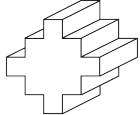


QKDNETSIM demonstration

Dr Miralem Mehic



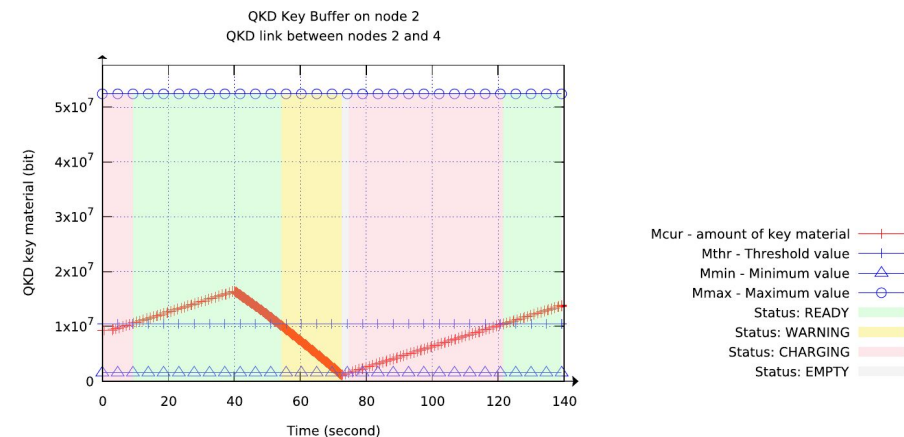
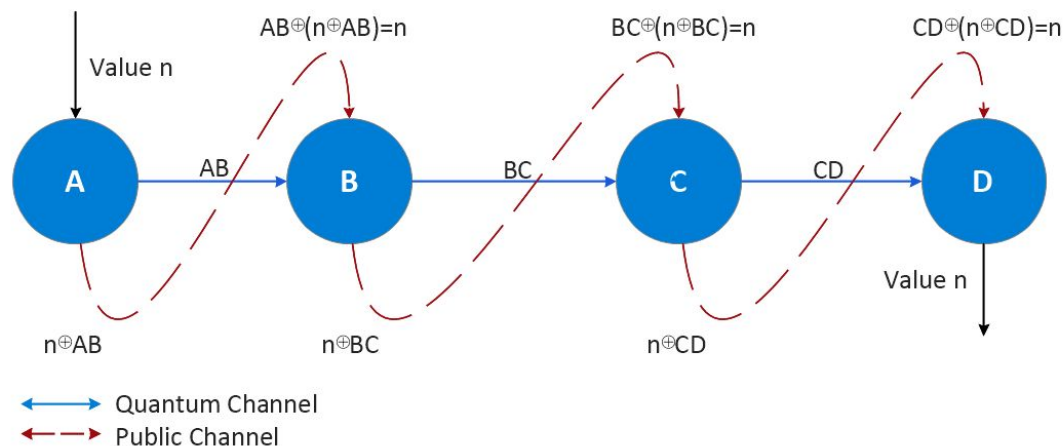
VSB TECHNICAL
UNIVERSITY
OF OSTRAVA

OPEN  QKD

GÉANT Infoshare: Quantum Key Distribution simulation
13 October 2021

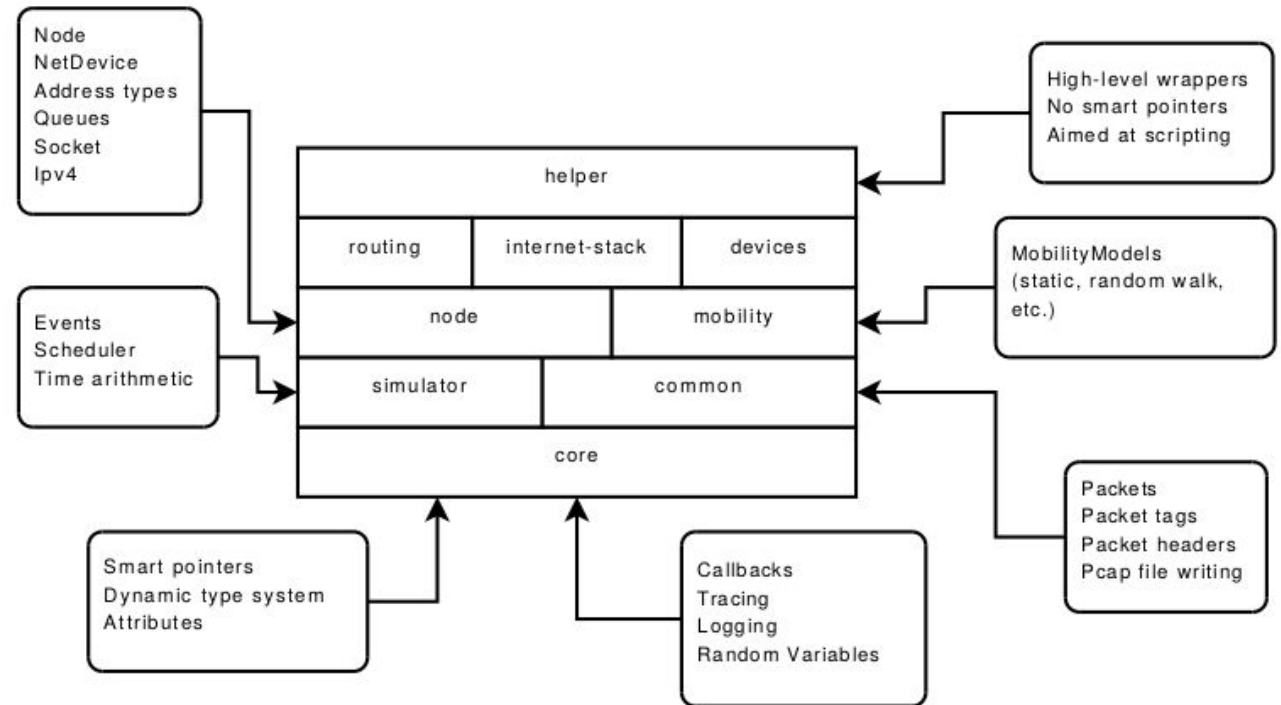
Quantum Key Distribution Network Simulation Module (QKDNetSim)

