys from its local KMS using the ETSI GS QKD 014 REST protocol. The interval between key requests from each client is exponentially distributed. We varied the mean interval and number of keys per request to emulate different network loads. In the first experiment, the number of keys per request was equal to one and the mean interval between requests were chosen in the range 0.5 s to 2 s. In the second experiment, the number of keys per request was varied between 1 and 10 and the mean interval between key requests was adjusted to keep the average key consumption rate constant at 256 bit/s. That means, every client requested 10 keys of 256 bits at once and waited 10 seconds on average before the next 10 keys were requested. This allowed the KMS to relay multiple keys in a single relay message and was done to investigate the impact of ITS authentication of relay messages on the achieved success rate. The responders were selected at random with uniform probability. The clients wrote relevant information about their key requests to a database on the shared drive. This database is used for the analysis in section IV.

During each experiment, it was recorded for each key request whether it succeeded or failed. A key request succeeded if the client key was issued to the initiator and client keys were only issued after key relaying finished successfully. Key requests by all clients were recorded over a period of time and the ratio of key requests that succeeded is called the success rate. A python script was used to initialize the network emulation and start all containers. The containers running the client software were started four minutes after the initialization of the QKD network to allow for exchanging key material and filling buffers on the KMS before the first key request. This was done to avoid that all key requests would fail due to lack of keys at the start of the experiment. However, this caused a peak in the measured success rate at the start of the experiment. Plots were made for each experiment to investigate after how many key requests the success rate stabilized. We discarded all measurement data recorded before the success rate stabilized.

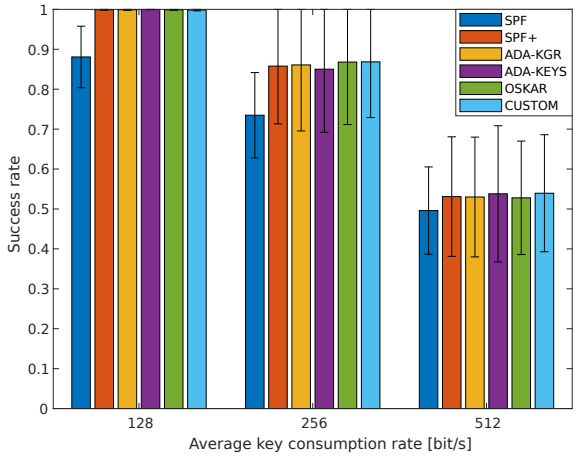

Fig. 4. Success rate vs the average key consumption rate of each client in bits per seconds for different routing strategies. Each client periodically requests one 256 bit key after every key consumption interval. The average key consumption interval is varied between 0.5 s and 2 s.

The remaining data were split into multiple segments based on time windows. Each segment represents an experiment of limited duration. This allowed for repeated measurements without initializing the QKD network again. The average value of the success rates of all segments is calculated and presented in the next section. In addition, the standard deviation is indicated with error bars. All emulated processes were executed in real time and only a limited number of key requests could be handled simultaneously due to hardware constraints. Therefore, experiments were time-consuming and the number of recorded key requests is limited. In combination with the varying availability of link keys this gives rise to larger calculated standard deviations, but Welch's t-test can be used to determine if the measured differences in average success rates are statistically significant or not.

IV. Results and Discussion

Fig. 4 shows the success rate achieved by each routing strategy. It can be seen that the success rate achieved by SPF is consistently lower than the success rates by the other, resource-aware routing strategies, especially for the lower key consumption rates. The maximum observed difference in success rates achieved using SPF and the resource-aware strategies is 13.4 percentage point. Although the calculated standard deviations are large, Welch's t-test confirms that the differences between the success rates achieved by SPF and the other strategies are statistically significant ($p < 0.05$). The differences get smaller for increased network loads. The differences amongst the success rates achieved by the resource-aware routing strategies

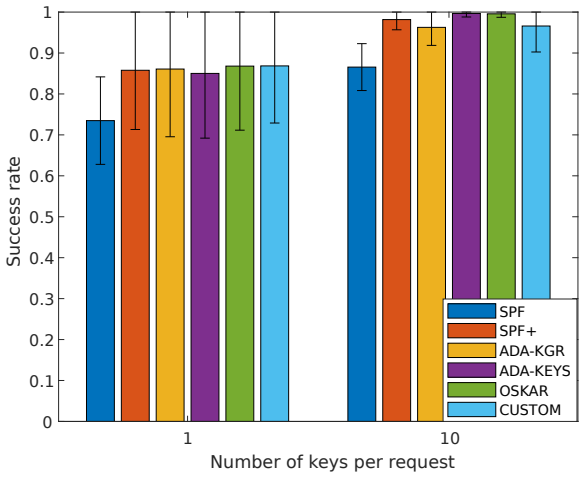

Fig. 5. Success rate vs number of keys per request for different routing strategies. The average key consumption rate per client is 256 bit/s.

are statistically not significant, but the computational complexity of each routing strategy is different. In particular, SPF and SPF+ are computationally less expensive than the other strategies. Assuming that potential QKD network operators will aim for a success rate over 99 %, the higher average key consumption rate that can be sustained is the most significant advantage of resource-aware routing but it is not necessary to use complex algorithms to achieve this.

