Key Management Systems for Large-Scale Quantum Key Distribution Networks

Paul James
paul.james@ait.ac.at
AIT Austrian Institute of Technology
Vienna, Austria

Stephan Laschet
stephan.laschet@ait.ac.at
AIT Austrian Institute of Technology
Vienna, Austria

Sebastian Ramacher
sebastian.ramacher@ait.ac.at
AIT Austrian Institute of Technology
Vienna, Austria

Luca Torresetti
luca.torresetti@ait.ac.at
AIT Austrian Institute of Technology
Vienna, Austria

ABSTRACT

The Key Management System (KMS) is an important component in scaling up from link-to-link key generation to large key distribution networks. In this work we provide an overview of a KMS in the context of Quantum Key Distribution Networks (QKDN) and give a thorough summary of the functionality of a KMS in such an application. Beyond classical QKDNs, we discuss Post Quantum Cryptography (PQC) hybridization techniques at the KMS level. These methods add an additional layer of security against quantum computer driven attacks. We also discuss selected topics regarding the development, deployment and operation of components for such security infrastructure. In addition, relevant standards in the realm of Quantum Key Distribution (QKD) are outlined and analyzed. As some of the necessary interfaces have not been standardized, namely the interface between two KMS instances and the interface between the KMS and the Software Defined Network (SDN) Agent, we propose APIs for these two cases. The design of the interface between the KMS and QKD modules is discussed and, considering their resource constraints, a push mode for the ETSI GS QKD 004 standard is proposed. Finally, implementation details of a prototype KMS are outlined and trade-offs are discussed.

CCS CONCEPTS

• **Hardware → Quantum communication and cryptography**.

ACM Reference Format:
Paul James, Stephan Laschet, Sebastian Ramacher, and Luca Torresetti. 2023. Key Management Systems for Large-Scale Quantum Key Distribution Networks. In *The 18th International Conference on Availability, Reliability and Security (ARES 2023), August 29–September 01, 2023, Benevento, Italy*. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3600160.3605050

1 INTRODUCTION

Classical cryptography is in danger because of the progress on the development of quantum computers and the threat of attacks based on Shor's quantum algorithm [49] for the prime factorization of large integers. As a result the scientific community and a growing number of companies are working to replace cryptographic systems with quantum-resistant solutions. In addition, due to attacks based on Grover's algorithm [31], the key size in symmetric encryption schemes and the digest length of hash functions need to be increased. Public-key encryption schemes, key encapsulation mechanisms and digital signature schemes must now be built from hardness assumptions for powerful quantum computer.

Quantum Key Distribution (QKD) [1, 38] provides a method to generate keys for symmetric encryption at two endpoints which does not rely on computational hardness security assumptions, unlike post-quantum secure cryptographic schemes. Instead, QKD is based on quantum mechanics which guarantees security against an adversary with unbounded computational capabilities. Bennett and Brassard [3] proposed prepare-and-measure protocols in their seminal work, which can be summarized as follows: two endpoints are connected via an optical medium, one endpoint encodes a random number in quantum states of photons and transmits them, the other endpoint extracts the random number from measurements of the received quantum states. Thus if an attacker measures the photon quantum states during transmission the information is destroyed according to the no-cloning theorem [14, 53]. Since Bennett and Brassard's proposal, implementation techniques have evolved leading to discrete variable-based [2, 4], continuous variable based [30, 44, 50] and entanglement-based [54] protocols.

The transmission distance of QKD systems is limited to, at most, a few hundred kilometers and only between two directly connected endpoints. This provides a challenge when deploying large scale QKD networks (QKDN) for end-to-end secure communication. The QKD Key Management System (KMS) solves these limitations by using *Trusted Nodes* [32, 38] (TN) to forward keys through a network, thus providing keys between any network nodes. In addition, as outlined in Section 2.1, a QKD KMS includes features such as: database consistency across distributed KMS instances, security requirements, telemetry and key consumption monitoring.

1.1 Related Work

The European Quantum Communication Infrastructure (EuroQCI) initiative [12] aims to establish a QKD network connecting all member states of the European Union and its institutions. This network is planned to include both terrestrial and satellite quantum

ARES 2023, August 29–September 01, 2023, Benevento, Italy
© 2023 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0772-8/23/08.
https://doi.org/10.1145/3600160.3605050

communication infrastructures (QCI) and the national parts are currently being developed. Also notable in Europe is MadQCI [37] which is one of the most advanced of this type of network with more than ten QKD links deployed.

Outside Europe, China successfully launched a space QCI in the Quantum Experiments at Space Scale (QUESS) program in 2016 [17] and is now deploying terrestrial QKD networks [38]. The USA built the DARPA Quantum Network [16] which was the first of its kind, though it is now discontinued. Japan deployed a QKD Network and demonstrated TV conferencing over a distance of 45km [47].

1.2 Contribution

This work contributes to various aspects of KMSes in large-scale QKD networks as described in this section.

The current efforts on QKD KMS standardization focus on the interaction between a KMS and other components, such as security applications, but leave other aspects open, such as an inter-KMS interface. Hence Section 7.1 proposes an interface that deals with communication between KMS instances throughout a network. The issues considered include establishing consensus in distributed systems [36] and key confirmation, which are standard practice in common protocols (e.g., TLS 1.3 [45]). Additionally, no specification for an interface between a KMS and an SDN Agent exists therefore one is proposed in Section 7.2.

Schemes for implementing key relay in QKDNs are analyzed in Section 2.2. Methods outlined in standards for use in large scale multi-operator and multi-customer networks are discussed. Further, to strengthen the overall security guarantees provided by a QKDN, key forwarding methods with post-quantum secure cryptographic schemes are extended (Section 3) to ensure end-to-end security properties even in the event of compromised trusted nodes. These are based on the hybrid authenticated key exchange protocol first presented by Dowling, Brandt Hansen, and Paterson [15] and the recently proposed extension Muckle+ [9].

Two different modes of operation of the ETSI GS QKD 004 standard are outlined in Section 6.1. With these standard-compliant modes the key on demand requirement of applications and the instant delivery requirement of QKD devices are satisfied.

2 KMS ARCHITECTURE

Figure 1 illustrates that the KMS is one of the main components for scaling up direct links (connections *q*) into a Quantum Key Distribution Network (QKDN). The KMS connects to the QKD modules (connection *b*) and receives keys from directly connected nodes which it then distributes to any node in the network. QKD nodes can host multiple applications that retrieve keys from the KMS (connection *a*).

