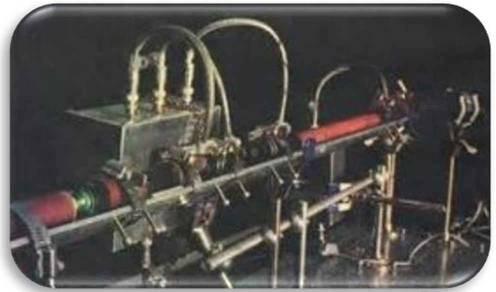
Experimenting with Post-Quantum Cryptography in TLS/SSH with the Open Quantum Safe project

INTERNATIONAL CRYPTOGRAPHIC MODULE CONFERENCE 2020
September 21-24 | Virtual Conference and Vendor Event

Christian Paquin
@chpaquin
Principal Program Manager
Microsoft Research

About

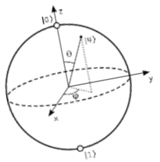
- Studied quantum cryptography 25 years ago at University of Montreal
- Worked in the industry as a cryptographic engineer
- Joined Microsoft more than a decade ago
  - Now with *MSR Security & Crypto* team, working with cutting-edge crypto tech



Université de Montréal

https://www.microsoft.com/en-us/research/group/security-and-cryptography/

The Quantum Revolution



$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

- Quantum computers will revolutionize information processing
  - My colleagues are building the full stack, from the chip to the SDK https://www.microsoft.com/quantum/
- Quantum computers are bad news for cryptography!
  - Shor breaks RSA, DSA, DH and the ECC variants
- Could be built within 10-15 years
- Breaks ~~most~~ all the asymmetric crypto in use today

https:// SSH

- We need new *quantum-safe* cryptography

Need to migrate to quantum-safe crypto soon

- Capture now, decrypt later
- Updating standards takes time
  - TLS, SSH, IKE, PKI, S/MIME, …
- Unknown impact on code base
  - Longer key/message/sig sizes
  - Slower running times
  - Code agility

Do your apps protect data that needs to be kept secret for more than 10 years?

NIST competition

Nov 2017	Apr 2018	Jan 2019	Aug 2019	2020/2021	2022-2024
• Round 1 • 69 submitted	• 1st NIST PQC workshop	• Round 2 • 26 remaining	• 2nd NIST PQC workshop	• Round 3 • 15 remaining	• Draft standards

Encryption / Key Encapsulation

Finalists

- Classic McEliece
- CRYSTALS-KYBER
- NTRU
- SABER

Alternates

- BIKE
- FrodoKEM
- HQC
- NTRU Prime
- SIKE

Signature

Finalists

- CRYSTALS-DILITHIUM
- FALCON
- Rainbow

Alternates

- GeMSS
- Picnic
- SPHINCS+

MSR's collaborations

- FrodoKEM (KEM)
  - Learning With Errors (LWE)
  - https://frodokem.org/
- SIKE (KEM)
  - Supersingular Isogeny elliptic curves
  - https://sike.org/
- Picnic (sig)
  - Zero-knowledge proofs, hash, and block ciphers
  - https://microsoft.github.io/Picnic/

