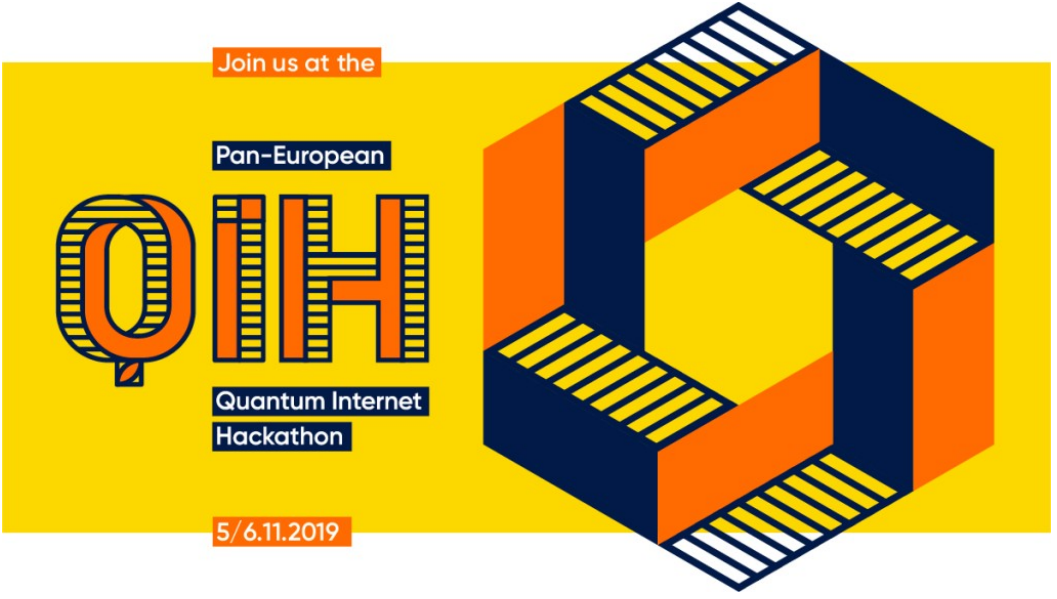
Quantum Key Distribution in OpenSSL

QIRG Singapore 19 November 2019

Introduction

- On November 5-6 RIPE Labs organised a *Pan-European Quantum Internet Hackathon*



Introduction

- Participants from multiple “nodes” (Delft, Paris, Sarajevo, Padova, Geneva, Dublin) tackled various quantum internet challenges
- One of them was to integrate a QKD protocol into the OpenSSL library

The OpenSSL challenge

- Challenge descripton: https://github.com/PEQI19/PEQI-OpenSSL
- The end-goal of the challenge is to use an off-the-shelf browser (e.g. Firefox) and connect it to a secure HTTPS website hosted on an off-the-shelf web server (e.g. Apache), while using a QKD algorithm as the key agreement protocol.