When scaling up to a network, managing the overall system dynamically is best implemented by a Central Network Controller, outside of the KMS, connected to a Node Control instance deployed on each QKD node. This in turn is connected to the individual components (connection *c* and *d*). The Software Defined Network (SDN) concept fulfills this task and standards have been established for QKD [22].

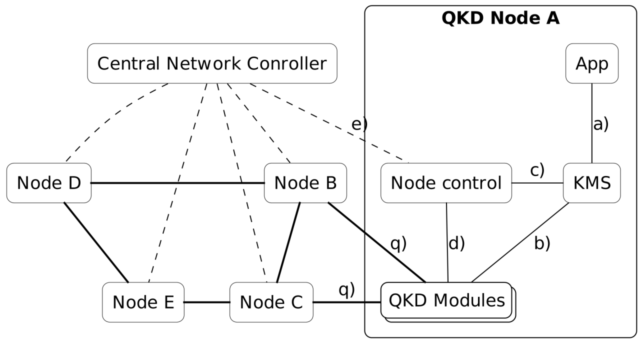

Figure 1: QKDN System architecture [22]. QKD node A shows the building blocks of one QKD node containing QKD Modules, KMS, node controller and optional applications. Nodes B to E contain of the same blocks. The Central Network Controller monitors and controls the QKDN. Dashed lines depict network communication, normal lines interfaces within the same security boundary and bold lines quantum channels.

2.1 Main KMS Tasks Within a QKDN

There is an important distinction to be made between a QKDN-KMS (QKMS) and a Cryptographic Key Management System (CKMS) in classical cryptography applications as outlined in NIST SP 800-130 [40] and NIST SP 800-57 [43]. Most CKMS implementations focus on being a secure key storage with management functions and standardized interfaces. They rely on current state of the art cryptography algorithms, for example, they may use asymmetric cryptography over the internet to forward keys. However, a QKMS has to consider the physical implementation limitations of QKD devices, the different interface requirements and the information-theoretic security (ITS) promise. Therefore, a QKMS requires a significantly different design. The main tasks of a QKD KMS are:

Key Forwarding: Also referred to as "key relay" or "multi-hop" in some literature [20, 35], key forwarding is one of the core functions of a KMS. It is the process of establishing an end-to-end key using the intermediate trusted nodes of the QKDN. These TNs must be trustworthy, since during forwarding they temporarily obtain the final key as discussed in Section 2.2.

Key management: The use of TNs means that a breach of QKD keys is one of the worst-case scenarios. Therefore, to reduce the harm that compromised keys can cause, QKD keys that are not used within a certain time period are deleted. For the same reason, securely deleting keys after delivery is important. Key management also keeps track of the state of keys, such as whether a key has been synchronized or if a reserved key has been delivered to one node but is not yet at the corresponding node.

Database synchronization: The KMS can be considered a distributed database system, therefore database consistency must be ensured. This includes verifying that the data on peer KMSes is consistent as entries are created, resized, deleted or changed.

Bootstrapping: Pre-Shared-Keys (PSK) are used to authenticate and encrypt the initial QKD messages, i.e. before QKD keys are available.

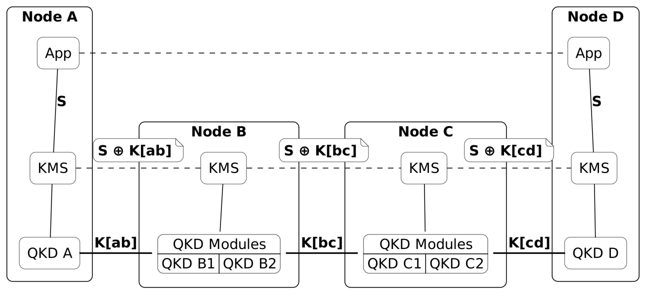

Figure 2: Key Forwarding through intermediate nodes. $K[xy]$ denotes a key established between nodes x and y, regardless by which QKD device. S is the secret generated by an RNG and is the key delivered to the app. Dashed lines depict network communication, normal lines interfaces within the same security boundary and bold lines quantum channels.

Communication interfaces: The KMS is a central element in a QKD node with interfaces to the QKD modules, apps, node controller and optional components, such as an RNG or a tamper controller.

Quality of Service management: The KMS informs the network controller of key availability so that an optimal path, at the required bandwidth, can be selected. The KMS must enforce the performance values agreed with the apps and network provider, such as key consumption rate.

Additionally other features such as logging, monitoring, interface handling, configuration, remote maintenance are implemented in a KMS. These features, however, are not specific to a KMS and are thus not discussed in detail in this work.

2.2 Key Forwarding

Figure 2 illustrates the forwarding of keys through intermediate nodes when an application requests keys for its peer application on a QKDN node with which it has no direct connection. The Network Controller configures the forwarding tables of the KMSes that are in the chosen path through the QKDN to the destination node. The source KMS then generates a random value, to be used by both applications as a key, and encrypts it using the key generated by the QKD device on the next directly linked node. The encrypted key is forwarded to the KMS on the next node in the path, which then decrypts it, obtaining the randomly generated value, re-encrypts it using QKD generated keys and forwards it. The common ITS de- and encryption method is a One Time Pad (OTP) bitwise exclusive or (XOR) operation, denoted as ⊕, as discussed in Section 2.3. This forwarding process continues until the final destination is reached at which point the two end-nodes, which are not connected directly, share symmetric keys that can be provided to the applications.

Figure 2 depicts this process with an RNG generated secret key S being delivered to the applications. An alternative would be to use a QKD key from the source KMS, such as $K[ab]$ for node A, which would thus eliminate the first forwarding message between nodes A and B.

Other approaches exist [35], one of which is shown in Figure 3. Each intermediate node contributes data to a computation on a

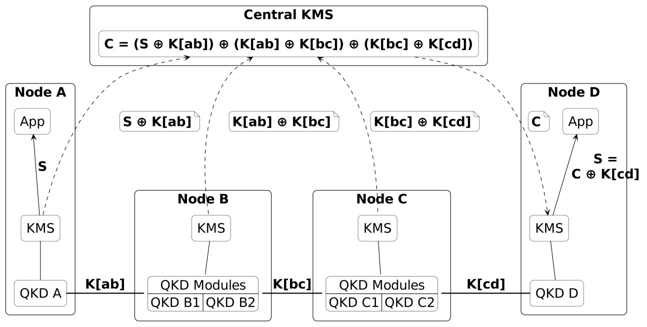

Figure 3: Key Forwarding with central KMS. $K[xy]$ is a key established between nodes x and y, regardless of the QKD device that produced it. S is the secret generated by an RNG and is the key delivered to the application. Dashed lines depict network communication, normal lines interfaces within the same security boundary and bold lines quantum channels.

central KMS entity, which in turn sends the result to the endpoint. The starting KMS (on node A) sends the secret S, encrypted with the next neighbor's QKD key, to the central KMS. Each intermediate KMS (on nodes B and C) only send the OTP product of the QKD keys shared with the next nodes in the path. The destination KMS (on node D) only receives the computation result C from which it can extract the secret S.