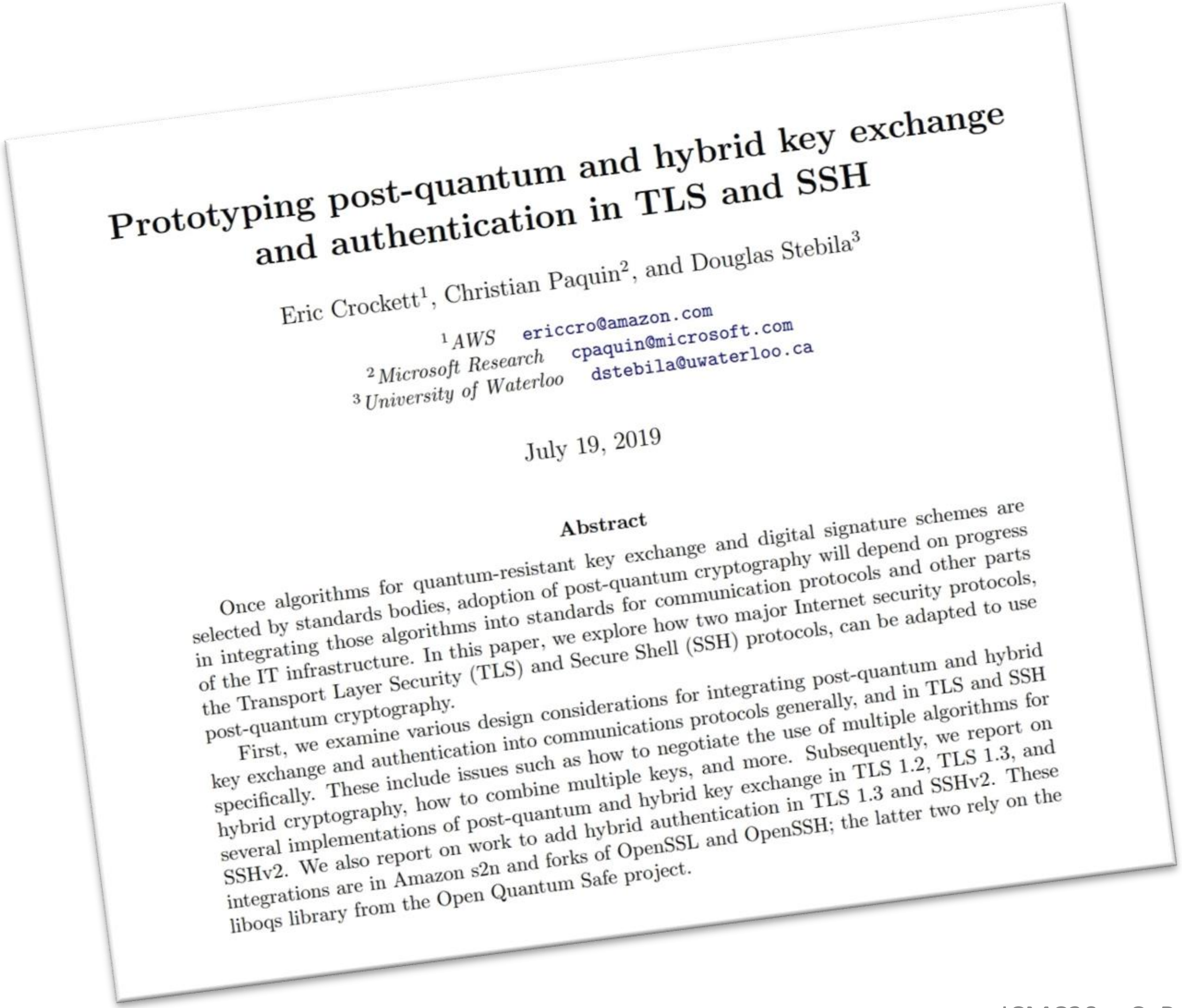
OPEN QUANTUM SAFE

- C library created to simplify integration of PQC into applications
- Contributions from

University of Waterloo, Microsoft Research, amazon web services, cisco, IBM, Royal Holloway University of London, evolution, SRI International, Radboud University, Institut Tutte Institute

- Supports many NIST round 2 KEM and signature schemes
  - Round 3 updates in next release
- Integrations into boringssl, OpenSSL (TLS, CMS), OpenSSH, OpenVPN
- C++, C#, Go, Java, and Python wrappers
- Test server, docker images for curl, Apache, nginx, Chromium
- https://openquantumsafe.org/

Prototyping PQC



Prototyping post-quantum and hybrid key exchange and authentication in TLS and SSH

Eric Crockett[1], Christian Paquin[2], and Douglas Stebila[3]

[1] *AWS* ericcro@amazon.com
[2] *Microsoft Research* cpaquin@microsoft.com
[3] *University of Waterloo* dstebila@uwaterloo.ca

July 19, 2019

Abstract

Once algorithms for quantum-resistant key exchange and digital signature schemes are selected by standards bodies, adoption of post-quantum cryptography will depend on progress in integrating those algorithms into standards for communication protocols and other parts of the IT infrastructure. In this paper, we explore how two major Internet security protocols, the Transport Layer Security (TLS) and Secure Shell (SSH) protocols, can be adapted to use post-quantum cryptography.

