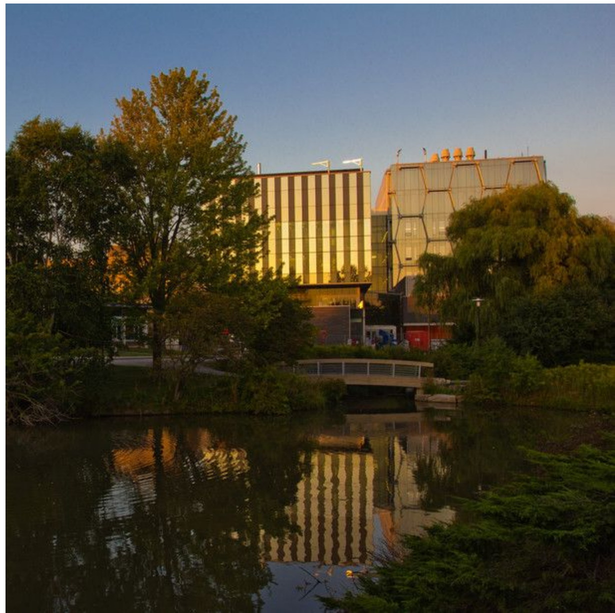
ETSI/IQC Quantum Safe Cryptography Workshop 2020