It has been claimed that the approach in Figure 3 reduces the trust requirements of the individual trusted nodes [13], although the following analysis shows that the forwarding design in Figure 2 is slightly favorable.

An issue with the centralized architecture is that it has a single point of failure. If the central KMS fails, the whole QKDN is not operational. The QKDN premise is to have ITS quality and that transmitted messages can be read, since the attack model commonly used assumes an attacker has unbounded computational capabilities and can spoof the network traffic. From this arises another issue. The peer-to-peer messages can at least be authenticated with a symmetric ITS authentication algorithm, but since the central KMS cannot share a key with each node in an ITS way, the messages cannot be authenticated. This makes it possible for an attacker to impersonate the central KMS and obtain all messages more easily.

The final point on why this approach cannot relax the trust assumption on each intermediate node becomes apparent when breaking down the computation in the central node:[1]

$$C = \underbrace{\underbrace{(S \oplus K_{ab}) \oplus (K_{ab} \oplus K_{bc})}_{=S \oplus K_{bc}} \oplus (K_{bc} \oplus K_{cd})}_{=S \oplus K_{cd}} \tag{1}$$

The combined terms of the Equation (1) are equal to the messages forwarded in the peer-to-peer forwarding architecture shown in Figure 2. If an attacker can obtain the keys of any one node they know the secret S because they are assumed to be able to read every message. Depending on which QKD key was obtained, a subset of messages may be sufficient to obtain S.

[1] K_{xy} in the equation is equivalent to $K[xy]$ in the figures

In both approaches, if one of the nodes is compromised or the attacker obtains intermediate QKD keys in some way, the final key delivered to applications can be obtained by an adversary. Therefore, each intermediate node must be trustworthy, hence the commonly used term *trusted node (TN)* [34].

2.3 Security Algorithms

As stated, QKD modules promise security against an adversary with unbounded computational power. Thus when scaling up to a QKD network, this security guarantee needs to be upheld. To retain information-theoretic security of the overall system, it is important to build the key forwarding functionality from ITS primitives. This thus means being limited to a set of cryptographic schemes and primitives that are secure against unbounded adversaries.

Encryption: The only known ITS encryption technique is the OTP using the bitwise XOR operation. This is provable secure because the bitwise XOR has an equal 50% chance of transforming a 1 in a message into a 0 or a 1 in the cipher text, using the bit value of a random key value. “In other words, there is at least one key which transforms any given message into any of the cryptograms.” [39]. The only option left for an attacker is to guess. A disadvantage with this kind of scheme is that each key can only be used once and has to be the same size as the message.

Authentication: Information-theoretic secure authentication can be built from message authentication codes (MAC) based on universal hashing functions (UHF), as initially proposed by Wegman and Carter [10, 52]. Such UHFs can be instantiated using polynomial hashes. Examples of ITS MACs include UMAC [7] and Poly1305 [6]. Poly1305 is highly efficient and widely available in implementations of cryptographic libraries.

Note that here also, keys can only be used once for authentication. Therefore, an adversary would need to forge an authentication tag under a specific key whilst seeing at most one authenticated message. In MACs built from UHFs, an adversary’s advantage is bounded by the probability of finding a collision in the tag space, i.e. if the tag space T has size $|T|$ the collision probability is $1/|T|$.

In the original proposal by Wegman and Carter (W&C), keys must be approximately twice as long as the message, thus rendering it impractical, especially under the constraint that keys cannot be re-used. The restrictions on the key size and key re-use can be lifted by restricting the message space to continuously numbered messages and splitting the key of the MAC into one part that selects the hash function and a second part that masks the output of the hash function for up to n messages [52]. Hence, keys have the form $(h_k, k_1, \dots, k_n)$ and the MAC tag for message $(i\|m)$ is computed as $h_k(m) \oplus k_i$.

Asymmetric cryptography and other schemes: By using only ITS algorithms, many cryptography building blocks must be omitted. Hence, if schemes need to be used where the desired properties are reduced to computational hardness assumptions, they are unfit for use in an ITS regime. Consequently, desired properties in large-scale networks cannot be achieved, such as end-to-end authenticity and confidentiality, which rely on computationally secure key encapsulation mechanisms (KEM) and digital signature schemes.

3 HYBRIDIZATION WITH POST QUANTUM CRYPTOGRAPHY

As discussed above, QKD networks suffer from the lack of end-to-end authentication. Recently, the hybrid authenticated key exchange protocol Muckle+ [9] was proposed which solves this issue for applications. The protocol combines key material from a QKD network with an ephemeral key from a KEM-based key exchange, whereas the peers are authenticated via digital signature schemes. By using such a system, users may rely on the security of QKD or post-quantum secure key exchanges as long as at least one of the two technologies remains secure. Their approach however does not directly translate to the main task of a KMS – except for KMS to KMS communication – but provides a starting point to solve the trusted node problem with the additional use of post-quantum secure cryptographic primitives.

A PQC KEM would allow a symmetric key to be generated at the two endpoint KMSes, basically bypassing the QKDN and therefore all the downsides associated with it, especially the TN assumption. As soon as the two endpoint KMSes in the system have the QKDN forwarded key with the ITS quality and the PQC generated key with the end-to-end quality, they derive a final key from the two keys using a key derivation function (KDF), or more generally, with a key combiner [29]. While OTP for encryption has disadvantages as for example pointed out by the German Federal Office for Information Security (BSI) [28], the XOR combiner preserves security against chosen plaintext attacks (IND-CPA) provided at least one of the keys was obtained from a IND-CPA secure key exchange mechanism. With the XOR combiner approach the system can provide a key that is generally ITS against any technology and additionally end-to-end against quantum computer driven attacks.

On top of that, digital signatures can be used to authenticate the other KMS instances. In the process of delivering the key to the application, the endpoint KMSes could exchange some messages signed with a PQC algorithm to verify that the communication parties are legitimate. Another application of PQC signatures is that each intermediate TN signs during forwarding, so the end point KMS can verify that the forwarding path chosen was the actual path taken. This mitigates an attack that forwards all messages through one compromised node without being noticed.