Next, the influence of the information-theoretically secure authentication of relay messages on the success rate is discussed. In the discussed experiments client keys of 256 bit were relayed and each relay message had to be authenticated using a 256 bit key. This is an efficiency of at most 50 %. The efficiency increases if larger keys or more keys are relayed in a single relay message. This can lead to an increased success rate as shown in Fig. 5.

V. Conclusions

Many containerized clients and servers were used to emulate key relaying and evaluate the performance of quantum key distribution networks based on trusted relay nodes. The main focus was on how the success rate of key requests is influenced by the selection of relay paths in case of on-demand relaying. Different proposals for routing metrics from multiple papers were compared using a centralized routing approach. It was found that using centralized resource-aware routing can increase the success rate up to 13.4 percentage point compared to selecting the shortest path without taking the availability of link keys into account. However, the differences amongst the success rates achieved by different resource-aware routing algorithm are not statistically relevant, if the centralized routing approach presented in this paper is used. Hence, this paper recommends to focus on resource-aware routing algorithms that are computationally inexpensive and suitable for large-scale implementation. If a centralized routing approach is used, the success rate of key requests can be significantly increased in case of on-demand relaying by disregarding all relay paths where the required link keys are not available and selecting the path with the least number of hops from the remaining feasible paths. This method is simple and effective.

In addition, it was investigated how the success rate is influenced by information-theoretically secure authentication of relayed keys. This security measure limits the amount of link keys that is available for relaying and limits the success rate. This effect can be mitigated by relaying multiple keys per relay message.

Acknowledgment

Authors would like to thank SURF and dr. ir. R.C.J. Smets for their collaboration and support.

References

[1] D. J. Bernstein and T. Lange, "Post-quantum cryptography," *Nature*, vol. 549, no. 7671, pp. 188–194, Sep 2017. [Online]. Available: https://doi.org/10.1038/nature23461

[2] Alagic et al. (2022) Status Report on the Third Round of the NIST Post-Quantum Cryptography Standardization Process. [Online]. Available: https://doi.org/10.6028/NIST.IR.8413-upd1

[3] C. H. Bennett and G. Brassard, "Quantum cryptography: Public key distribution and coin tossing," *Theoretical Computer Science*, vol. 560, pp. 7–11, 2014, Theoretical Aspects of Quantum Cryptography – celebrating 30 years of BB84. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0304397514004241

[4] M. Takeoka, S. Guha, and M. M. Wilde, "Fundamental rate-loss tradeoff for optical quantum key distribution." *Nat. Comm.*, vol. 5, p. 5235, 2014, 5235. [Online]. Available: https://doi.org/10.1038/ncomms6235

[5] H. Briegel, W. Dür, J. I. Cirac, and P. Zoller, "Quantum Repeaters: The Role of Imperfect Local Operations in Quantum Communication," *Physical Review Letters*, vol. 81, no. 26, pp. 5932–5935, 1998, 5932. [Online]. Available: https://doi.org/10.1103/PhysRevLett.81.5932

[6] M. Dianati, R. Alléaume, M. Gagnaire, and X. S. Shen, "Architecture and protocols of the future European quantum key distribution network," *Security and Communication Networks*, vol. 1, no. 1, pp. 57–74, 2008, 57. [Online]. Available: https://doi.org/10.1002/sec.13

[7] Y. Tanizawa, R. Takahashi, and A. R. Dixon, "A routing method designed for a Quantum Key Distribution network," in *2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN)*, 2016, pp. 208–214.

[8] J. Moy, "OSPF Version 2," RFC 2328, apr 1998. [Online]. Available: https://www.rfc-editor.org/info/rfc2328

[9] C. Yang, H. Zhang, and J. Su, "The QKD network: model and routing scheme," *Journal of Modern Optics*, vol. 64, no. 21, pp. 2350–2362, 2017, 2350. [Online]. Available: https://doi.org/10.1080/09500340.2017.1360956

[10] L.-Q. Chen, M.-N. Zhao, K.-L. Yu, T.-Y. Tu, Y.-L. Zhao, and Y.-C. Wang, "ADA-QKDN: a new quantum key distribution network routing scheme based on application demand adaptation," *Quantum Information Processing*, vol. 20, no. 9, 2021. [Online]. Available: https://doi.org/10.1007/s11128-021-03246-2

[11] ETSI. (2019) Quantum Key Distribution (QKD); Protocol and data format of REST-based key delivery API. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/QKD/001_099/014/01.01.01_60/gs_qkd014v010101p.pdf

[12] M. Peuster, H. Karl, and S. van Rossem, "MeDICINE: Rapid prototyping of production-ready network services in multi-PoP environments," in *2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, 2016, pp. 148–153.

[13] D. J. Bernstein, ""The Poly1305-AES Message-Authentication Code"," in *"Fast Software Encryption"*, H. Gilbert and H. Handschuh, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 32–49.

[14] A. Hagberg, D. Schult, and P. Swart. (2023) NetworkX. [Online]. Available: https://networkx.org

[15] C. Yang, H. Zhang, and J. Su, "Quantum key distribution network: Optimal secret-key-aware routing method for trust relaying," *China Communications*, vol. 15, no. 2, 2018. [Online]. Available: https://doi.org/10.1109/CC.2018.8300270