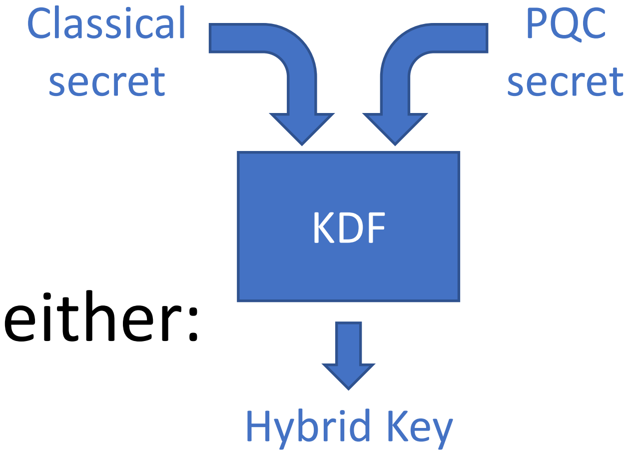
First, we examine various design considerations for integrating post-quantum and hybrid key exchange and authentication into communications protocols generally, and in TLS and SSH specifically. These include issues such as how to negotiate the use of multiple algorithms for hybrid cryptography, how to combine multiple keys, and more. Subsequently, we report on several implementations of post-quantum and hybrid key exchange in TLS 1.2, TLS 1.3, and SSHv2. We also report on work to add hybrid authentication in TLS 1.3 and SSHv2. These integrations are in Amazon s2n and forks of OpenSSL and OpenSSH; the latter two rely on the liboqs library from the Open Quantum Safe project.

- Analyze various options to integrate PQC into TLS and SSH
- Focus on hybrid scenarios
- Lessons learned from OpenSSL, OpenSSH, and s2n integrations

https://eprint.iacr.org/2019/858

Hybrid scenarios

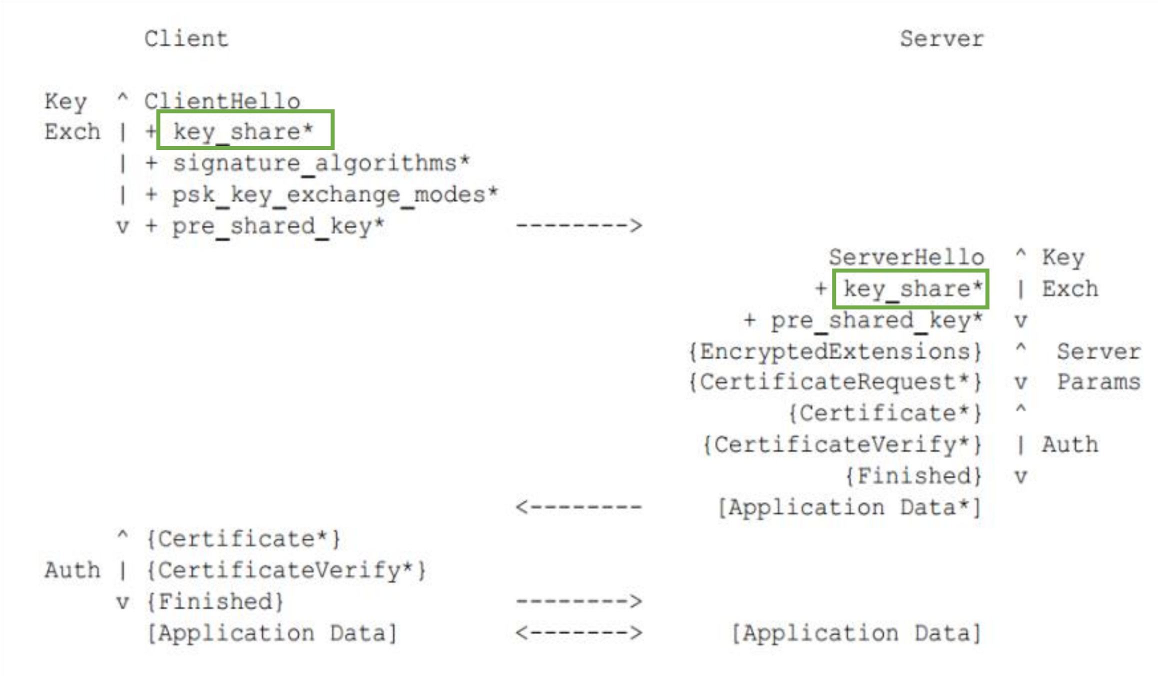
- Early migration should use a hybrid of classical/PQ schemes
  - Security of today + safety net against quantum computer
  - Secure if one of the two is secure