For an additional layer of security, PQC algorithms can be applied throughout the whole QKDN, such as in the KMS interfaces with the applications, the QKD modules and the SDN Agent. PQC algorithms can also be applied to secure the SDN Controller connections. This could allow a deployment of the SDN Controller outside the QKDN. The comparably high key rate of PQC algorithms, can improve the relatively high data traffic from an SDN Controller to all QKDN nodes and users negotiating Service Level Agreements (SLA).

In any case, to support the use of authentication mechanisms based on post-quantum schemes, a public key infrastructure (PKI) can be deployed to ensure the authenticity of the long term keys.

4 SECURITY INFRASTRUCTURE CONSIDERATIONS

When designing, developing, deploying and operating a security infrastructure, like a QKDN, the considerations outlined below must be taken into account. The aforementioned trusted node problem in particular requires special consideration. Usually node components, such as the KMS, are deployed within the same security boundary on a trusted node. Whilst this is a simple concept, in practice it can have severe consequences, necessitating management, operational and technical measurements to detect an attacker accessing the TN.

Controls: NIST SP 800-53 [41] formalizes twenty control families and their mechanisms, such as access control, audits and supply chain risk management. It could be argued that a TN must comply with most of these controls, however, the considerable operational overhead needs to be taken into account. On an organizational level the ISO/IEC 27000-series [33] provides further guidance for organizations developing security critical software or operating such systems.

Tampering: Anti-tamper technology is about techniques against malicious manipulation of a system. There are different techniques for dealing with this and for the KMS the most important one is tamper response, which is a systems response to a detected tamper event. FIPS 140-2 [27] outlines several security requirements for a cryptographic module, including recommendations on anti-tamper mechanisms. Among these zeroization, which deletes all sensitive information, is one of the most relevant mechanisms.

Security by design: This refers to the process of developing security critical software and is applicable to all QKDN components, i.e. the KMS also. One aspect is the secure software development life cycle which the aforementioned ISO 27000-family touches on. Also the NIST secure software development framework SP 800-218 [42] includes controls and best practices for requirements engineering, design, development, release, deployment, maintenance and end-of-life of the software.

Certification: Applications in the area of security are commonly certified, with one of the most relevant being the Common Criteria (CC) for Information Technology Security Evaluation, which results in Evaluation Assurance Level (EAL) certification [11]. It can be argued that it is applicable to all QKDN components, including the KMS, at a target level of 4, which means that the target of evaluation (TOE) was methodically designed, tested and reviewed. Protection profiles (PP) are used to describe the TOE and ETSI has a draft PP for QKD devices ETSI GS QKD 016 [23] available, with the first release scheduled for 2023. Also, certification by compliance with the aforementioned FIPS 140-2 and ISO 27000 family is possible.

5 NETWORK TECHNOLOGY

As outlined above, one KMS is typically connected to multiple QKD modules and one or more applications. The intention is to deploy each component on different hardware in a server rack and to connect the components via a network interface. Network APIs are often RESTful, using protocols such as CoAP and HTTP(S) [26], therefore CoAP [48], which is based on the REST architectural style [18] is discussed.

5.1 Representational State Transfer

The Representational State Transfer (REST) architectural style is about interactions between active elements, such as clients and servers. Applying REST constraints promotes scalability, generality, independent deployment, reduced network latency and increased security. With each constraint though there is a trade off between the advantages and disadvantages of applying it [25]. The main REST design principles [18] are:

- The key abstraction of information is a resource with a URL.
- There is a layer of indirection between an abstract resource and its concrete representation, for example, a resource can be accessed via its URL without knowing its precise location.
- Interactions are context-free. Each client request contains all necessary information for the server to understand the request, there is no stored context on the server.
- Only a few simple operations are available. Modules perform only a small set of well-defined methods on a resource. Information is transferred in a standardized form not in an application specific form.
- Techniques to support caching are encouraged, which improves performance but may reduce reliability.
- Filtering and redirection intermediaries can use metadata to transparently restrict or modify requests and responses.

5.2 The Constrained Application Protocol

CoAP is a low overhead machine to machine (M2M) protocol that is designed for networks that contain low powered nodes [48]. This is beneficial in a KMS because applications and QKD devices may have limited resources.

The Internet Engineering Task Force's (IETF) recommended method for transporting data over constrained node networks is CoAP over UDP, however CoAP also includes transport over TCP, TLS and Websockets [8].

CoAP has a client/server interaction model, however, for machine to machine applications, such as a KMS, a CoAP implementation can perform both client and server roles. A typical interaction is that a CoAP endpoint in a client role, sends a request to an endpoint in a server role, requesting an action on that server. The action is specified by a method code and the resource is identified by a URI. The CoAP server receiving the request performs the action and then returns a response message containing a response code to indicate the success, or otherwise, of the requested action. The requests and responses are exchanged asynchronously between CoAP endpoints.

CoAP contains a subset of REST that is optimized for M2M applications. Request messages contain a method code specifying either a GET, PUT, POST, or DELETE method. Response messages contain a response code taken from a subset of HTTP status codes or a CoAP-specific code.

For reliable transmission a message is marked as confirmable, which means that it is retransmitted until acknowledged by the corresponding endpoint. CoAP also includes duplicate message detection and defines several security modes including DTLS [46], a TLS based protocol allowing UDP as a transport protocol.

6 QKD STANDARDS

QKD is a relatively new technology which is continually developing. It has an increasing Technology Readiness Level (TRL) and early metropolitan deployments. In this phase, standards can improve the interoperability and design of QKDN components. The main standardization organizations are ETSI and ITU. This section gives an overview of the most relevant standards for a QKD KMS.

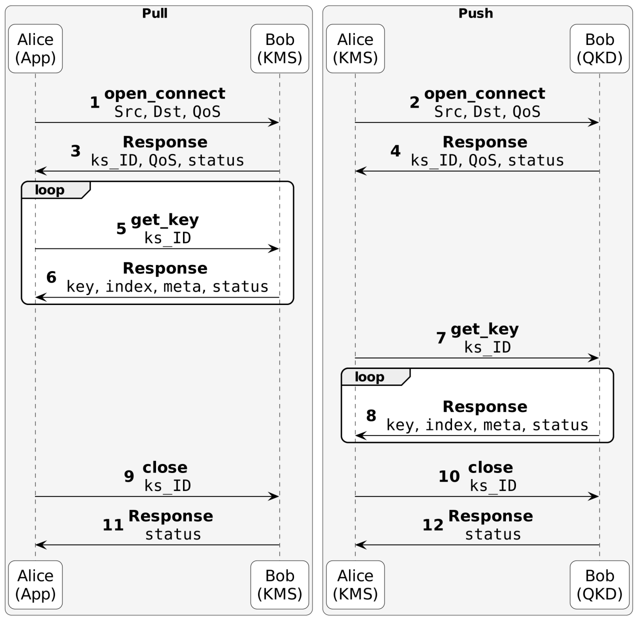
Pull
Alice (App)
Bob (KMS)
1 open_connect Src, Dst, QoS
3 Response ks_ID, QoS, status
loop
5 get_key ks_ID
6 Response key, index, meta, status
9 close ks_ID
11 Response status
Push
Alice (KMS)
Bob (QKD)
2 open_connect Src, Dst, QoS
4 Response ks_ID, QoS, status
7 get_key ks_ID
loop
8 Response key, index, meta, status
10 close ks_ID
12 Response status

Figure 4: ETSI 004 pull vs. push operational mode.