Open QKD Network

- An Open Source QKD Network Project

Kaiduan Xie

UNIVERSITY OF WATERLOO | IQC Institute for Quantum Computing

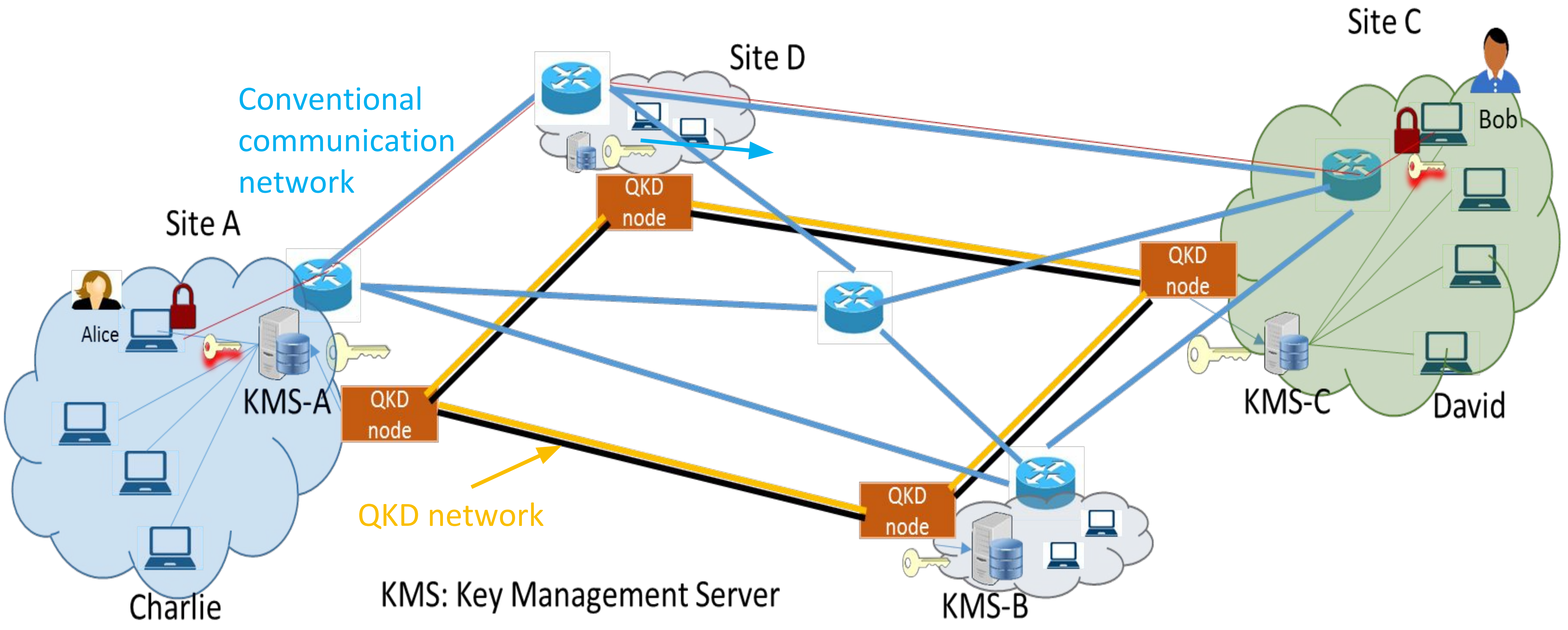


Contents

Project Goals

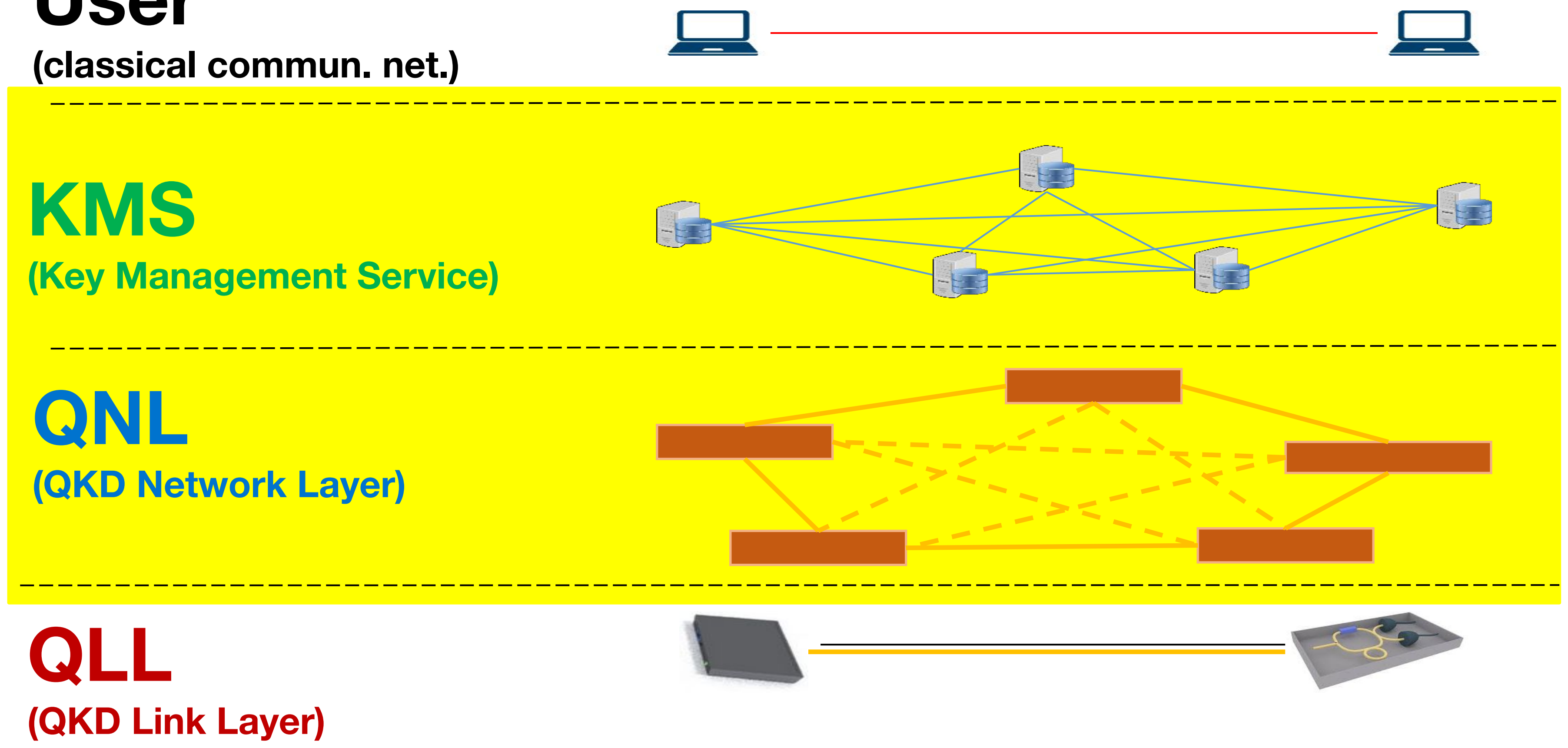
- Show the feasibility of integrating QKD technology with classical communication networks
- Provide a reference implementation of the 4-layer architecture
- Make it easy for people to showcase QKD technologies at a network scale
- Make it easy for people to deploy pilot QKD networks