- Developed in the Network Simulator NS-3
- Version 1.0 available at <http://www.qkdnetsim.info/>
- How to use?
 - Virtual Machine (<http://www.qkdnetsim.info/virtual-machine/>)
 - Documentation (<http://www.qkdnetsim.info/doc/models/build/html/qkd.html>)
 - Doxygen API (http://www.qkdnetsim.info/doc/html/group__qkd.html)
 - Examples (<http://www.qkdnetsim.info/documentation/examples/>)
- Versions 2.0 and 2.1 under heavy development
 - they are currently not publicly available



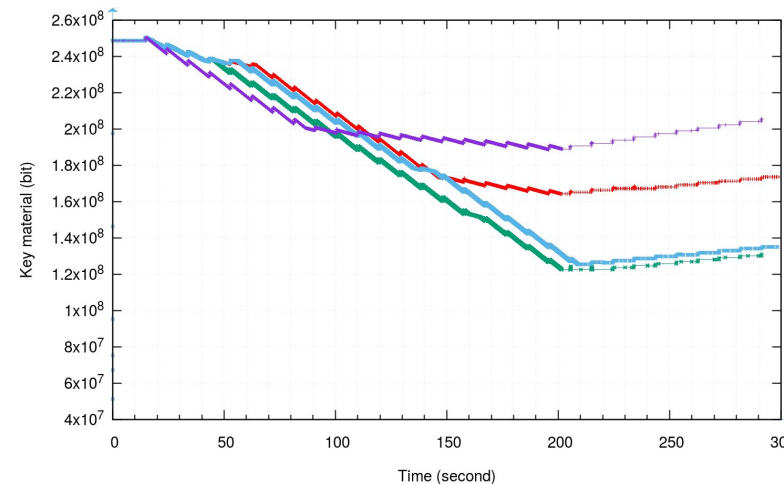
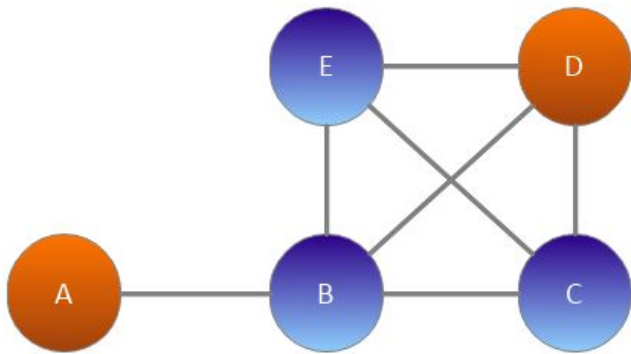
Quantum Key Distribution Network Simulation Module (QKDNetSim)

- Why NS-3?
 - Network Simulator of version 3 (NS-3) is an open-source software which is licensed under GNU GPLv2 and welcomes developers in contributing code from across academia, industry, and government.
- NS-3 releases - <https://www.nsnam.org/releases/>
 - ns-3.36 in preparation
 - ns-3.35 October 2021
 - ns-3.34 July 2021
 - ns-3.33 January 2021
 - ns-3.32 October 2020
 - ns-3.31 June 2020
 - ns-3.30 August 2019
 - ns-3.29 September 2018



Quantum Key Distribution Network Simulation Module (QKDNetSim)

- QKDNetSim is a unique simulation tool for QKD network with an unlimited number of QKD network nodes and links with maximum simplification of quantum channel
- Focus is placed on key usage considering:
 - network performance measurement and estimation,
 - quality of service (QoS) and network traffic management,
 - integration of QKD with other network technologies (SDN, IoT, WiFi, LTE, 5G, multipath routing, IPv6 and other)