6.1 ETSI GS QKD 004

The ETSI GS QKD 004 defines an API between an application and a QKD Key Manager (KM), which is synonymous with KMS. The standard defines the following functions:

`open_connect`. The application requests from the KM a key stream between the supplied source and destination endpoints with the characteristics defined in the Quality of Service (QoS) parameter. The KM responds with a key stream id, and optionally a QoS proposal, or rejects the request.

`get_key`. The application requests keys for the given key stream id. Additional metadata can be supplied.

`close`. The application requests that the key stream is closed. Keys already allocated for this key stream are held until the other endpoint also sends a `close` request or the key stream's Time To Live (TTL) parameter expires.

The standard also defines several other details such as error processing and message sequences. The important thing to note is that the standard is implementation technology independent. On the one hand, this allows flexibility and deployment on different systems or abstraction layers, on the other hand, it creates additional integration effort.

Pull and Push based operation mode:

The ETSI GS QKD 004 can also be used for communication between a QKD module and a KMS. However, applications and QKD modules have different needs. Applications need keys on demand whilst QKD devices deliver keys when generated because resource constraints make data storage difficult. Thus an application pulls keys when it needs them whilst a QKD device pushes keys as it generates them. The generic nature of ETSI 004 enables pull and push operations to be designed that comply with the standard. These two modes of operation are illustrated in Figure 4. The `open_connect` and `close` messages follow a request response pattern between the Alice and Bob (messages 1-4 and 9-12). In the pull mode, the `get_key` is initiated by Alice (an application) and the corresponding response is a key (messages 5,6). Whilst in the push mode, Alice (a KMS) sends a single request (message 8) to start the key retrieval process and Bob (a QKD device) sends a `get_key` response (message 8) whenever a new key is generated.

The two modes have their respective trade-offs. The push mode requires the QKD module to offer both server and client capabilities, since it has to reply to messages and send requests to the KMS. On the other hand, the pull mode forces the QKD device to safely store keys until requested by the KMS. Therefore, the push mode lightens the workload on the QKD module by shifting it towards the KMS.

6.2 ETSI GS QKD 014

The ETSI GS QKD 014 standard [21] defines a RESTful API for communication between a Secure Application Entity (SAE) and a Key Manager Entity (KME), which is just different terminology for an application and a KMS respectively. In the standard the SAE initiating the communication is referred to as the master SAE and the responding SAE is called the slave SAE. Communication is performed using the HTTPS protocol with TLS 1.2 or above. The API is composed of three methods:

`Get status`. Returns the KME's status data and information regarding the keys available with the slave SAE.

`Get key`. Returns key data to the master SAE, with optional parameters specifying additional key delivery requirements. The slave SAE may then request matching keys from its KME using the key ID provided in the response.

`Get key with Key IDs`. Returns key data for the specified key ID to the calling slave SAE.

The standard also defines several other details such as message URIs, error processing and message sequences. In contrast to ETSI GS QKD 004, this standard is implementation specific.

6.3 ETSI GS QKD 015

ETSI GS QKD 015 [22] introduces the concept of Software Defined Networks (SDN) for QKDNs. SDNs separate data and control by having the main control logic in a centralized SDN Controller. The standard defines an API between the SDN Controller and the SDN Agents which is deployed on the nodes to configure the node submodules, such as the KMS.

The standard is implementation independent. Interfaces to the individual node components, namely the KMS and QKD modules, is outside the scope of the standard, therefore, the API described in Section 7.2 is proposed for the KMS interface.

6.4 ITU-T Y.3803

The ITU has released several QKD recommendations, in particular ITU-T Y.3803 [35], for a KMS in a QKD system. It proposes a Key Manager (KM) architecture consisting of several components, with a split between a key management agent (KMA) and a key supply agent (KSA). The KMA's responsibility is to retrieve, store, forward and manage QKD keys, whilst the KSA deals with supplying keys to applications. The recommendation also proposes different key relay approaches, two of which are presented in the above Section 2.2.

7 PROPOSED INTERFACES

There are no appropriate standards for the KMS to KMS interface nor for the the KMS to SDN Agent interface. ETSI GS QKD 020 [24] is an unreleased draft for an interoperable KMS API standard which defines a KMS to KMS interface on the same node for cross vendor and border node applications. However, it only applies to two KMS instances on the same TN and communication beyond the node is not within its scope. Hence in the following sections we propose APIs for the two aforementioned non-standardized interfaces.

7.1 KMS to KMS Interface

This interface connects two KMS peers on two directly linked nodes. The exchanged messages are authenticated with a MAC, therefore, key material reserved for internal use must be available on the peers. As stated above, this interface is distinct from the unreleased ETSI GS QKD 020 [24].

7.1.1 Key modification. One of the main KMS functions is key confirmation, modification and forwarding. The APIs for these functions are grouped in this section.

`new_key_batch`. Synchronize a variable sized batch of keys, newly obtained from a QKD module, with the corresponding peer KMS. The message contains the key IDs and a MAC, but not the key values themselves. The MAC is calculated over the transmitted message as well as the key data received from the QKD device in order to ensure that the received data is consistent. Synchronization is done in batches to reduce internal key consumption. After the keys have been synchronized, they are considered confirmed at the peer and can be used.

`forward_keys`. Request key forwarding of a batch of keys to an intermediate KMS in the network. The message contains a list of encrypted keys, the corresponding key IDs and an ID that allows the peer to lookup the next node in the SDN configured forwarding table.

`split_key`. Notify a peer KMS of a key split, i.e. the division of a single key entry into smaller key entries. It contains a key ID, a list of new key IDs and their corresponding new lengths.

`merge_keys`. Notify a peer KMS of a key merge, i.e. the fusion of multiple key entries into a larger key entry. It contains a list of key IDs, a new key ID and the corresponding new length.