- TLS and SSH negotiate one algorithm; need to define either:
  - new combo schemes
  - a new hybrid approach

- Consider backward compatibility, performance, latency, data flow

TLS case study



- Added PQ/hybrid KEX & auth
- TLS 1.2 (OpenSSL 1.0.2)
- TLS 1.3 (OpenSSL 1.1.1)
  - PQ algs masquerade as EC curves
  - Concatenation strategy more secure than 1.2 (KDF hashes transcripts)
  - Spec pub key and sig limit: 2^{16}-1 bytes, cert limit: 2^{24}-1 bytes
    - OpenSSL limit is smaller
  - Tested with OpenSSL/boringssl tools, apache, nginx

- https://github.com/open-quantum-safe/openssl

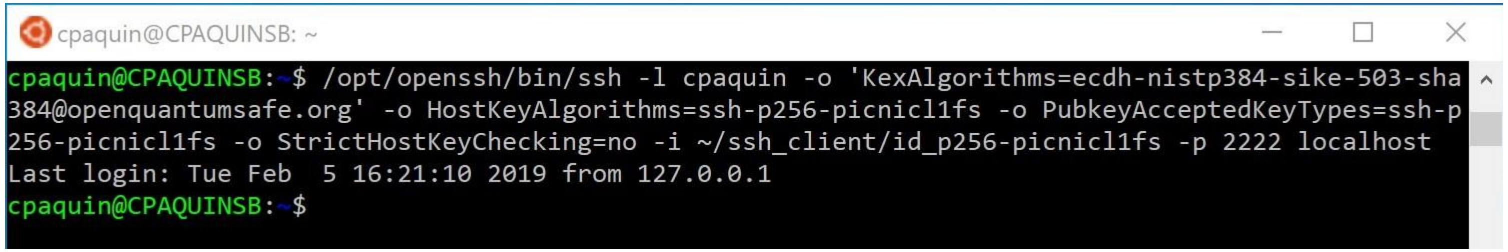
Demo

TLS 1.3 – OpenSSL 1.1.1

- KEX: ECDHE P256 + Frodo 640 AES
- Auth: ECDSA P256 + Picnic3-L1

SSH case study

- Added PQ/hybrid KEX & auth to OpenSSH
- Define new algorithms
  - e.g.: ecdh-nistp384-sike-503-sha384@openquantumsafe.org
- Supports both client and server public key authentication
- Spec message size limit: 2^{32} bytes
  - large enough for all round 2 candidates, but OpenSSH limit is smaller (2^{18})
- https://github.com/open-quantum-safe/openssh-portable



Key Encapsulation Mechanisms

KEM scheme	OpenSSL 1.0.2 TLS 1.2	OpenSSL 1.1.1 TLS 1.3	OpenSSH 7.9 SSH2
BIKE 1/2/3 L1/3/5 (round 1)	✓	✓	✓
Frodo KEM 640/976 AES/SHAKE	✓	✓	✓
Frodo KEM 1344 AES/SHAKE	☑	☑	✓
Kyber 512/768/1024	✓	✓	✓
LEDAcrypt KEM LT 12/32/52	✓	✓	✓
NewHope 512/1024 CCA	✓	✓	✓
NTRU HPS (2048-509/677)/(4096-821)	✓	✓	✓
NTRU HRSS 701	✓	✓	✓
NTS KEM (12,64)	✗	✗	✗
LightSaber/Saber/FireSaber KEM	✓	✓	✓
SIKE p434/p503/p610/p751	✓	✓	✓