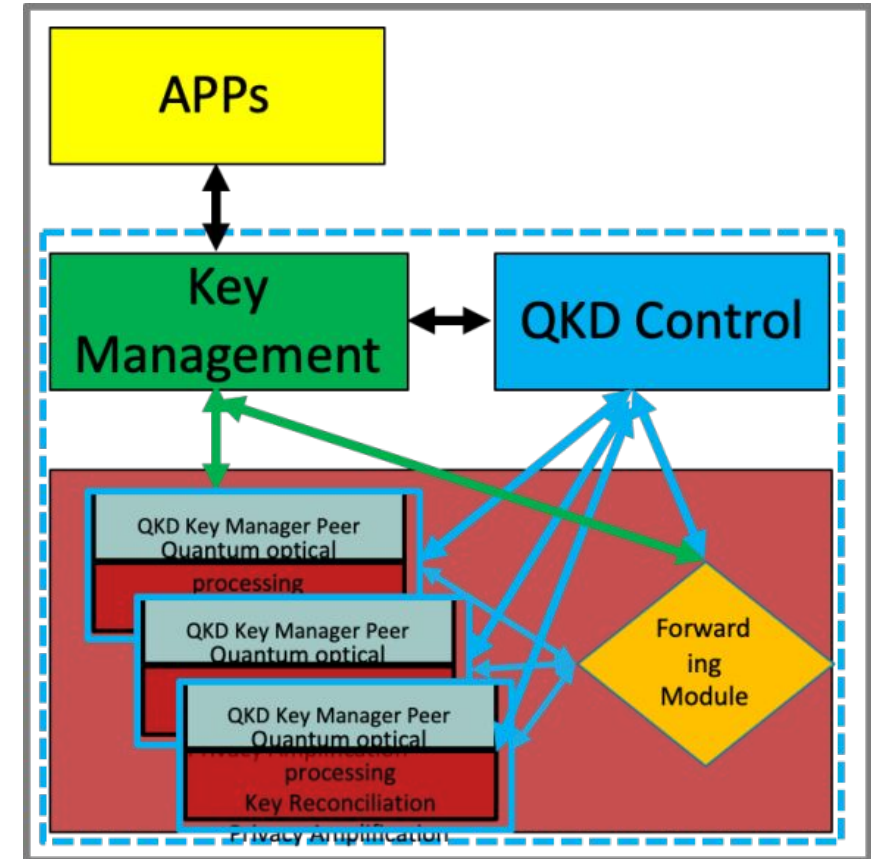


QKDNetSim Total Graph shows the overall consumption and generation of key material. Occasional increases of curves show the generation of new key material while the use of key material causes discharge of QKD buffers (curve decreases).

AODV - Amount of key material
DSDV - Amount of key material
OLSR - Amount of key material
OSPFv2 - Amount of key material

QKDNetSim structure

- How to organize QKD simulator?
- What are the main components?
- A minimal set of components of QKD node is defined in [1]
 - **Key Manager**
 - “A set of node-level Key Managers would become the network of the Key Management System”
 - **QKD Control (and management)**
 - “has knowledge of the status of the network and is able to control the QKD systems and their connections”
 - **QKD systems/modules**
 - Alice/Bob QKD devices
 - **Forwarding module**
 - “in charge of multi-hop key relaying”



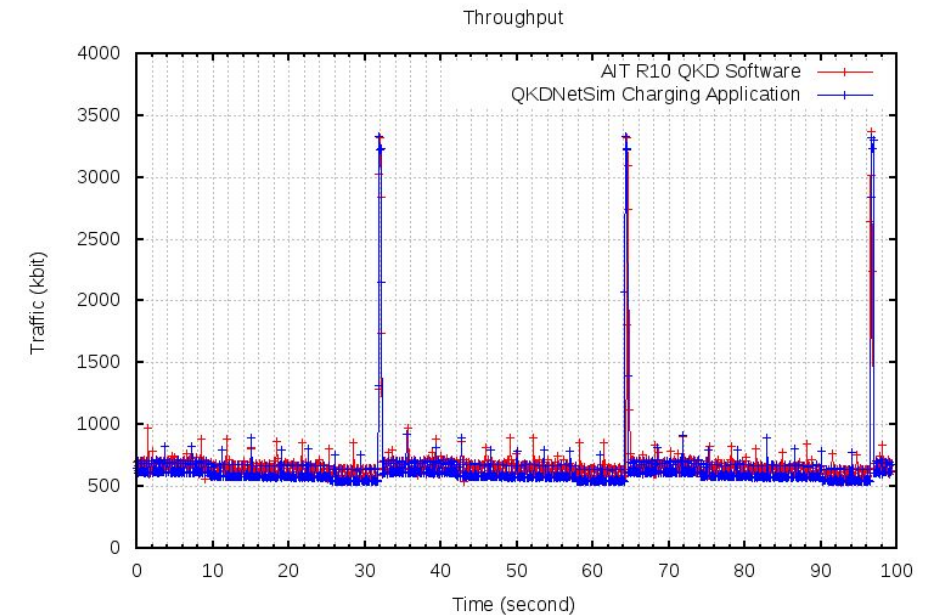
[1] V. Martin et al., “Quantum technologies in the telecommunications industry,” EPJ Quantum Technol., vol. 8, no. 1, p. 19, Dec. 2021, doi: 10.1140/epjqt/s40507-021-00108-9.