`delete_keys`. Request a peer KMS to delete a group of keys. It contains a list of key IDs.

`make_keys_internal`. Reserve keys for use in the KMS to generate MACs and forward keys. It contains a list of key IDs.

`make_keys_external`. Reserve keys for peer applications. It contains a list of key IDs, and a key stream ID.

7.1.2 Applications and key streams. Applications that register with a KMS must be synchronized. When an application closes the connection, the key stream ID becomes invalid and thus must be synchronized.

`new_app`. Notify a peer KMS of a new application's registration. The message contains the application ID, the source and destination address, a key stream ID and a QoS.

`key_stream_closed`. Notify a peer KMS of a key stream being closed. The message contains a key stream ID.

7.1.3 Peer availability. A KMS must be aware of the status of its peers. This information may be communicated through the SDN, but the ETSI 015 doesn't support that feature, therefore it is supported in this API.

`get_status`. Request the status of a peer KMS.

`post_status`. Notify the peer KMS of a status change. The message contains a status code and optionally additional data.

7.1.4 API Protocol. The lightweight CoAP is chosen as the message transport protocol. The payload can be disclosed to an attacker because the MAC ensures the authenticity and integrity of the messages and the encryption of the keys ensures the confidentiality of sensitive data. Hence the messages do not need protection on the transport protocol level, but could be done using TLS.

7.2 KMS to SDN Agent Interface

As already mentioned, ETSI QKD QS 015 does not specify how information between the SDN controller and the SDN Agent is communicated within the node. Hence the following interface is proposed:

7.2.1 KMS information. The SDN needs information about the KMS in order to model it as a virtual representation.

`capabilities.` The KMS communicates its key forwarding capability. There is potential to communicate additional capabilities, therefore this is foreseen as a list. One example is hybrid key establishment for selecting paths with higher security requirements.

`model.` The KMS communicates its device vendor, model and software version. The software version and capabilities may not correlate, since they could be configurable within the same software version.

`performance.` The KMS communicates its Secret Key Rate (SKR) and Effective Secret Key Rate (ESKR). The SKR indicates how many keys are received by the KMS and the ESKR takes into consideration that some keys are used internally.

7.2.2 KMS configuration. The SDN concept separates the data and control plane. The SDN Controller configures the forwarding paths of an end-to-end application link, such that each KMS on the path knows which direct neighbor KMS it should forward keys to.

`application_registration.` An application registers with the KMS which then communicates this registration, together with the QoS and source and destination addresses, to the SDN Agent. If the SDN Agent can confirm the registration, it creates a key stream ID and, if necessary, configures the next node in the forwarding path. In the response it may also suggest an alternative QoS if the requested one is unfeasible. In this manner an end-to-end link is established.

`link_config.` This message implements the HTTP methods POST, PUT and DELETE to respectively create, modify or delete a forwarding entry on each KMS affected by the establishment, modification or tear down of a link.

`application_close.` If an application requests closure of a key stream, the KMS communicates this event to the SDN Agent, specifying the corresponding key stream ID.

7.2.3 Reporting.

This is not considered in ETSI GS QKD 015, but reporting is relevant for centralized monitoring and configuration. The KMS reports its state and noteworthy events.

`status.` The KMS notifies the SDN if its status changes. The proposed states are operational, stopped and inactive. The stopped state communicates, for example, that the KMS is currently under maintenance or is transitioning to operational or inactive states. Additional details may also be sent.

`report.` This method sends information on noteworthy KMS events to the SDN. The message contains the severity of the event, an event code and detailed information.

7.2.4 API Protocol.

The proposed protocol is RESTful HTTP since it is commonly used by communication providers who integrate QKD into their SDN solutions. As the SDN Agent and the KMS are deployed within the same security boundary, securing this channel is not necessary, but could be done using HTTPS with TLS.

8 KMS PROTOTYPE

A KMS prototype is being developed within the EuroQCI framework. The prototype follows the principles and techniques outlined in the previous sections. This section discusses some design decisions and trade-offs in the prototype's development with reference to discussions in previous sections.

Overall Architecture: The prototype is designed for a system architecture that follows the ETSI GS QKD 015 SDN approach. It is most suitable for a European wide QKDN and provides a clear division between network management and key management.

Key Forwarding: The forwarding mechanism uses an external RNG as the final key source, therefore customers can use the RNG of their choice. Also, multi-hop forwarding through intermediate TNs is adopted. The downsides of a centralized KMS, as discussed earlier, like additional deployment efforts and potentially making attacks easier, make a centralized approach unappealing and hence this approach is not taken.

Security Algorithms: The prototype follows the ITS paradigm by encrypting sensitive data with an OTP, such as forwarded keys, and uses ITS MACs for messages between KMS instances. The preferred algorithm is Poly1305 because of its performance, the availability of favorably reviewed and tested libraries and its widespread use. A proprietary implementation of the algorithm proposed by W&C [51] is implemented as a configurable alternative, if a higher security guarantee is required.

PQC hybrid approach: The security analysis demonstrates that a hybridization with PQC KEMs is of great benefit to a QKDN. As discussed, the additional layer of security against an attacker with a quantum computer is the most urgent requirement. In this scenario, the PQC hybrid approach also reduces the trust assumption of the TNs greatly. Furthermore the KMS is designed with cryptographic agility in mind thus making future algorithm upgrades of the second non-ITS key straightforward.

Securing the KMS: Whilst the outlined certification and formal development methods are respected, they are not fully followed at the moment because the emphasis is on developing a prototype in a rapidly changing environment. Development of the KMS complies with the best practices in secure software development, such as test-driven development, static and dynamic code analysis and security by design concepts. The KMS design considers deployment on secured hardware and choices have been made to ensure that such deployment is feasible. Also, the use of zeroization as a tamper response in the KMS is being considered.

Network Technology: The prototype implements the ETSI GS QKD 004 standard using libcoap [19], an open source C library implementation of the lightweight CoAP protocol. It is particularly useful for devices that have constrained resources [5]. Applications use the pull mode of operation and QKD modules use the push mode of operation. This creates standard compliant interfaces that take the resource constraints of the QKD devices into consideration. The ETSI GS QKD 014 standard is also provided as an interface to the applications since it is well accepted. The proposed interfaces to the SDN Agent and between KMS instances are the APIs described above.

Standard compliance: The outlined ETSI standards, i.e. ETSI GS QKD 004, 014 and 015, are followed. ITU-T Y.3803 is being considered though not fully followed because of technical reasons.

9 CONCLUSION

The KMS is one of the central components of large-scale quantum networks, therefore security aspects in many different areas must be considered. Secure coding practices and the implementation of critical cryptographic components, as well as network interfaces, are required.