Integrate Communication Networks with QKD



In Scope

4-Layer Architecture: QKD technology-independent design



Requests and uses keys

Manages and Issues Keys to User applications

Extends QKD from Point-to-Point Links to a Network

Produces Raw Key Bits using QKD technologies

Out-of-Scope

User Layer

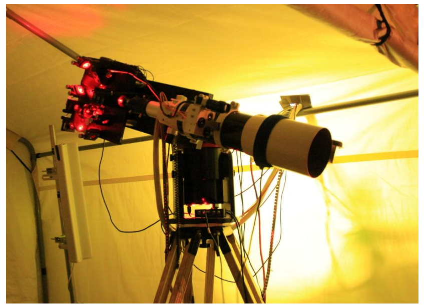
User utilizes QKD keys at its wish

- can be ***any*** of the layers 2-5 entities in classical networks, e.g.
- enterprise applications (L5)
- TLS/DTLS (L4)
- IPsec (L3)
- link encryptor (L2)

QLL (QKD Link Layer)

All about QKD Technologies

- A plethora of protocols and platforms to choose from
- Optical fiber based QKD
- Satellite QKD
- …



Note: It's the enterprise's responsibility to develop User Layer, thus beyond the scope of the project.

Main Functionalities Implemented

- KMS
  - Interface with user applications
    - ETSI GS QKD 014 V1.1.1 REST API (new feature)
    - University of Waterloo API
  - Basic key management capabilities
  - Interface with the QNL
- QNL
  - Establish shared keys across a network via trusted relay nodes
  - A routing module that can react to QKD network topology dynamics and hide it from KMS
  - Key relaying hop by hop via OTP using QLL generated keys
  - All QNL requests/responses are HMAC protected (new feature)

U of Waterloo API vs ETSI REST API

	U of Waterloo API	ETSI API
client A	api/getkey?siteid=B&index=&blockid=	api/v1/keys/B/dec_keys?key_ID=index-blockid
server B	api/newkey?siteid=A returns index/blockid/hexkey in JSON	api/v1/keys/A/enc_keys returns index/blockid/hexkey in JSON

Demo Applications Provided

- To showcase how (easy it is) to get and use QKD-generated keys
  - Not the main effort of this project

- Demo 1: TLS with PSK
  - A TLS tunnel is established between a client and a server, with QKD keys as the pre-shared key (PSK)
  - A file is transferred over the TLS tunnel