QKDNetSim v2.0 structure

- QKDNetSim v2.0 structure is motivated by ETSI 004 and ETSI 014 specifications
- It implements following components:
 - **QKD keys** - *elementary class used to describe the established QKD keys (timestamp, size, ID, meta-data)*
 - **QKD buffers** - *QKD key material storage/container/buffer*
 - **QKD post-processing application** - *an application that mimics post-processing network traffic and adds new QKD keys to a QKD buffer*
 - **QKD encryptor (SAE)** - *to perform encryption, decryption, authentication, authentication-check*
 - **QKD application** - *end-user application that generates traffic to be encrypted/authenticated*
 - **QKD manager (KMS)** - *central entity that connects applications and other components*
 - **QKD control** - *application to control QKD entities (power off, reboot, install and other)*

QKD post-processing application

- QKDNetSim implements simple application for generation of QKD key material
 - It credibly imitates the traffic generated by the real-world post-processing applications such as AIT R10 QKD software [2]
 - Implemented in ping-pong approach
 - It takes following input values:
 - Average key size (bits)
 - Average QKD key rate (bps)
 - Average data packet size - packets used in post-processing (bytes)
 - Average data traffic rate of post-processing
 - Excessive traffic (postprocessing and user applications) can lead to network congestion [3]

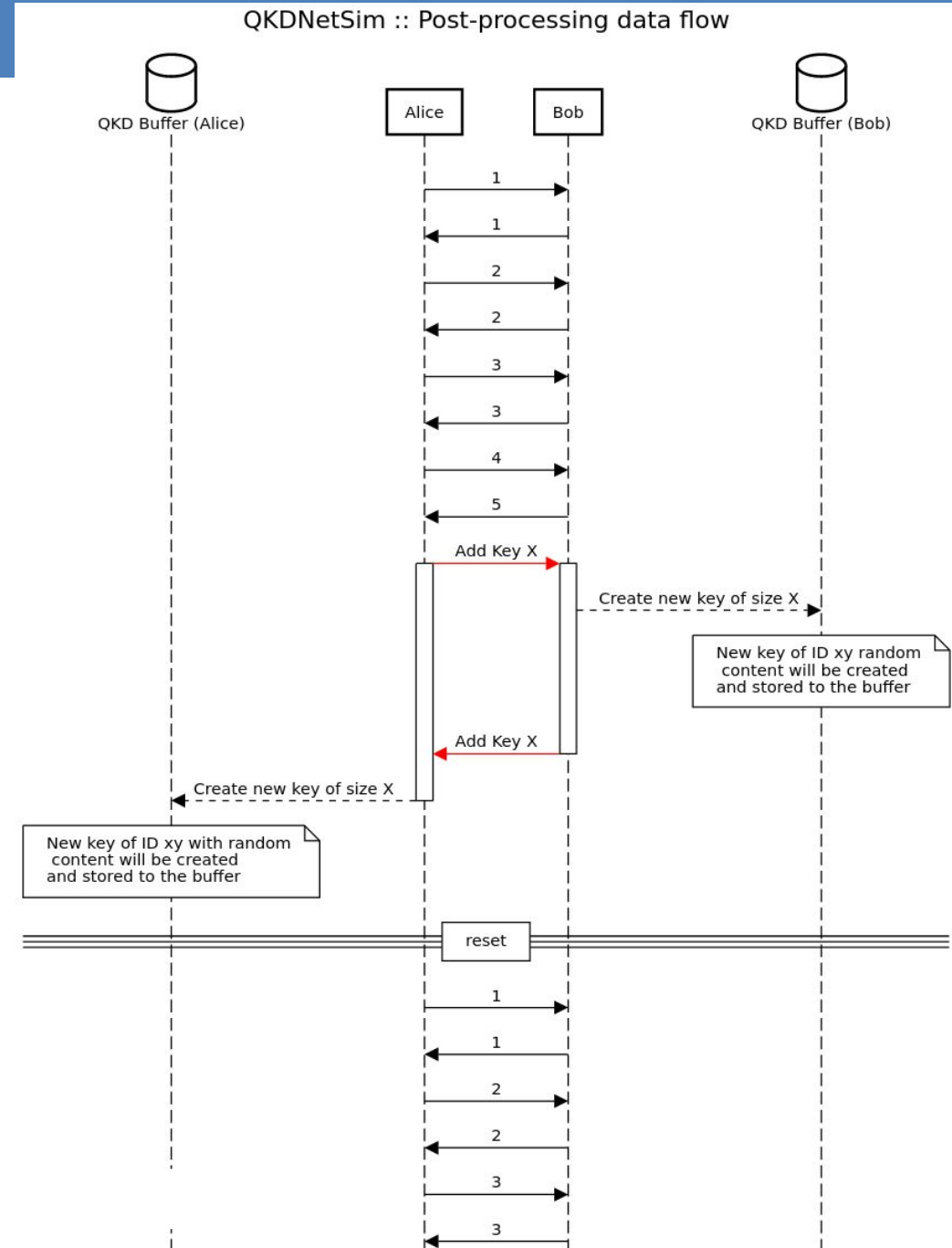


[2] Maurhart, O. et al.: New release of an open source QKD software: design and implementation of new algorithms, modularization and integration with IPsec. In: Qcrypt 2013