KEM integrations for both PQ and hybrid (with ECDHE)

<u>Legend</u>:

✓ Success

☑ Works with code mods

✗ Did not work

Signatures

KEM scheme	OpenSSL 1.1.1 TLS 1.3	OpenSSH 7.9 SSH2
Dilithium 2/3/4	✓	✓
MQDSS 31 48/64	☑	✓
Picnic L1 FS/UR	☑	✓
Picnic L3/L5 FS/UR	✗	✓
Picnic2 L1 FS	✓	✓
Picnic2 L3/L5 FS	☑	✓
qTesla I/III-size/III-speed (round 1)	✓	✓
Rainbow Ia Classic	☑	☑
Rainbow Ia Cyclic/Compressed	✓	✓
Rainbow IIIc/Vc Classic/Cyclic/Compressed	☑	✗
SPHINCS+ * 128s *	✓	✓
SPHINCS+ * 128f/192f/192s/256f/256s *	☑	✓

Signature integrations for both PQ and hybrid (with ECDSA)

Legend:

✓ Success

☑ Works with code mods

✗ Did not work

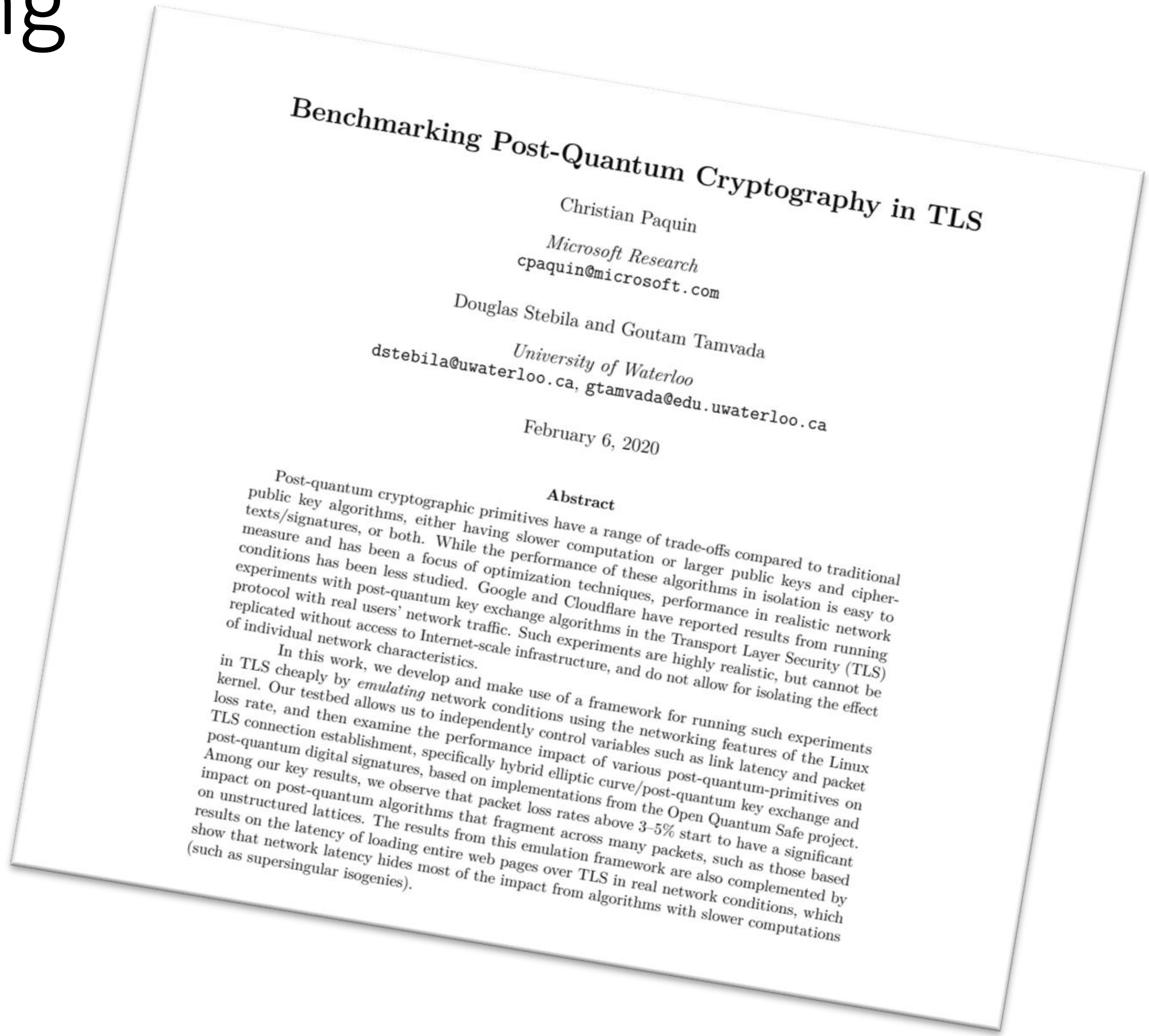
PQC TLS benchmarking

Two experiments to measure PQC impact on TLS performance

1) Simulated connections with various latency and packet-loss

2) Real-world retrievals of various page sizes from various geo-located VMs

https://eprint.iacr.org/2019/1447.pdf



Benchmarking Post-Quantum Cryptography in TLS

Christian Paquin