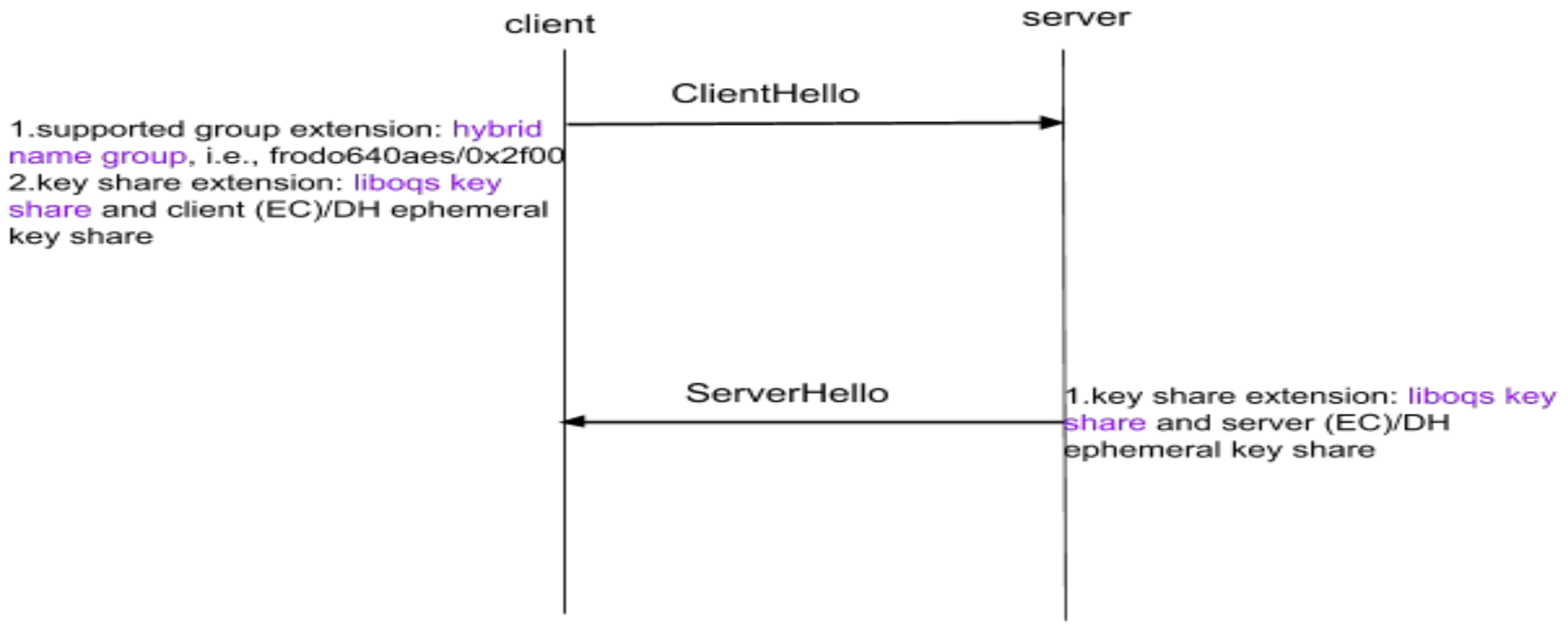
- Demo 2: video chat
  - using qTox* between two users
  - using QKD Keys to protect the confidentiality of the session

* qTox is an open source peer-to-peer communication tool

OpenQKD Network/OpenQuantumSafe/openSSL integration

- TLS 1.3 liboqs + (EC)DH hybrid key exchange
- TLS 1.3 OpenQKD Network + liboqs + (EC)DH triple key exchange
- New OpenSSL APIs and libopenqkd
- Overall process