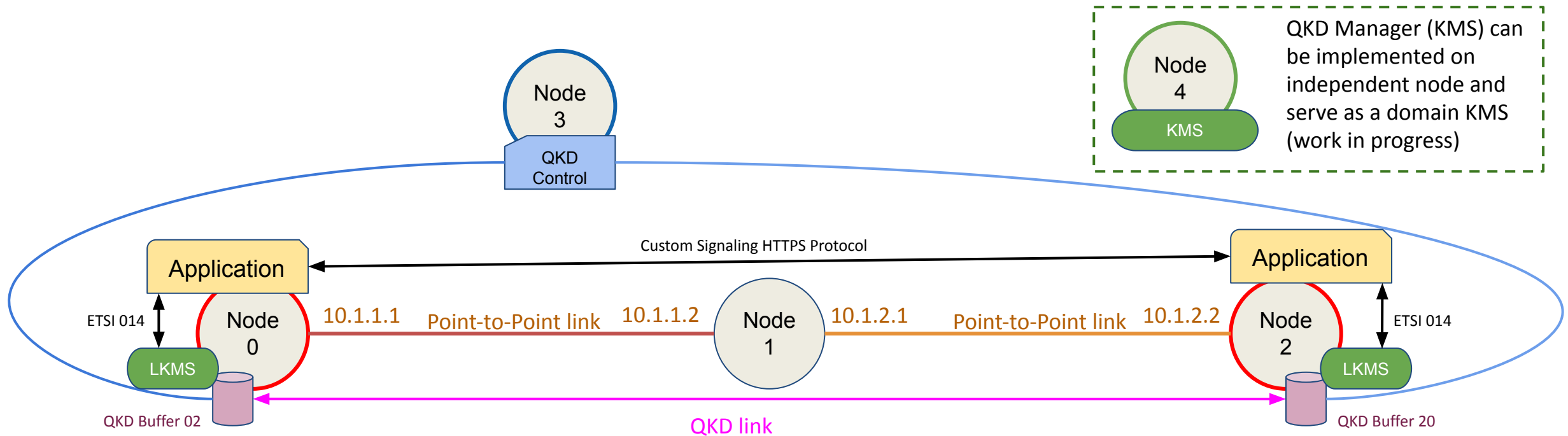
[3] Mehic, M., et al.: Analysis of the public channel of quantum key distribution link. *IEEE Journal of Quantum Electronics* 53.5 (2017): 1-8.

```
//Example of c++ code
ApplicationContainer postprocessingApplications;
postprocessingApplications.Add(
    QAHelper.InstallPostProcessing(
        n.Get(0), //srcNode
        n.Get(2), //dstNode
        InetSocketAddress (i0i1.GetAddress(0), 102), //srcAddr
        InetSocketAddress (ili2.GetAddress(1), 102), //dstAddr
        8092, //length of key to be added to QKD buffer (bit)
        DataRate ("50kbps"), //average QKD key rate
        100, //average data packet size (bytes)
        DataRate ("1Mbps") //average data traffic rate
    )
);
```

- Instead of implementing multiple QKD protocols (B84, B92, CV...) a simple post-processing application is implemented to mimic QKD establishment process.