Within our KMS prototype we demonstrate potential extensions of a system that is initially only based on QKD with post-quantum secure primitives. Thereby, we are able to provide additional end-to-end security guarantees in the KMS-to-KMS communication even when the KMSes do not share physical links.

ACKNOWLEDGMENTS

This work received funding from Digital Europe Program under project numbers 101091642 (“QCI-CAT”), 101091588 (“QUARTER”), and 101091564 (“eCausis”). We thank Vicente Martin and his team of the Universidad Politécnica de Madrid (UPM) for the discussions especially regarding to the ETSI GS QKD 004 standard.

REFERENCES

[1] Romain Alléaume, Cyril Branciard, Jan Bouda, Thierry Debuisschert, Mehrdad Dianati, Nicolas Gisin, Mark Godfrey, Philippe Grangier, Thomas Länger, Norbert Lütkenhaus, Christian Monyk, Philippe Painchault, Momtchil Peev, Andreas Poppe, Thomas Pornin, John G. Rarity, Renato Renner, Gregoire Ribordy, Michel Riguidel, Louis Salvail, Andrew Shields, Harald Weinfurter, and Anton Zeilinger. 2014. Using quantum key distribution for cryptographic purposes: A survey. *Theor. Comput. Sci.* 560 (2014), 62–81.

[2] Charles H Bennett. 1992. Quantum cryptography using any two nonorthogonal states. *Physical review letters* 68, 21 (1992), 3121.

[3] Charles H. Bennett and Gilles Brassard. 1984. An Update on Quantum Cryptography (Impromptu Talk). In *CRYPTO'84 (LNCS)*, G. R. Blakley and David Chaum (Eds.), Vol. 196. Springer, Heidelberg, 475–480.

[4] Charles H. Bennett and Gilles Brassard. 2014. Quantum cryptography: Public key distribution and coin tossing. *Theoretical Computer Science* 560 (dec 2014), 7–11.

[5] Olaf Bergmann. 2015—2022. *C-Implementation of CoAP.* https://libcoap.net/

[6] Daniel J. Bernstein. 2005. The Poly1305-AES Message-Authentication Code. In *FSE 2005 (LNCS)*, Henri Gilbert and Helena Handschuh (Eds.), Vol. 3557. Springer, Heidelberg, 32–49. https://doi.org/10.1007/11502760_3

[7] John Black, Shai Halevi, Hugo Krawczyk, Ted Krovetz, and Phillip Rogaway. 1999. UMAC: Fast and Secure Message Authentication. In *CRYPTO'99 (LNCS)*, Michael J. Wiener (Ed.), Vol. 1666. Springer, Heidelberg, 216–233. https://doi.org/10.1007/3-540-48405-1_14

[8] Carsten Bormann, Simon Lemay, Hannes Tschofenig, Klaus Hartke, Bilhanan Silverajan, and Brian Raymor. 2018. CoAP (Constrained Application Protocol) over TCP, TLS, and WebSockets. *RFC* 8323 (2018), 1–54.

[9] Sonja Bruckner, Sebastian Ramacher, and Christoph Striecks. 2023. Muckle+: End-to-End Hybrid Authenticated Key Exchanges. In *PQCrypto (Lecture Notes in Computer Science)*. Springer. to appear.

[10] Larry Carter and Mark N. Wegman. 1977. Universal Classes of Hash Functions (Extended Abstract). In *STOC*. ACM, 106–112.

[11] CCMB-2017-04-001 2017. *Common Criteria for Information Technology Security Evaluation*. Publication 3.1, Rev. 5. Common Criteria (CC).

[12] European Commission. 2023. *The European Quantum Communication Infrastructure (EuroQCI) Initiative*. http://web.archive.org/web/20230419230756/https://digital-strategy.ec.europa.eu/en/policies/european-quantum-communication-infrastructure-euroqci

[13] Airbus Secure Communications. 2023. *Towards a better approach for Quantum-Key-Distribution (QKD) Networks keymanagement*. https://web.archive.org/web/20230504083700/https://securecommunications.airbus.com/en/news/quantum-key-distribution-qkd-networks-key-management

[14] D. Dieks. 1982. Communication by EPR devices. *Physics Letters A* 92, 6 (Nov. 1982), 271–272.

[15] Benjamin Dowling, Torben Brandt Hansen, and Kenneth G. Paterson. 2020. Many a Mickle Makes a Muckle: A Framework for Provably Quantum-Secure Hybrid Key Exchange. In *Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020*, Jintai Ding and Jean-Pierre Tillich (Eds.). Springer, Heidelberg, 483–502. https://doi.org/10.1007/978-3-030-44223-1_26

[16] Chip Elliott and Henry Yeh. 2007. *DARPA quantum network testbed*. Technical Report. BBN TECHNOLOGIES CAMBRIDGE MA.

[17] eoPortal. 2023. *QUESS (Quantum Experiments at Space Scale) / Micius*. https://web.archive.org/web/20230218065358/https://www.eoportal.org/satellite-missions/quess

[18] Justin R. Erenkrantz, Michael M. Gorlick, Girish Suryanarayana, and Richard N. Taylor. 2007. From Representations to Computations: The Evolution of Web Architectures. In *Proceedings of the 6th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT International Symposium on Foundations of Software Engineering, 2007, Dubrovnik, Croatia, September 3-7, 2007*, Ivica Crnkovic and Antonia Bertolino (Eds.). ACM, 255–264.

[19] Olaf Bergmann et al. 2010—2023. *A CoAP (RFC 7252) implementation in C*. https://github.com/obgm/libcoap

[20] ETSI GS QKD 004 2020. *Quantum Key Distribution (QKD); Application Interface*. Group Specification v2.1.1. European Telecommunications Standards Institute (ETSI), Industry Specification Groups(ISG).

[21] ETSI GS QKD 014 2019. *Quantum Key Distribution (QKD); Protocol and data format of REST-based key delivery API*. Group Specification v1.1.1. European Telecommunications Standards Institute (ETSI), Industry Specification Groups(ISG).

[22] ETSI GS QKD 015 2022. *Control Interface for Software Defined Networks*. Group Specification v2.1.1. European Telecommunications Standards Institute (ETSI), Industry Specification Groups (ISG).

[23] ETSI GS QKD 016 2021. *Common Criteria Protection Profile Pair of Prepare and Measure Quantum Key Distribution Modules*. Group Specification Draft v0.6.2. European Telecommunications Standards Institute (ETSI), Industry Specification Groups(ISG).