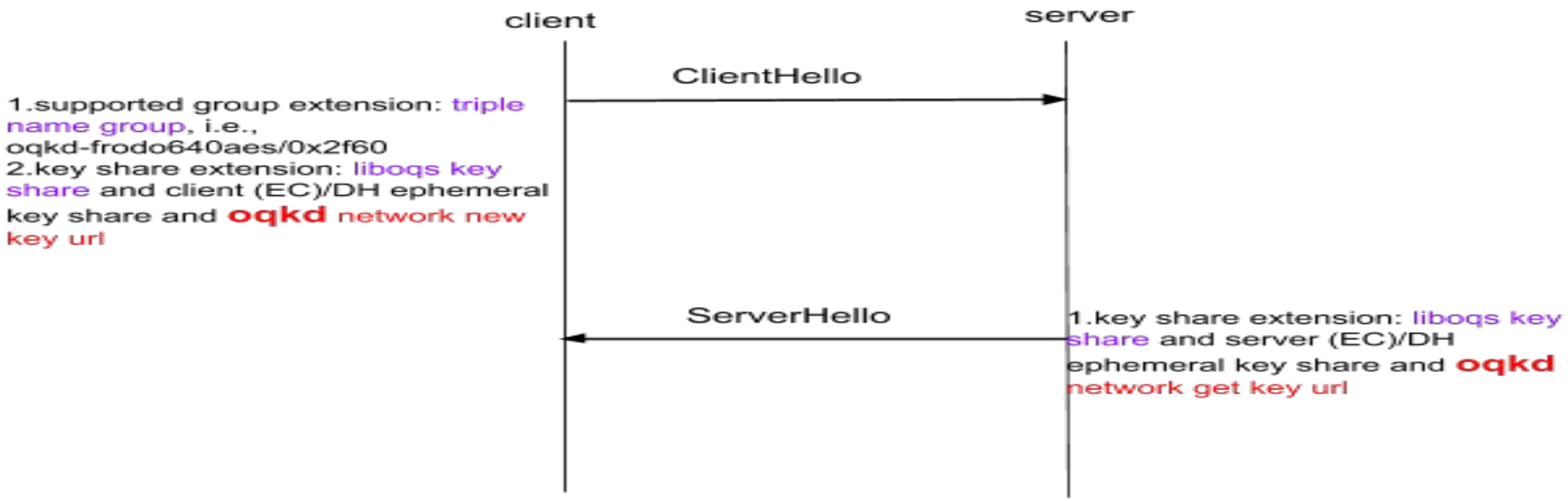
TLS 1.3 liboqs + (EC) DH hybrid key exchange


client
server
ClientHello
1.supported group extension: hybrid name group, i.e., frodo640aes/0x2f00
2.key share extension: liboqs key share and client (EC)/DH ephemeral key share
ServerHello
1.key share extension: liboqs key share and server (EC)/DH ephemeral key share

