Annex I
Draft new Recommendation ITU-T Y.3814 (ex. Y.QKDN-ml-fra)

Quantum key distribution networks - functional requirements and architecture for machine learning enablement

Summary

QKDN is expected to maintain stable operations and meet the requirements of various cryptographic applications efficiently. Due to the advantages of machine learning (ML) related to autonomous learning, ML can help to overcome the challenges of QKDN in terms of quantum layer performances, key management layer performances and QKDN control and management efficiency. Based on the functional requirements and architecture of QKDN in [ITU-T Y.3801] and [ITU-T Y.3802], this recommendation is to specify one possible set of functional requirements and a possible architecture for ML-enabled QKDN (QKDNml), including the overview, the functional requirements, architecture and operational procedures of QKDNml.

Keywords
Functional architecture; functional requirements; machine learning (ML); procedure; quantum key distribution (QKD); QKD network (QKDN).

Table of Contents

Draft new Recommendation ITU-T Y.3814 (ex. Y.QKDN-ml-fra)

Quantum key distribution networks - functional requirements and architecture for machine learning enablement

1. Scope

This Recommendation specifies one possible set of functional requirements and a possible architecture for ML-enabled QKDN (QKDNml).

In particular, the Recommendation includes:

- Overview of QKDNml;
- Functional requirements of QKDNml;
- Functional architecture of QKDNml;
- Operational procedures of QKDNml.

This draft Recommendation specifies requirements for generic data collection. It does not specify the requirements for specific data related to Personal Identifiable Information (PII).

2. References

[ITU-T Y.3800] Recommendation ITU-T Y.3800 (2019), *Framework for Networks to support Quantum Key Distribution*.

[ITU-T Y.3801] Recommendation ITU-T Y.3801 (2020), *Functional requirements for quantum key distribution networks*.

[ITU-T Y.3802] Recommendation ITU-T Y.3802 (2020), *Functional architecture of the Quantum Key Distribution network*.

[ITU-T Y.3172] Recommendation ITU-T Y.3172 (2019), *Architectural framework for machine learning in future networks including IMT-2020*.

3. Terms and definitions

3.1. Terms defined elsewhere

This recommendation uses the following terms defined elsewhere:

3.1.1 **key manager (KM)** [ITU-T Y.3800]: A functional module located in a quantum key distribution (QKD) node to perform key management in the key management layer.

3.1.2 **machine learning (ML)** [ITU-T Y.3172]: processes that enable computational systems to understand data and gain knowledge from it without necessarily being explicitly programmed.

NOTE - Definition adapted from [b-ETSI GR ENI 004].

3.1.3 **machine learning function orchestrator (MLFO)** [ITU-T Y.3172]: a logical orchestrator that can monitor and manage the nodes in a machine learning pipeline.

3.1.4 **machine learning model** [ITU-T Y.3172]: model created by applying machine learning techniques with data to learn from.

NOTE 1 – A machine learning model is used to generate predictions on new (untrained) data.

NOTE 2 – A machine learning model may be encapsulated in a deployable fashion in the form of a software or hardware component.

NOTE 3 – Machine learning techniques include learning algorithms (e.g. learning the function that maps input data attributes to output data).

3.1.5 **machine learning pipeline** [ITU-T Y.3172]: a set of logical nodes, each with specific functionalities, that can be combined to form a machine learning application in a telecommunication network.

NOTE – The nodes are entities that are managed in a standard manner and can be hosted in a variety of network functions.

3.1.6 **machine learning sandbox** [ITU-T Y.3172]: an environment in which machine learning models can be trained, verified and their effects on the network analysed.

NOTE – A machine learning sandbox is designed to prevent a machine learning application from affecting the network, or to restrict the usage of certain machine learning functionalities.

3.1.7 **quantum key distribution (QKD)** [b-ETSI GR QKD 007]: Procedure or method for generating and distributing symmetrical cryptographic keys with information theoretical security based on quantum information theory.

3.1.8 **quantum key distribution link (QKD link)** [ITU-T Y.3800]: A communication link between two quantum key distribution (QKD) modules to operate the QKD.

NOTE – A QKD link consists of a quantum channel for the transmission of quantum signals, and a classical channel used to exchange information for synchronization and key distillation.