The OpenSSL challenge

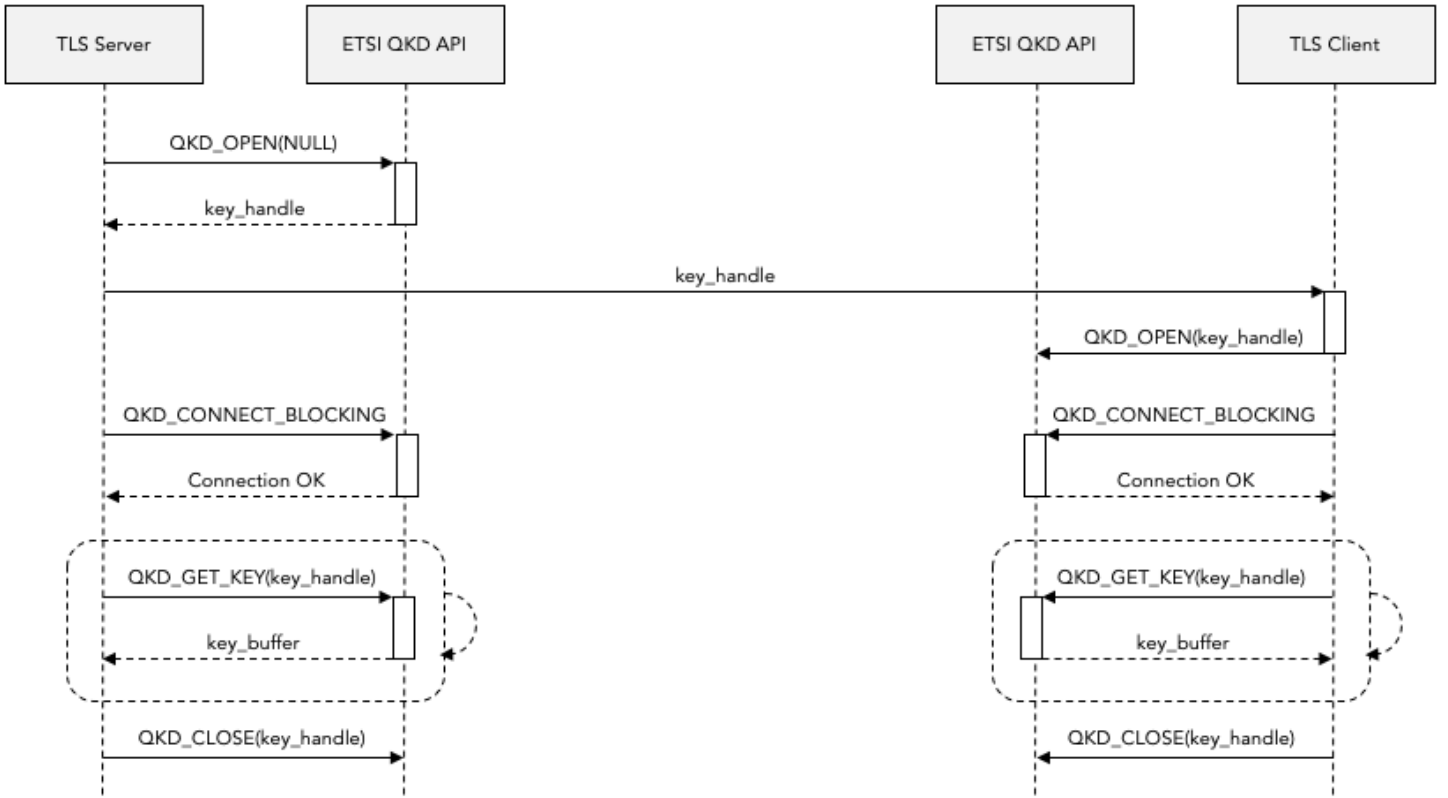
- Motivation for the challenge:
  - Today quantum networking is either at the level of hardware and low-level operations or abstract protocols with complex mathematical proofs
  - Bring quantum networks to a user-level application
  - Open up the field to software engineers without a quantum physics background
  - Relate the hackathon to ongoing QKD work which is already commercialised

ETSI QKD API

- Challenge was built around the ETSI QKD API: https://www.etsi.org/deliver/etsi_gs/QKD/001_099/004/01.01.01_60/gs_QKD004v010101p.pdf
- The API is very simple and defines 5 functions:

```
Interface QKD_AppInt{
   QKD_OPEN (in destination, in QoS, inout key_handle , out status);

   QKD_CONNECT_NONBLOCK (in key_handle, out status);

   QKD_CONNECT_BLOCKING (in key_handle, in timeout, out status);

   QKD_GET_KEY (in key_handle, out key_buffer, out status);

   QKD_CLOSE (in key_handle, out status);
   }
```

ETSI QKD API


TLS Server
ETSI QKD API
ETSI QKD API
TLS Client
QKD_OPEN(NULL)
key_handle
key_handle
QKD_OPEN(key_handle)
QKD_CONNECT_BLOCKING
QKD_CONNECT_BLOCKING
Connection OK
Connection OK
QKD_GET_KEY(key_handle)
QKD_GET_KEY(key_handle)
key_buffer
key_buffer
QKD_CLOSE(key_handle)
QKD_CLOSE(key_handle)

Elements of the challenge

- The API splits the challenge into two parts:
  - Integrate the QKD API with OpenSSL
  - Implement the QKD API on top of a quantum network (simulated for now)
  - Simulated network uses same low-level API as the Dutch demonstration network (2021)