[24] ETSI GS QKD 020 2023. *Protocol and data format of REST-based Interoperable Key Management System API*. Group Specification Draft v0.2.1. European Telecommunications Standards Institute (ETSI), Industry Specification Groups (ISG).

[25] Roy Thomas Fielding. 2000. *Architectural Styles and the Design of Network-based Software Architectures*. Ph.D. Dissertation. University of California.

[26] Roy T. Fielding, Mark Nottingham, and Julian F. Reschke. 2022. HTTP Semantics. *RFC* 9110 (2022), 1–194.

[27] FIPS PUB 140-2 2001. *Security requirements for cryptographic modules*. Standards Publication. National Institute of Standards and Technology (NIST) Federal Information Processing Standards (FIPS).

[28] Federal Office for Information Security (BSI). 2021. Quantum-safe cryptography – fundamentals, current developments and recommendations. (2021). https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/Brochure/quantum-safe-cryptography.pdf?__blob=publicationFile&v=4

[29] Federico Giacon, Felix Heuer, and Bertram Poettering. 2018. KEM combiners. In *Public-Key Cryptography–PKC 2018: 21st IACR International Conference on Practice and Theory of Public-Key Cryptography, Rio de Janeiro, Brazil, March 25-29, 2018, Proceedings, Part I 21*. Springer, 190–218.

[30] Frédéric Grosshans and Philippe Grangier. 2002. Continuous variable quantum cryptography using coherent states. *Physical review letters* 88, 5 (2002), 057902.

[31] Lov K. Grover. 1996. A Fast Quantum Mechanical Algorithm for Database Search. In *28th ACM STOC*. ACM Press, 212–219. https://doi.org/10.1145/237814.237866

[32] Bruno Huttner, Romain Alléaume, Eleni Diamanti, Florian Fröwis, Philippe Grangierand Hannes Hübel, Vicente Martin, Andreas Poppe, Joshua A. Slater, Tim Spiller, Wolfgang Tittel, Benoit Tranier, Adrian Wonfor, and Hugo Zbinden. 2022. Long-range QKD without trusted nodes is not possible with current technology. *npj Quantum Information* 8, 1 (2022), 1–5.

[33] ISO/IEC 27000-family 1995-2023. *Information security management systems*. International Standard. International Organization for Standardization (ISO).

[34] ITU-T Y.3800 2019. *Overview on networks supporting quantum key distribution*. Recommendation v1.0. International Telecommunication Union (ITU) Telecommunication Standardization Sector (ITU-T).

[35] ITU-T Y.3803 2020. *Quantum key distribution networks – Key management*. Recommendation v1.0. International Telecommunication Union (ITU) Telecommunication Standardization Sector (ITU-T).

[36] Leslie Lamport, Robert E. Shostak, and Marshall C. Pease. 1982. The Byzantine Generals Problem. *ACM Trans. Program. Lang. Syst.* 4, 3 (1982), 382–401.

[37] Diego R. López, Juan Pedro Brito, Antonio Pastor, Vicente Martín, C. Sánchez, D. Rincon, and Víctor López. 2021. Madrid Quantum Communication Infrastructure: a testbed for assessing QKD technologies into real production networks. In *OFC*. IEEE, 1–4.

[38] Miralem Mehic, Marcin Niemiec, Stefan Rass, Jiajun Ma, Momtchil Peev, Alejandro Aguado, Vicente Martín, Stefan Schauer, Andreas Poppe, Christoph Pacher, and Miroslav Voznák. 2020. Quantum Key Distribution: A Networking Perspective. *ACM Comput. Surv.* 53, 5 (2020), 96:1–96:41.

[39] Nithin Nagaraj, Vivek Vaidya, and Prabhakar G Vaidya. 2005. Re-visiting the One-Time Pad. *arXiv preprint cs/0508079* (2005).

[40] NIST SP 800-130 2013. *A Framework for Designing Cryptographic Key Management Systems*. Special Publication Revision 5. National Institute of Standards and Technology (NIST).

[41] NIST SP 800-130 2020. *Security and Privacy Controls for Information Systems and Organizations*. Special Publication Revision 5. National Institute of Standards and Technology (NIST).

[42] NIST SP 800-218 2022. *Secure Software Development Framework (SSDF)*. Special Publication Version 1.1. National Institute of Standards and Technology (NIST).

[43] NIST SP 800-57 2020. *Recommendation for Key Management*. Special Publication Part 1, Revision 5. National Institute of Standards and Technology (NIST).

[44] Timothy C Ralph. 1999. Continuous variable quantum cryptography. *Physical Review A* 61, 1 (1999), 010303.

[45] Eric Rescorla. 2018. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446.

[46] Eric Rescorla, Hannes Tschofenig, and Nagendra Modadugu. 2022. The Datagram Transport Layer Security (DTLS) Protocol Version 1.3. RFC 9147.

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

[48] Zach Shelby, Klaus Hartke, and Carsten Bormann. 2014. The Constrained Application Protocol (CoAP). *RFC* 7252 (2014), 1–112.

[49] Peter W. Shor. 1994. Algorithms for Quantum Computation: Discrete Logarithms and Factoring. In *35th FOCS*. IEEE Computer Society Press, 124–134. https://doi.org/10.1109/SFCS.1994.365700

[50] Ch Silberhorn, Timothy C Ralph, Norbert Lütkenhaus, and Gerd Leuchs. 2002. Continuous variable quantum cryptography: Beating the 3 dB loss limit. *Physical review letters* 89, 16 (2002), 167901.

[51] Mark N Wegman and J Lawrence Carter. 1981. New hash functions and their use in authentication and set equality. *Journal of computer and system sciences* 22, 3 (1981), 265–279.

[52] Mark N. Wegman and Larry Carter. 1981. New Hash Functions and Their Use in Authentication and Set Equality. *J. Comput. System Sci.* 22 (1981), 265–279.

[53] W. K. Wootters and W. H. Zurek. 1982. A single quantum cannot be cloned. *Nature* 299, 5886 (Oct. 1982), 802–803.

[54] Juan Yin, Yu-Huai Li, Liao Shengkai, Meng Yang, Yuan Cao, Liang Zhang, Jianyu Wang, Wen-Qi Cai, Wei-Yue Liu, Shuang-Lin Li, Rong Shu, Yong-Mei Huang, Lei Deng, Li Li, Qiang Zhang, Nai-Le Liu, Yu-Ao Chen, Chao-Yang Lu, Xiang-Bin Wang, and Jian-Wei Pan. 2020. Entanglement-based secure quantum cryptography over 1,120 kilometres. *Nature* 582 (06 2020), 1–5.