3.1.9 **quantum key distribution module (QKD module)** [ITU-T Y.3800]: A set of hardware and software components that implements cryptographic functions and quantum optical processes, including quantum key distribution (QKD) protocols, synchronization, distillation for key generation, and is contained within a defined cryptographic boundary.

NOTE – A QKD module is connected to a QKD link, acting as an endpoint module in which a key is generated. These are two types of QKD modules, namely, the transmitters (QKD-Tx) and the receivers (QKD-Rx).

3.1.10 **quantum key distribution network (QKDN)** [ITU-T Y.3800]: A network comprised of two or more quantum key distribution (QKD) nodes connected through QKD links.

NOTE – A QKDN allows sharing keys between the QKD nodes by key relay when they are not directly connected by a QKD link.

3.1.11 **quantum key distribution network controller (QKDN controller)** [ITU-T Y.3800]: A functional module, which is located in a quantum key distribution (QKD) network control layer to control a QKD network.

3.1.12 **quantum key distribution network manager (QKDN manager)** [ITU-T Y.3800]: A functional module, which is located in a quantum key distribution (QKD) network management layer to monitor and manage a QKD network.

3.1.13 **quantum key distribution node (QKD node)** [ITU-T Y.3800]: A node that contains one or more quantum key distribution (QKD) modules protected against intrusion and attacks by unauthorized parties.

NOTE – A QKD node can contain a key manager (KM).

3.2. Terms defined in this Recommendation

This chapter defines all the terms used in this recommendation.

3.2.1 **Machine learning-enabled quantum key distribution network (ML-enabled QKDN):** A quantum key distribution network (QKDN) that extends or enhances its functionalities enabled by machine learning (ML) capabilities to achieve different objectives.

NOTE – ML is an optional functionality for QKDN.

NOTE – Examples of different objectives are specified in [b-ITU-T Y-Suppl.70].

4. Abbreviations and acronyms

This chapters describes all the abbreviations and acronyms used in the recommendation.

C	Collector (ML pipeline)
D	Distribution (ML pipeline)
FCAPS	Fault, Configuration, Accounting, Performance and Security
KM	Key Manager
M	Model (ML pipeline)
ML	Machine Learning
MLFO	Machine Learning Function Orchestrator
P	Policy (ML pipeline)
PP	Pre-Processor (ML pipeline)
QKD	Quantum Key Distribution
QKDN	Quantum Key Distribution Network
QKDNml	ML-enabled QKDN
QL	Quantum Layer
RUL	Remaining Use Life
SRC	Source node
SINK	Sink node
XLMO	Cross Layer Management and Orchestration

5. Conventions

In this Recommendation:

The keywords "is required to" indicate a requirement which must be strictly followed and from which no deviation is permitted if conformance to this document is to be claimed.

The keywords "is recommended" indicate a requirement which is recommended but which is not absolutely required. Thus this requirement need not be present to claim conformance.

6. Overview

QKDN is a technology that extends the reachability and availability of QKD, which is stated in [ITU-T Y.3800]. It is comprised of two or more QKD nodes connected through QKD links. In a QKDN, two or more designated parties in a user network can share the keys for various cryptographic applications. QKDN is expected to maintain stable operations and meet the requirements of various cryptographic applications in an efficient way. However, when the QKDN becomes large-scale and complex, QKDN performance optimization in quantum layer, key management layer, QKDN control and management layer can be challenging.

In detail, to improve QKDN performances, QKDN faces the following important challenges:

1) Without the awareness of sudden QKDN performance deterioration in advance, high cost (e.g. time cost, labour cost) and instability of QKDN will increase.

2) For the large amount of heterogenous data in QKDN, there is difficulty to accurately perceive the needed and valuable information for use, which will affect QKDN performances.

3) Since the requirements of cryptographic applications vary (e.g. different security requirements) and a large number of cryptographic applications arrive and leave dynamically, it is difficult to schedule the QKDN resources for cryptographic applications with the limited resources.

To overcome the above challenges, applying ML technology into QKDN is a promising solution. ML can extract implicit relationships between input and output data, and use this learned mapping to analyse new data. It has been applied to networking field which can intelligently learn various network environments and react to dynamic situations ([ITU-T Y.3170]). In recent years, ML technologies based on neural networks have seen many developments in both hardware and software, and they have attracted attention from both the academia and the industry. There is also an increasing number of new low-power devices implementing on-board acceleration chips for neural networks.