Integrating with OpenSSL

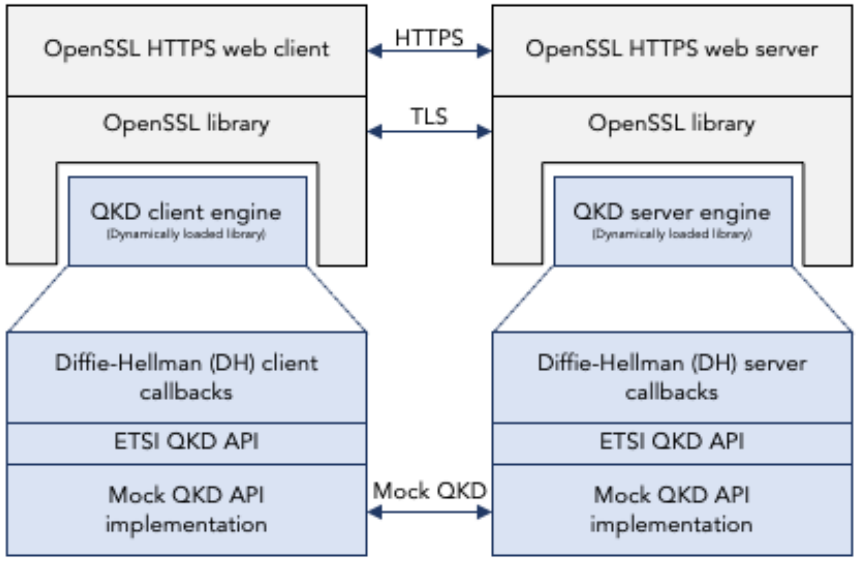
- A team in Delft attempted to integrate the QKD API into OpenSSL
  - Bruno Rijsman, Yvo Keuter, Tim Janssen
- A very thorough write-up with plenty of introductory material and running code: https://brunorijsman.github.io/openssl-qkd/

Integrating with OpenSSL

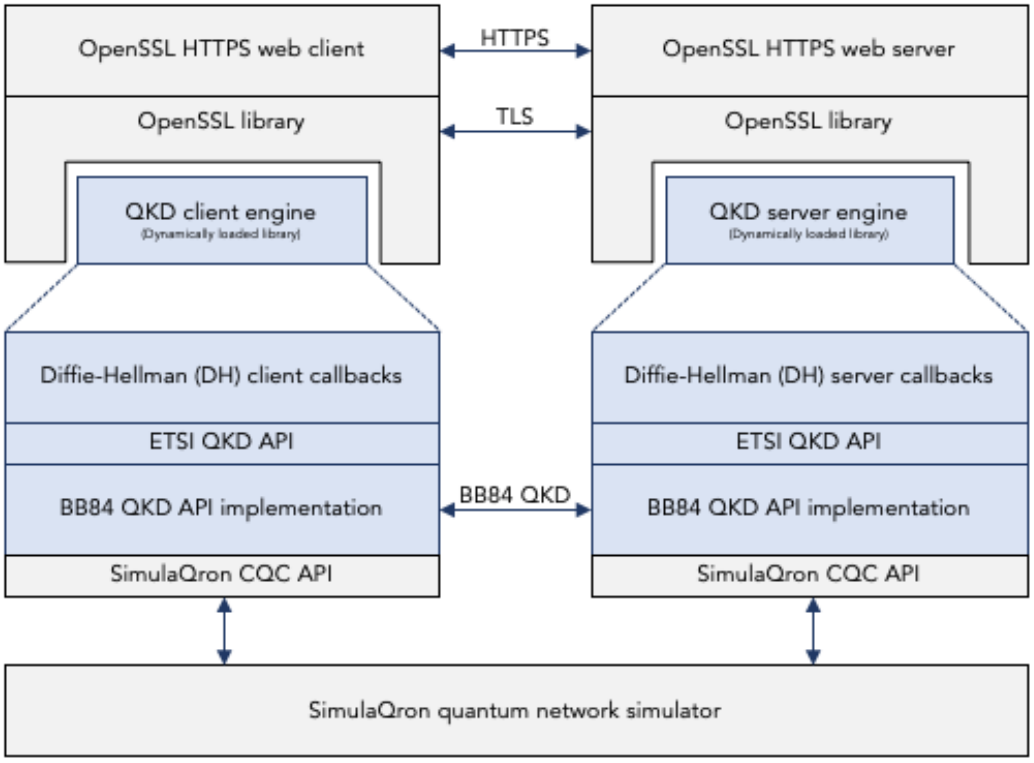
- There are two ways of extending OpenSSL
  - Hacking the existing engine-based extension mechanism to extend existing protocols
  - Introducing a new first-class key exchange protocol state machine
- Team opted to implement engine by “abusing” the Diffie-Hellman protocol

Integrating with OpenSSL

Mock QKD implementation



BB84 QKD implementation running on SimulaQron



Integrating with OpenSSL

- Difficulties and challenges:
  - The engine behaves differently on the server and client side, but this is not reflected in the engine callbacks from OpenSSL
  - The provided Mock API coupled with a hacked DH engine led to deadlocks

Conclusions

- OpenSSL is challenging to extend
- The “easy” hacky way of abusing the DH protocol engine had its own challenges
- Write up of the project (including running code): https://brunorijsman.github.io/openssl-qkd/