Microsoft Research
cpaquin@microsoft.com

Douglas Stebila and Goutam Tamvada
University of Waterloo
dstebila@uwaterloo.ca, gtamvada@edu.uwaterloo.ca

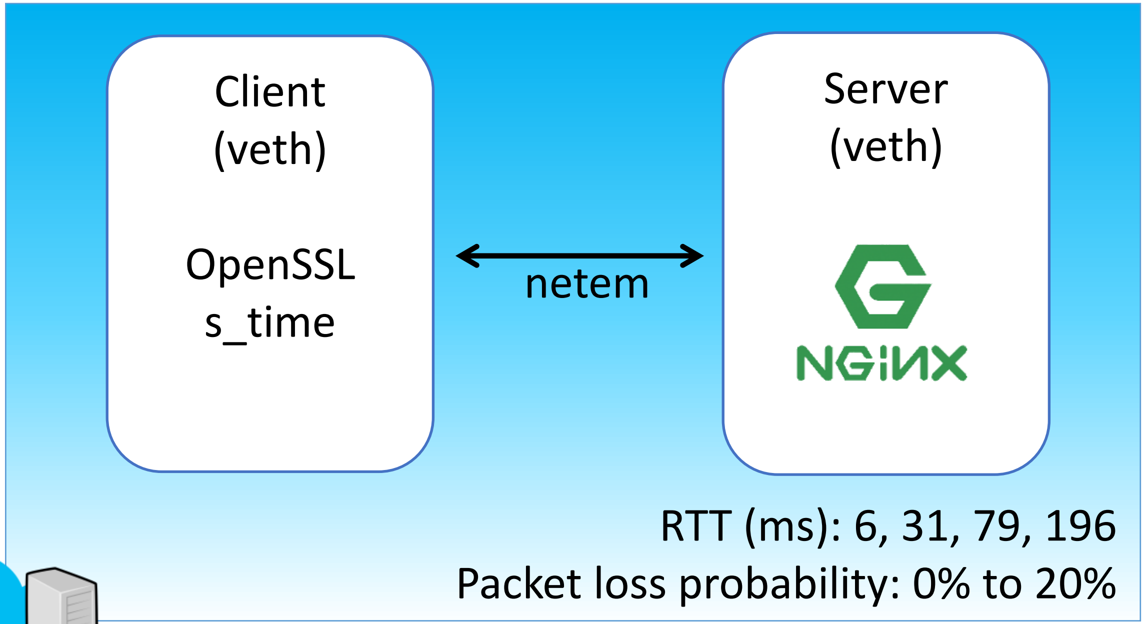
February 6, 2020

Abstract

Post-quantum cryptographic primitives have a range of trade-offs compared to traditional public key algorithms, either having slower computation or larger public keys and ciphertexts/signatures, or both. While the performance of these algorithms in isolation is easy to measure and has been a focus of optimization techniques, performance in realistic network conditions has been less studied. Google and Cloudflare have reported results from running experiments with post-quantum key exchange algorithms in the Transport Layer Security (TLS) protocol with real users' network traffic. Such experiments are highly realistic, but cannot be replicated without access to Internet-scale infrastructure, and do not allow for isolating the effect of individual network characteristics.