There can be many benefits of enabling ML in QKDN. Application use cases of enabling ML to achieve different objectives in QKDN have been specified in [b-ITU-T Y-Suppl.70]. In the quantum layer of QKDN, ML can be applied to realize quantum channel performance prediction, QKD system parameter optimization and RUL prediction of components in a QKD system; in the key management layer of QKDN, ML can be applied to realize intelligent key formatting, key storage management, and suspicious behavior detection; in the control and management layers of QKDN, ML can be applied in key relay routing and fault prediction to improve control and management efficiency.

With the advantages of ML especially related to autonomous learning, ML can support overcoming the challenges of QKDN performance optimization in quantum layer, key management layer, QKDN control and management layer. Thus, an ML-enabled QKDN (QKDNml) can accelerate the optimization of QKDN by extending or enhancing QKDN functionalities. Note, however, that ML is an optional functionality for QKDN according to the functional requirements and architecture of QKDN in [ITU-T Y.3801] and [ITU-T Y.3802]. To enable ML for QKDN, this Recommendation specifies one possible set of functional requirements and a possible architecture for QKDNml, including overview, the functional requirements, architecture and operational procedures of QKDNml.

7. Functional requirements of QKDNml

The additional high-level and functional requirements related to ML are specified in this subclause to extend high-level requirements defined in [ITU-T Y.3801].

7.1 High-level requirements of QKDNml

The high-level requirements of QKDNml are as follows.

- It is required to support the configuration, management and orchestration for ML-related functional components;
- It is required to support the data collection, data pre-processing, data repository, modelling and training functions;
- It is required to support ML models for different objectives in QKDNml;
- It is recommended to use the existing reference points defined in [ITU-T Y.3802] and extend them with reference points specific to ML capabilities.
- It is recommended to use declarative specifications for specifying different objectives in QKDNml.

7.2 Functional requirements of QKDNml data collection

The QKDN data can be collected from the quantum layer, key management layer, QKDN control layer, QKDN management layer, service layer and user network management layer either passively or actively. The functional requirements of QKDNml data collection are as follows.

- It is required to be able to collect both static and dynamic QKDN data from quantum layer, key management layer, QKDN control layer and QKDN management layer.

  NOTE 1 - Static QKDN data can be collected from quantum layer such as parameters of QKD modules, history status information of QKD modules; dynamic QKDN data can be collected from quantum layer such as quantum bit error rates, key generation rates, performance information of QKD modules.

  NOTE 2 - Static QKDN data can be collected from key management layer such as history key management layer data set; dynamic QKDN data can be collected from key management layer such as status of key storage and status of key authentication.

  NOTE 3 - Static QKDN data can be collected from QKDN control layer such as parameters of QKD modules and history status information of QKD modules; dynamic QKDN data can be collected from QKDN control layer such as routing and rerouting information and status of resource allocation.

  NOTE 4 - Static QKDN data can be collected from QKDN management layer such as history data of fault management, history data of configuration and history data of security management; dynamic QKDN data can be collected from QKDN management layer such as multi-layer resource usage data and multi-layer performance data.
- It is recommended to collect both static and dynamic QKDN data from service layer and user network management layer.

  NOTE 1 - Static QKDN data can be collected from service layer such as history cryptographic application information; dynamic QKDN data can be collected from service layer such as current cryptographic applications information.

  NOTE 2 - Static QKDN data can be collected from user network management layer such as history user requirements; dynamic QKDN data can be collected from user network management layer such as current user requirements.

7.3 Functional requirements for QKDNml data pre-processing and repository

The functional requirements of QKDNml data pre-processing and repository are as follows.

- It is required to perform extract-transform-load and transform the collected multi-source, heterogeneous QKDN raw data into understandable, unified and easy-to-use structures.
- It is required to clean and filter noisy data from the collected heterogeneous QKDN raw data.
- It is recommended to normalize and unify the data format of the collected heterogeneous QKDN raw data for further storage and analysis.
- It is recommended to store the heterogeneous QKDN pre-processed data.
- It is recommended to store catalogues and data sets for ML models.