QKDNetSim example

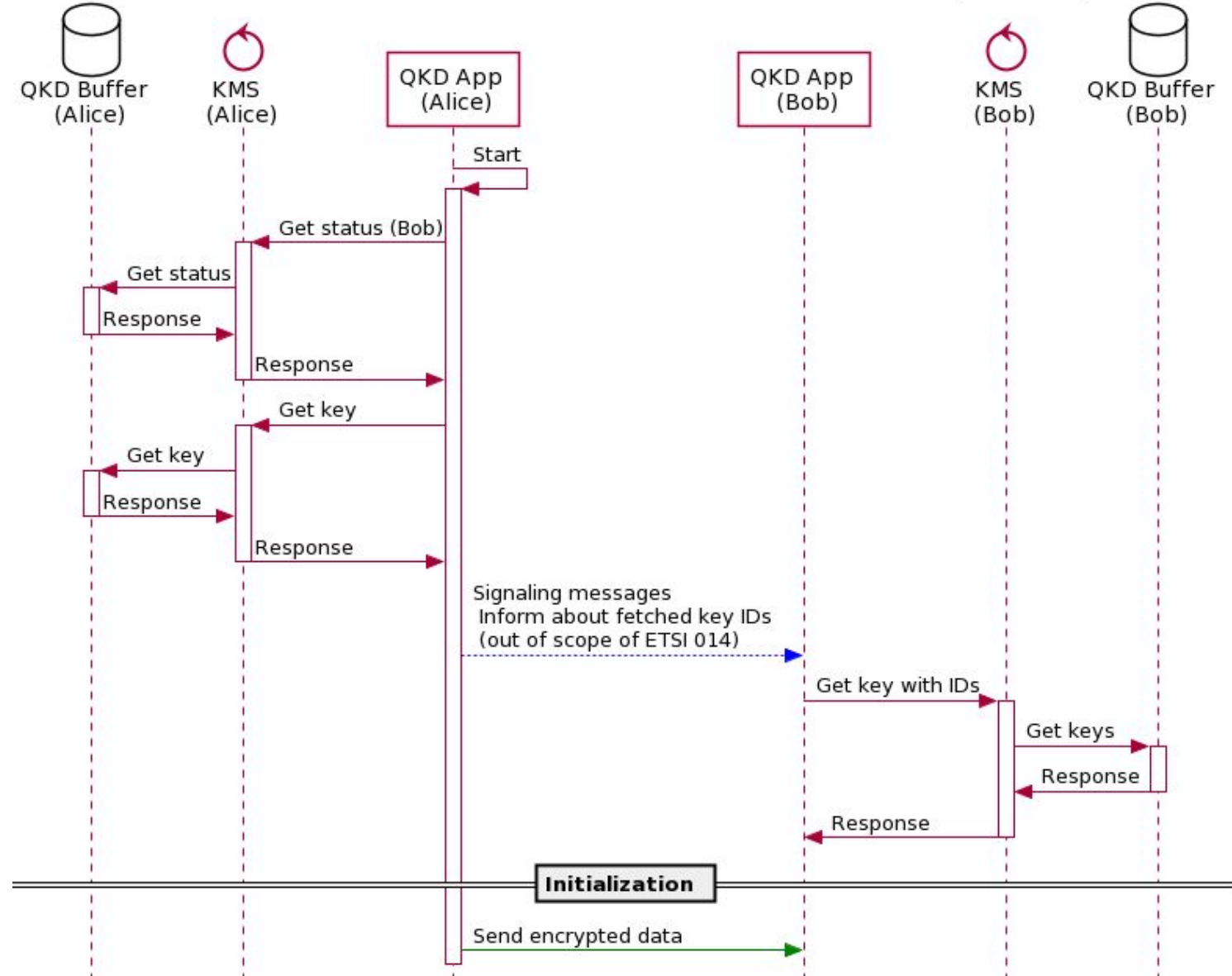


- Specification of QKD link parameters:
 - average key rate and key size
 - post-processing application parameters (packet sizes and average amount of traffic)
- Specification of QKD application:
 - type of encryption/authentication
 - amount of traffic and packet sizes
 - number of keys to fetch from KMS