concatenated_shared_secret = (EC)DH key || liboqs key *

*https://tools.ietf.org/html/draft-ietf-tls-hybrid-design-00

TLS 1.3 OpenQKD Network/liboqs/DH triple key exchange

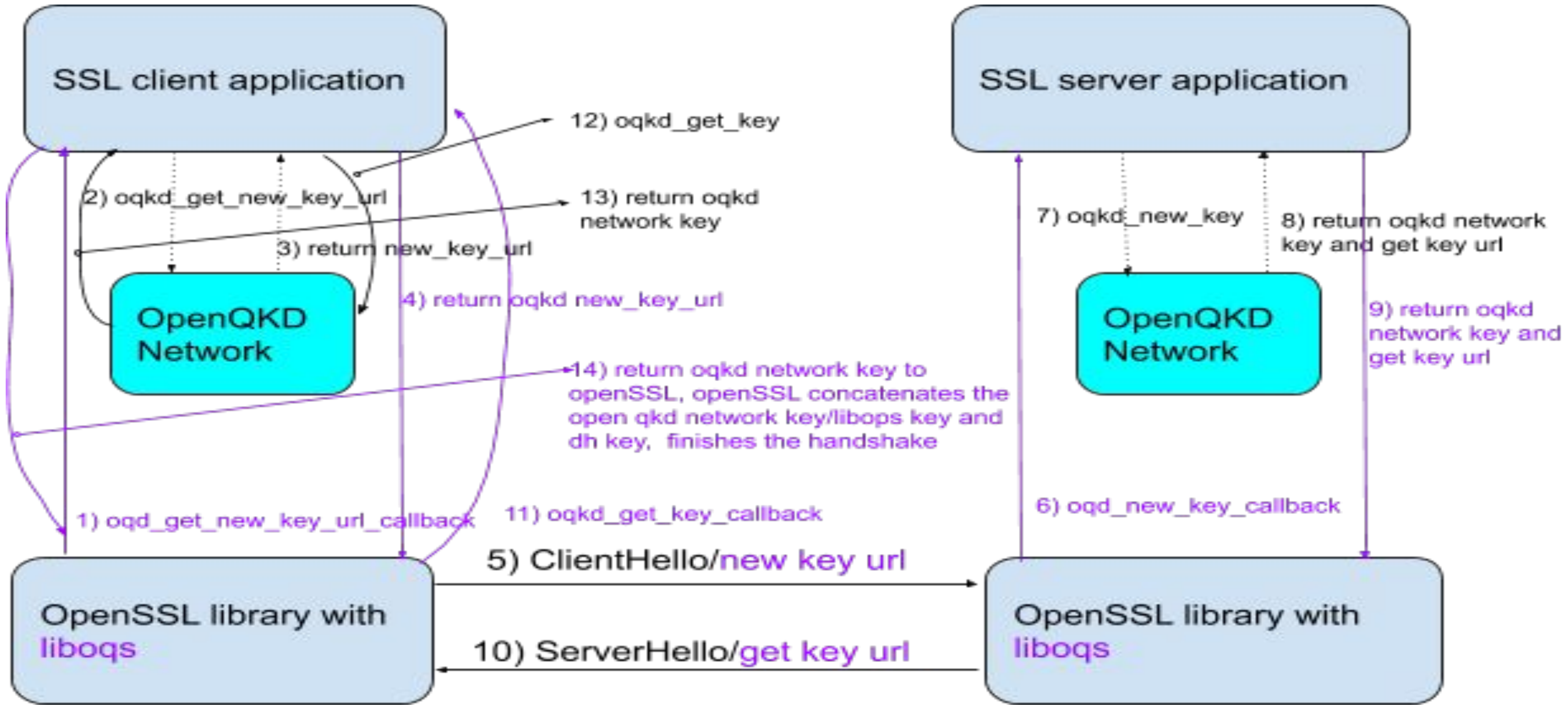

client
server
ClientHello
1.supported group extension: triple name group, i.e., oqkd-frodo640aes/0x2f60
2.key share extension: liboqs key share and client (EC)/DH ephemeral key share and oqkd network new key url
ServerHello
1.key share extension: liboqs key share and server (EC)/DH ephemeral key share and oqkd network get key url

- OpenQKD Network key is **NOT** sent in the SSL ClientHello/ServerHello message
- OpenQKD Network new_key_url/get_key_url is sent in ClientHello/ServerHello respectively
- Client and server news key and gets key from new_key_url/get_key_url via OpenQKD Network REST API
- *concatenated_shared_secret* = (EC)DH key || liboqs key || **OpenQKD Network key**

New OpenSSL APIs & libopenqkd

- New OpenSSL APIs
  - Client side
    - *SSL_set_oqkd_new_key_url_callback*
    - *SSL_set_oqkd_get_key_callback*
  - Server side
    - *SSL_set_oqkd_new_key_callback*
- OpenQKD Network library/libopenqkd
  - Client side
    - *oqkd_get_new_key_url*
    - *oqkd_get_key*
  - Server side
    - *oqkd_new_key*
- Sample application
  - Updated OpenSSL built-in s_client/s_server

Overall process


SSL client application
SSL server application
12) oqkd_get_key
2) oqkd_get_new_key_url
13) return oqkd network key
3) return new_key_url
OpenQKD Network
4) return oqkd new_key_url
14) return oqkd network key to openSSL, openSSL concatenates the open qkd network key/libops key and dh key, finishes the handshake
1) oqd_get_new_key_url_callback
11) oqkd_get_key_callback
7) oqkd_new_key
8) return oqkd network key and get key url
OpenQKD Network
9) return oqkd network key and get key url
6) oqd_new_key_callback
5) ClientHello/new key url
10) ServerHello/get key url
OpenSSL library with liboqs
OpenSSL library with liboqs

Contact

Dr. Michele Mosca
University Research Chair
519-888-4567 x37484
michele.mosca@uwaterloo.ca

Dr. Norbert Lütkenhaus
Professor
519-888-4567 x32870
lutkenhaus.office@uwaterloo.ca

Webpage: https://openqkdnetwork.ca/

GitHub page: https://github.com/Open-QKD-Network

UNIVERSITY OF WATERLOO | IQC Institute for Quantum Computing