7.4 Functional requirements for QKDNml modelling and training

The functional requirements of QKDNml modelling and training are as follows.

- It is required to support ML models based on the pre-processed QKDN data and the specified objective.
- It is required to support ML model training and model updates while preventing impact on QKDNml.
- It is recommended to train ML models based on the available pre-processed QKDN data for the specified objective in QKDNml.

8. Functional architecture of QKDNml

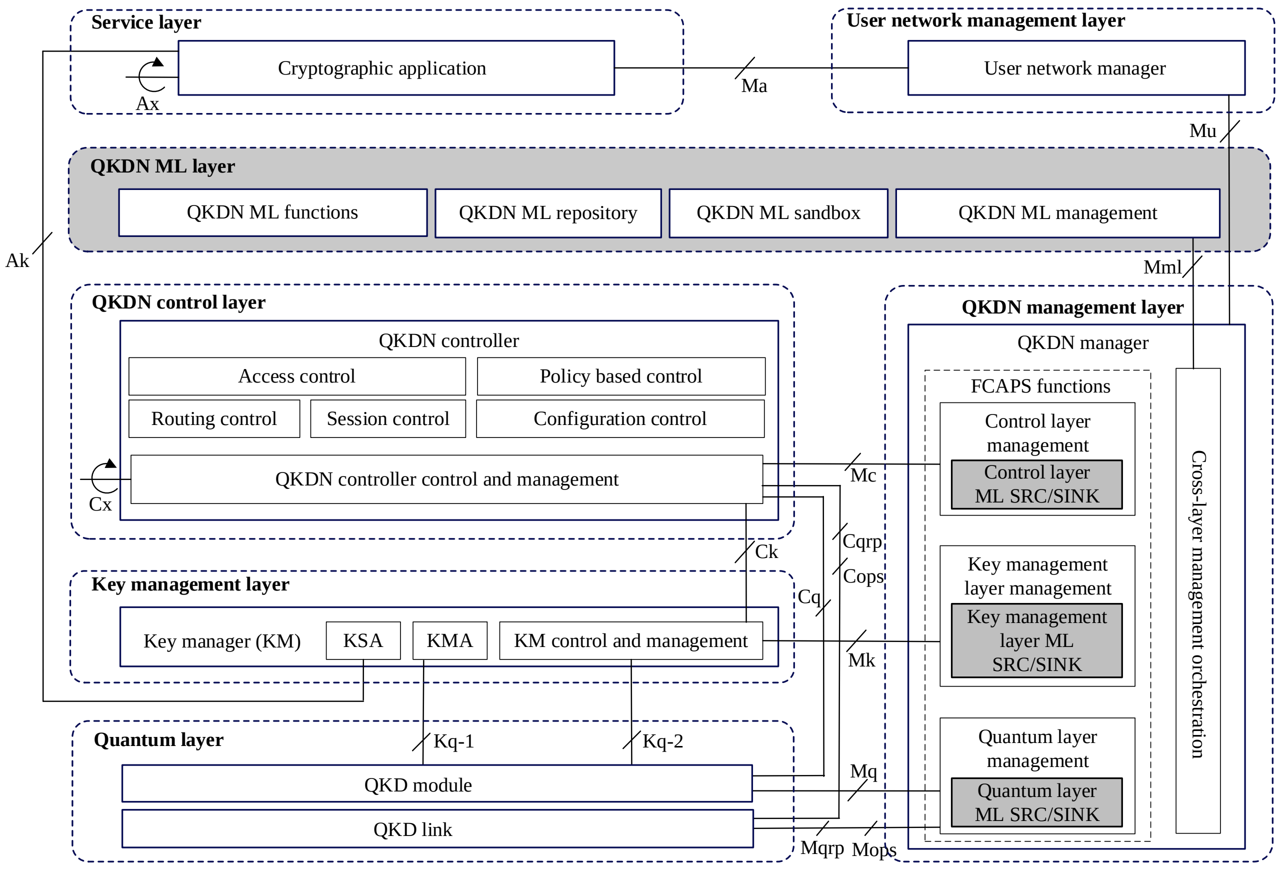

Fig. 8.1. Functional architecture model of QKDNml

To enable ML capabilities for QKDN, a new QKDN ML layer is introduced with QKDN layer specific ML capabilities. The QKDN ML layer consists of QKDN ML functions, QKDN ML repository, QKDN ML sandbox, and QKDN ML management. The QKDN management layer also specifies QKDN layer specific ML SRCs and SINKs. ML pipelines in QKDNml can be constructed to realize ML applications for different objectives. The interaction between QKDN ML layer and QKDN management layer is made through a newly introduced reference point Mml between QKDN ML management in the QKDN ML layer and the cross-layer management orchestration in the QKDN management layer. The functional architecture model of QKDNml is specified in Fig. 8.1.