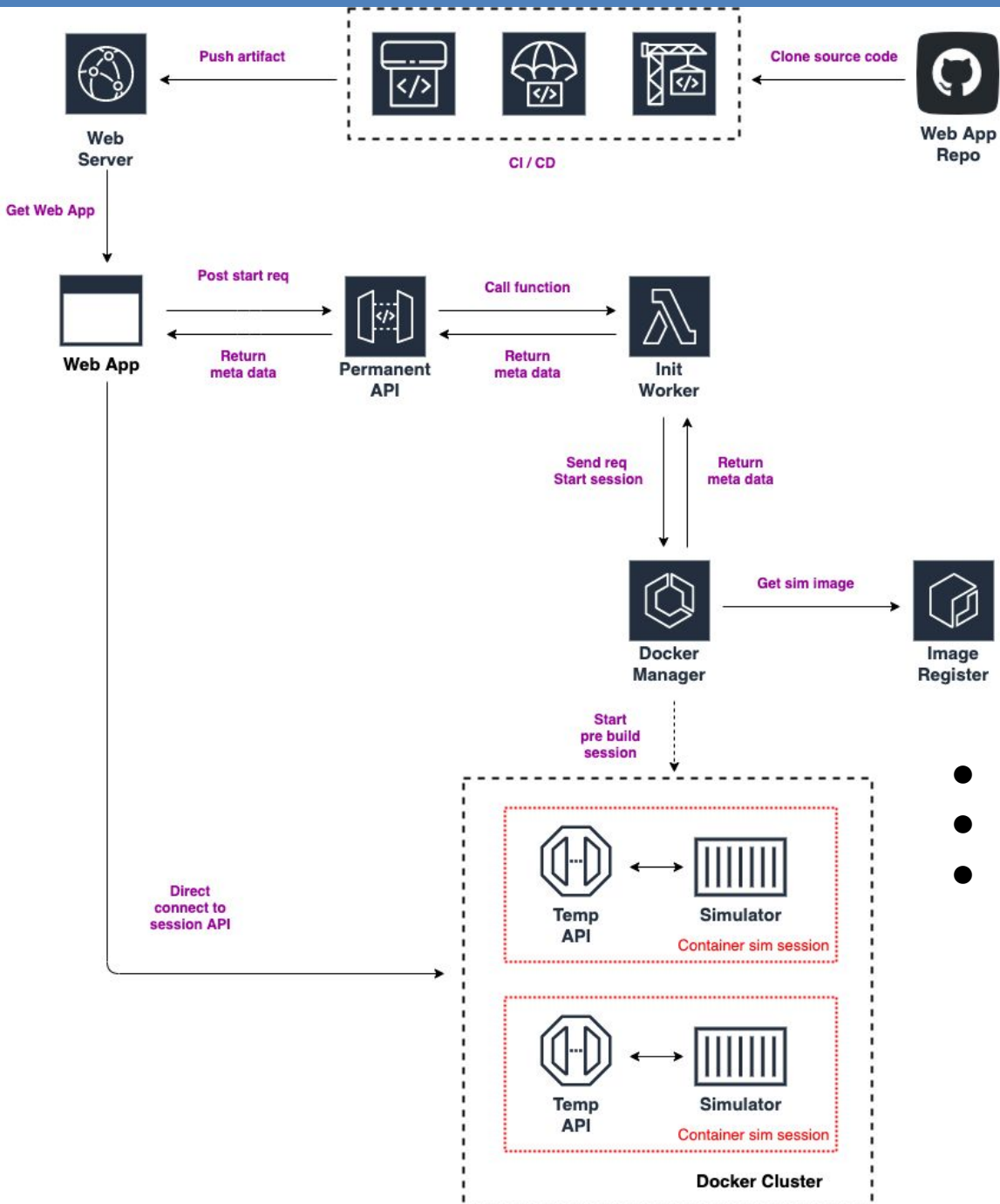
ETSI ISG QKD 014

- QKDNetSim implements ETSI 014 specification [4]
 - Quantum Key Distribution (QKD); Protocol and data format of REST-based key delivery API
 - It specifies the communication between KMSs and QKD applications
 - Communication between KMSs and between applications is out-of-scope

ETSI QKD 014 - Protocol and data format of REST-based key delivery API



[4] ETSI ISG QKD 014, "Quantum Key Distribution (QKD); Protocol and data format of REST-based key delivery API," vol. 1, pp. 1–22, 2019, [Online]. Available: https://www.etsi.org/deliver/etsi_gs/QKD/001_099/014/01.01.01_60/gs_qkd014v010101p.pdf



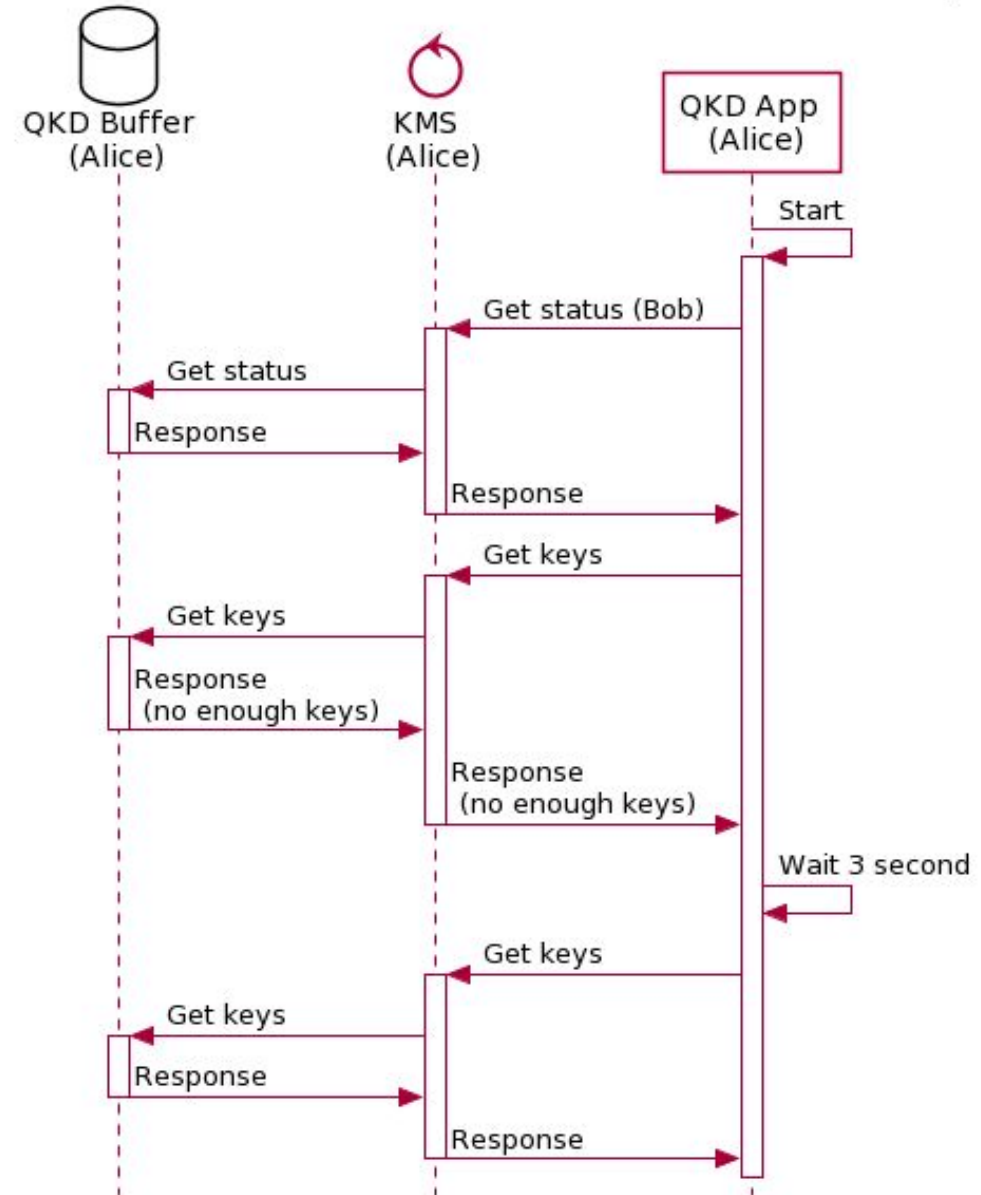
- QKDNetSim is deployed in docker with web Interface.
- It is available at www.open-qkd.eu