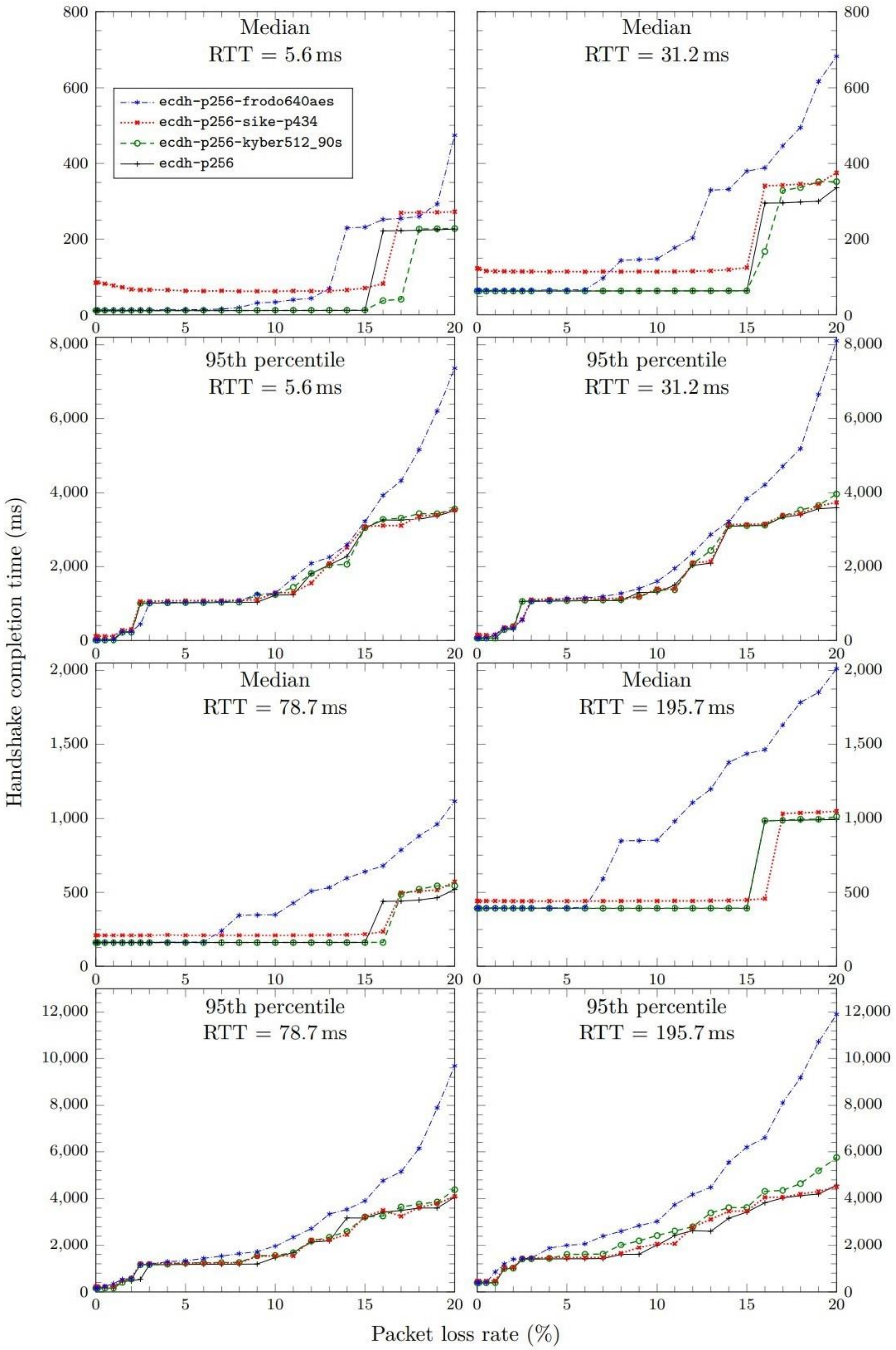
In this work, we develop and make use of a framework for running such experiments in TLS cheaply by *emulating* network conditions using the networking features of the Linux kernel. Our testbed allows us to independently control variables such as link latency and packet loss rate, and then examine the performance impact of various post-quantum-primitives on TLS connection establishment, specifically hybrid elliptic curve/post-quantum key exchange and post-quantum digital signatures, based on implementations from the Open Quantum Safe project. Among our key results, we observe that packet loss rates above 3–5% start to have a significant impact on post-quantum algorithms that fragment across many packets, such as those based on unstructured lattices. The results from this emulation framework are also complemented by results on the latency of loading entire web pages over TLS in real network conditions, which show that network latency hides most of the impact from algorithms with slower computations (such as supersingular isogenies).