8.1 QKDN ML layer in QKDNml

There are QKDN ML functions, QKDN ML repository, QKDN ML sandbox ([ITU-T Y.3172]), and QKDN ML management in the QKDN ML layer, which are responsible for configuration, management and orchestration for ML-related functional components in QKDNml.

8.1.1 QKDN ML functions

The QKDN ML functions support a set of functional elements in a ML pipeline subsystem including collector (C), pre-processor (PP), model (M), policy (P) and distributor (D).

– **C:** responsible for collecting QKDN data from one or more SRC(s). The C may have the capability to configure SRC nodes. Such configurations may be used to control the nature of data, its granularity and periodicity while it is generated from SRCs.
– **PP:** responsible for cleaning, aggregating, normalizing or performing any other PP of heterogeneous data that should be in a suitable form so that the M can consume it.
– **M:** responsible for deploying the trained ML models for different objectives in QKDNml.
– **P:** responsible for making P decisions in QKDNml based on the results of the M.

– **D:** responsible for identifying the SINK(s) and distributing the P decisions to the corresponding SINK(s) in QKDNml.

8.1.2 QKDN ML repository

ML repository function is to store various data sets collected from ML SRCs, pre-processed by PP, generated by ML models, and catalogues of ML models, etc. It can be used by QKDN ML management, ML functions, and ML sandbox.

8.1.3 QKDN ML sandbox

A QKDN ML sandbox is an isolated domain that allows the hosting of separate ML pipelines to train, test and evaluate them before deploying them in a QKDN. The QKDN ML sandbox subsystem allows network operators to study the effect of ML outputs before deploying them on live QKDNs. For training or testing, the QKDN ML sandbox can use data generated from a simulated ML underlay QKDN.

8.1.4 QKDN ML management

This QKDN ML management includes intent and MLFO.

– **Intent in QKDNml:** Intent is a declarative description ([ITU-T Y.3172]) in QKDNml. The intent provides a basis for mapping ML use cases to different QKDN layers.

  NOTE – Intent is a declarative description which is used to specify a ML application. Intent does not specify any technology-specific network functions to be used in the ML application and provides a basis for mapping ML use cases to diverse technology-specific instantiations. Intent can use a meta language specific for machine learning to define ML applications. ([ITU-T Y.3172])

– **MLFO in QKDNml:** MLFO has functionalities that manage and orchestrate the nodes of the ML pipelines and ML sandbox based on the intent or dynamic conditions in QKDNml. The MLFO provides chaining functionality, i.e., connecting ML nodes together to form an ML pipeline. It also supports the ML model selection for different objectives.

  NOTE – For example, chaining can be used to connect an SRC specific to the quantum layer with the ML functions in the QKDN ML layer. The MLFO determines the chaining considering the constraints (e.g., timing constraints for prediction). ([ITU-T Y.3172])

8.2 ML pipeline SRC/SINK in QKDNml

An ML pipeline in QKDNml is a set of logical nodes, each with specific functionalities that can be combined to form an ML application in QKDNml. The ML pipeline in QKDNml has three parts including SRCs, ML functions and SINKs. The ML functions are able to collect input data from SRCs ([ITU-T Y.3172]) in different QKDN layers. The SINK, as the target of the ML output ([ITU-T Y.3172]), can be the elements in quantum layer, key management layer and QKDN control and management layers. The ML pipeline SRCs and SINKs are managed in FCAPS function in the QKDN management layer. More details related to ML pipeline can be found in [ITU-T Y.3172].