- QKDNetSim is a console-oriented simulator
- A web interface has been implemented.
- The user can select points on the map, between which the distance is automatically calculated, and adjust simulation parameters such as
 - the key generation rate,
 - type of cryptographic techniques, and
 - ETSI 014 parameters.

QKDNetSim example

ETSI QKD 014 - Protocol and data format of REST-based key delivery API

- Application behaviour:
 - If no keys are available, application will wait for 3 seconds before sending request to KMS again
 - It is an implementation detail introduced to protect KMS from malicious applications



QKDNetSim Example #1 - Configuration

- Let's define example configuration values as shown in table.
- Note that QKD storages are empty on start. Therefore, it is advised to generate keys prior usage (set start time of QKD systems to be lower than the APP start time)

	Parameter	Value
Application configuration	Start time [s]	10
	Stop time [s]	50
	Number of keys to fetch per GET_KEY request	10
	Data rate [kbps]	10
	Packet size [bytes]	300
QKD configuration	Start time [s]	0
	Stop time [s]	50
	Key size [bytes]	1000
	Key rate [kbps]	100
	Post-processing rate [bps]	1000
	Post-processing packet size [bytes]	100

QKDNetSim Example #1 - Results

- Signaling data are data exchanged between Alice and Bob's application carrying keyIDs fetched from KMSs. They are implemented using REST API and their specification is out of scope of ETSI 014.
 - number of signaling data and number of packets exchanged with KMS increases as higher refresh rate is used (OTP, AES*refresh5)
- Column “# keys consumed” registers the number of keys consumed from the KMS perspective.
- The number of keys consumed when encryption and VMAC authentication are applied compared to the case where only encryption is slightly reduced.

Cryptographic method	# data packets exchanged	# signaling data packets exchanged	# packets to KMS exchanged	# keys consumed	# keys generated
Unprotected	167	0	0	0	158
OTP	70	14	23	70	158
OTP + VMAC	53	12	16	60	158
AES (*refresh 5)	164	8	9	40	158
AES (*refresh 5) + VMAC	166	8	9	40	158
AES (*refresh 10)	167	4	5	20	158
AES (*refresh 10) + VMAC	167	4	5	20	158

QKNetSim Example #2 - Results

- The results show that the achieved application data rate noticeably decreases in the case of OTP cipher being employed.
- When application consumes all keys, the traffic exchange is stopped until a new set of keys is obtained.
- Large number of keys means larger packets to exchange and it takes more time to complete key transfer from KMS to application. It is best demonstrated when OTP and AES (1 – refresh rate) is employed for the desired data rate of 100 kbps.
- If the application's request cannot be satisfied (the requested number of keys is not available), the application is stopped. The results obtained in the case of AES (1) and AES (10) represent this scenario.

User defined application data rate [kbps]	Encryption method	# data packets exchanged	# application punishments	# keys consumed
10	Unprotected	167	0	0
	OTP	70	8	70
	AES (10)	167	0	20
20	Unprotected	334	0	0
	OTP	63	10	70
	AES (10)	328	0	40
50	Unprotected	834	0	0
	OTP	60	12	70
	AES (10)	682	2	70
80	Unprotected	1334	0	0
	OTP	60	12	70
	AES (10)	678	5	70
100	Unprotected	1667	0	0
	OTP	64	13	70
	AES (1)	70	13	70
	AES (10)	700	8	70

Conclusion

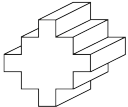
- QKDNetSim web interface is suitable to test the load of QKD links and evaluate communication with KMS entities.
- The current version of web interface supports ETSI 014; Feel free to test it at www.open-qkd.eu
- The next version of the QKDNetSim will include route specifications
 - work on ETSI 004 specification is in final stage
 - support for Software Defined Networking (SDN) is in heavy development phase



VSB TECHNICAL
UNIVERSITY
OF OSTRAVA

THANK YOU!

Please send your questions and queries to miralem.mehic@ieee.org

OPEN  QKD



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 857156.