8.2.1 SRCs in QKDNml

The SRCs in QKDNml are the source nodes of QKDN data that can be used as input to the ML functions in QKDNml. The types of SRCs include:

– **Quantum layer ML SRC:** responsible for reporting the data (static, dynamic) from QKD modules and links in the quantum layer to QKDN ML functions;
– **Key management layer ML SRC:** responsible for reporting the data (static, dynamic) from the key manager in the key management layer to QKDN ML functions;
– **Control layer ML SRC:** responsible for reporting the data (static, dynamic) from the QKDN controller in the QKDN control layer to QKDN ML functions.

8.2.2 SINKs in QKDNml

The SINKs are the targets of the ML output in QKDNml on which actions are taken. The types of SINKs can include:

– **Quantum layer ML SINK:** represents that the QKD module or the QKD link is the target of configurations (as a result of ML pipeline execution) in the quantum layer;

  NOTE - ML output is applied to QKD modules or QKD links to optimize quantum layer performances of QKDN. The use cases can include ML-based QKD system parameter optimization, ML-based quantum channel performance prediction and ML-based RUL prediction of components in a QKD system ([b-ITU-T Y-Suppl.70]).

– **Key management layer ML SINK:** represents that key manager is the target of configurations in the key management layer;

  NOTE - ML output is applied to the key manager in the key management layer to optimize key management efficiency and stability. Three use cases are ML-based key formatting, ML-based key storage management, and ML-based suspicious behavior detection in the key management layer ([b-ITU-T Y-Suppl.70]).

– **Control layer ML SINK:** represents that QKDN controller is the target of configurations in the QKDN control layer.

  NOTE – ML output is applied to the QKDN controller in the QKDN control layer to improve QKDN control efficiency. Two use cases are ML-based data collection and data pre-processing and ML-based routing ([b-ITU-T Y-Suppl.70]).

8.3 Reference points

Most of the reference points in Figure 8.1 have been defined in [ITU-T Y.3802]. This Recommendation defines the newly added one and presents the existing ones related to ML functionalities.

NOTE –The new, extended ML enabled functions are implemented by extending the interaction information of reference points.

The newly added reference point is:

- **Mml:** a reference point connecting QKDN ML management and QKDN manager. It is responsible for exchanging the intent information and the management and orchestration information of MLFO between the QKDN ML management and the QKDN manager.

The existing reference points in [ITU-T Y.3802] related to ML include:

- **Mc:** a reference point connecting the QKDN manager and a QKDN controller control and management function in a QKDN controller. It is responsible for the QKDN manager to collect the data from the QKDN controller for ML functions and apply ML output on the QKDN controller.

- **Mk:** a reference point connecting the QKDN manager and a KM control and management function in a KM. It is responsible for the QKDN manager to collect the data from KM for ML functions and apply ML output on the KM.

- **Mq:** a reference point connecting the QKDN manager with a QKD module control and management function in a QKD module. It is responsible for the QKDN manager to collect the data from QKD modules for ML functions and apply ML output on QKD modules.

- **Mqrp, Mops:** reference points connecting the QKDN manager and the QKD link. They are responsible for the QKDN manager to collect the data from QKD links for ML functions and apply ML output on QKD links.

9. Operational procedures of QKDNml

In QKDNml, QKDN functionalities are extended or enhanced by enabling ML capabilities for different objectives. Fig. 9.1 shows a general operational procedure of QKDNml for a specific objective.

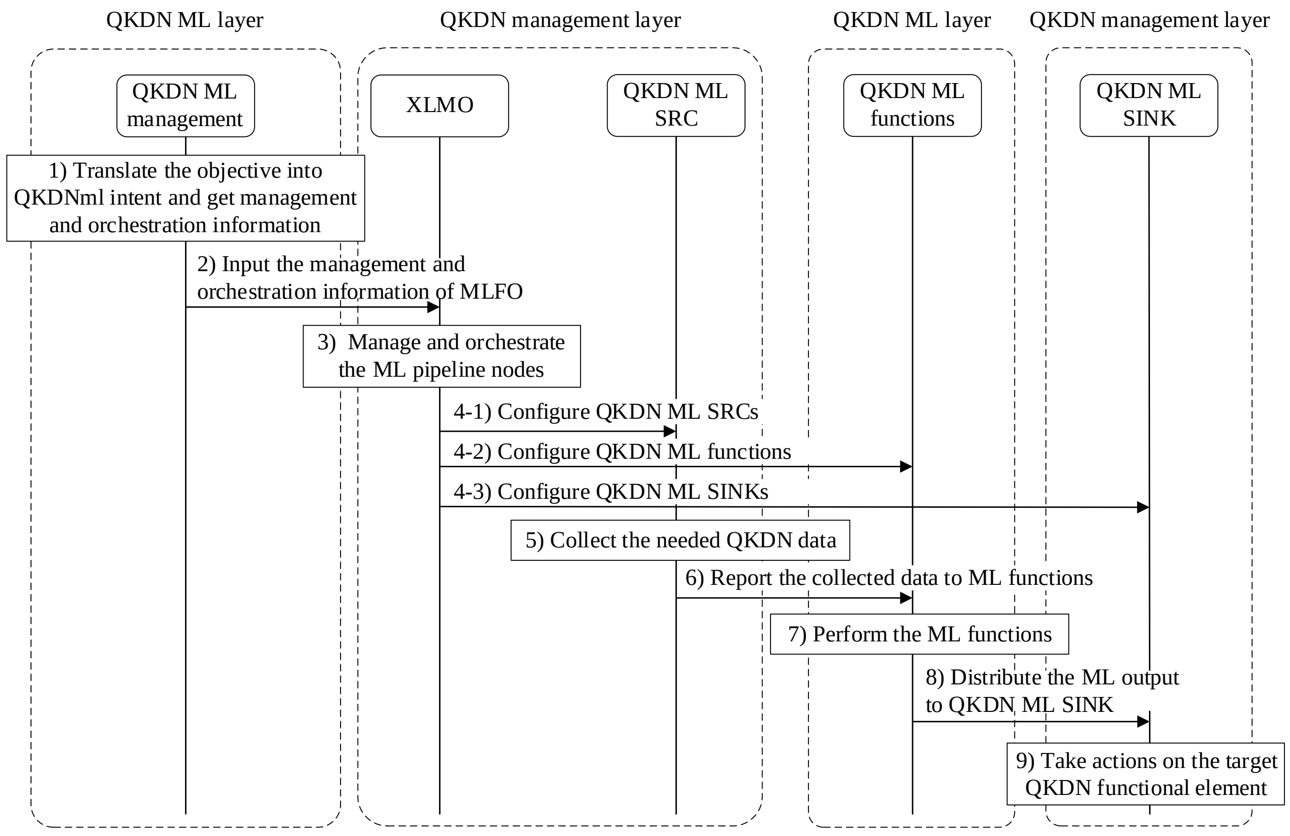
QKDN ML layer
QKDN management layer
QKDN ML layer
QKDN management layer
QKDN ML management
XLMO
QKDN ML SRC
QKDN ML functions
QKDN ML SINK
1) Translate the objective into QKDNml intent and get management and orchestration information
2) Input the management and orchestration information of MLFO
3) Manage and orchestrate the ML pipeline nodes
4-1) Configure QKDN ML SRCs
4-2) Configure QKDN ML functions
4-3) Configure QKDN ML SINKs
5) Collect the needed QKDN data
6) Report the collected data to ML functions
7) Perform the ML functions
8) Distribute the ML output to QKDN ML SINK
9) Take actions on the target QKDN functional element

Fig. 9.1. A general operational procedure of QKDNml

1) QKDN ML management translates the objective of the ML application in QKDNml into QKDNml intent, which is a declarative description used to specify the objectives of extending or enhancing the QKDN functionalities using ML. The intent is the information for technology-specific implementation.

2) QKDN ML management inputs the management and orchestration information of MLFO to XLMO (Cross-layer management orchestration) in the QKDN manager using the reference point Mml.

3) XLMO manages and orchestrates the ML pipeline nodes based on the intent or dynamic network conditions.

4) XLMO configures QKDN ML SRCs and ML SINKs using the reference points among Mc, Mk, Mq, Mqrp, Mops and Mu; XLMO configures QKDN ML functions using the reference point Mml.

5) QKDN ML SRCs collect the needed QKDN data from QKDN layers.

6) QKDN ML SRCs report the collected QKDN data to the QKDN ML functions.

7) QKDN ML functions in the QKDN ML layer are performed.

   NOTE – In the ML functions, the C collects the data reported from SRCs to PP. PP cleans the collected QKDN data by removing noisy data and transforms the cleaned data into a unified data format. The pre-processed QKDN data is transferred to M. The intent is performed by the deployed M selected by the MLFO. The outputs of the ML model are transferred to the P. Given

ML output of M, a QKDN policy decision can be made by P. The policy decision results are transferred to D.

8) D distributes the ML output to the QKDN ML SINKs instantiated by the components in QKDN layers corresponding to the intent in QKDNml.

9) QKDN ML pipeline SINK takes actions on the target QKDN functional element.

Appendix I

Use case of the operational procedures for ML-enabled quantum channel performance prediction

(This appendix does not form an integral part of this Recommendation.)

During the QKD process, the noise in the quantum channel will reduce the quality of the quantum channel and cause low key rate. The use case of ML-enabled quantum channel performance prediction ([b-ITU-T Y-Suppl.70]) is shown. Firstly, the quantum channel related data and the corresponding quantum channel performance are collected through quantum channel measurement for ML model training and testing. Then, with the trained ML model, the quantum channel performance can be predicted based on the current input quantum channel related data. Lastly, according to the predicted channel performance, feedback and adjustment can be finished in advance to improve the channel environment and reduce unnecessary loss caused by key rate decreases. The detailed operational procedures are as follows.

1) The intent translated by QKDN ML management is used to specify the quantum channel performance prediction, and then is input into the MLFO to get management and orchestration information. The quantum channel performance prediction is to predict the future quantum channel performance under different channel noise environments, so that measures can be taken in advance based on the predictions to improve the channel environment and make the quantum channel in an optimal performance state.

2) QKDN ML management inputs the management and orchestration information of MLFO to XLMO in the QKDN manager using the reference point Mml. The information specifies managing and orchestrating the ML pipeline subsystems to realize ML-based quantum channel performance prediction. The information can include the need to configure the Quantum layer ML SRC, QKDN ML functions and Quantum layer ML SINK.

3) XLMO manages and orchestrates the ML pipeline nodes for connecting ML pipeline nodes together to form an ML pipeline including Quantum layer ML SRC, QKDN ML functions and Quantum layer ML SINK.

4) XLMO configures Quantum layer ML SRC and Quantum layer ML SINK using the reference points Mq; XLMO configures QKDN ML functions using the reference point Mml.

5) Quantum layer ML SRC collects the quantum channel parameters in quantum layer.

6) Quantum layer ML SRC reports the collected QKDN data to the QKDN ML functions.

7) QKDN ML functions in the QKDN ML layer are performed.

   NOTE 1 - C transfers the collected data to the PP, which includes the quantum-channel-performance-related parameters, such as QBER of quantum channel, the SPD photon detection output counter and code formation rates under different noise environments.

   NOTE 2 - PP cleans the collected QKDN data by removing noisy data and transforms the cleaned data into a unified data format. The pre-processed QKDN data is transferred to M after intelligent analysis.

   NOTE 3 - Quantum channel performance prediction is performed by the ML models in the M. The outputs of the ML model are the predicted quantum channel performance values, which are transferred to the P.

   NOTE 4 - Given the predicted quantum channel performance, a Q policy decision is made by the P to minimize impacts when the output of ML is applied to a live network. The policy decision results are transferred to the D.

8) D distributes the ML output related to the predicted channel performance to the Quantum layer ML SINK instantiated by the QKD modules in the quantum layer.

9) According to the predicted channel performance, feedback and adjustment can be finished by Quantum layer ML SINK to improve the channel environment and reduce unnecessary loss caused by key rate decreases.

Bibliography

[b-ETSI GR ENI 004]	ETSI GR ENI 004 V1.1.1 (2018), *Experiential Networked Intelligence (ENI); Terminology for Main Concepts in ENI*.
[b-ETSI GR QKD 007]	ETSI GR QKD 007 (2018), *Quantum Key Distribution (QKD); Vocabulary*.
[b-ITU-T Y-Suppl.70]	Supplement 70 to ITU-T Y.3800-series Recommendations (2021), *Quantum key distribution networks - Applications of machine learning*.

____