Benchmarking results #1

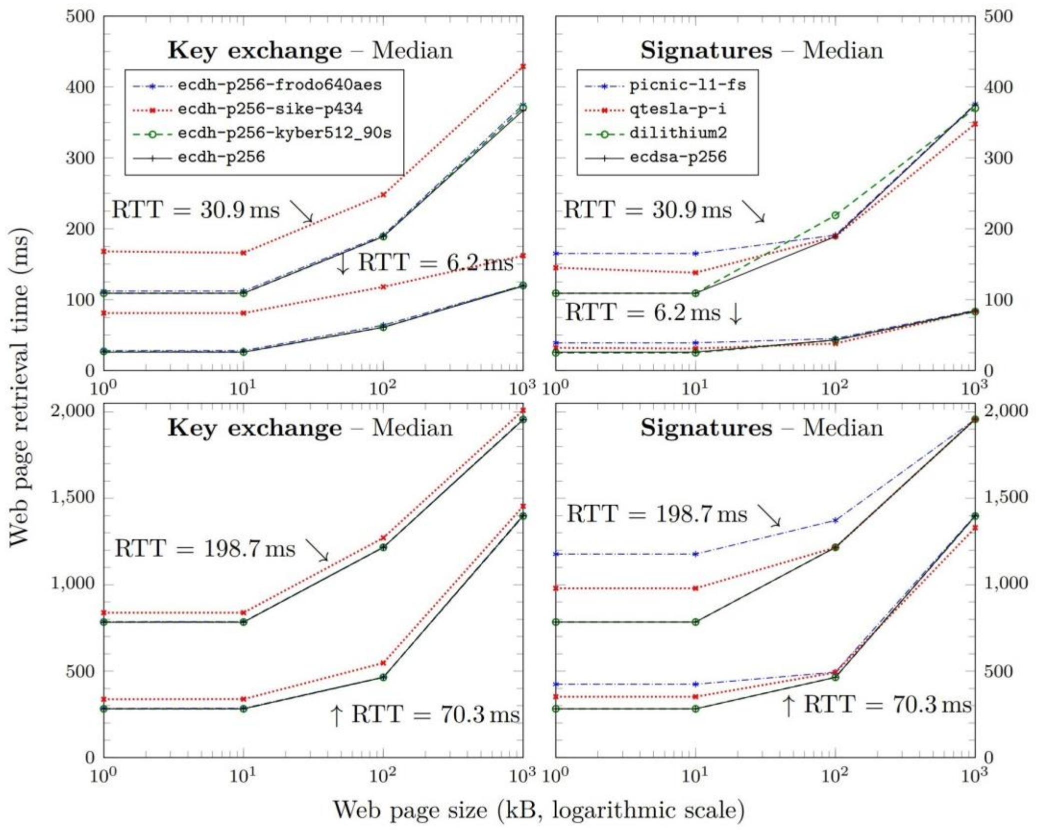


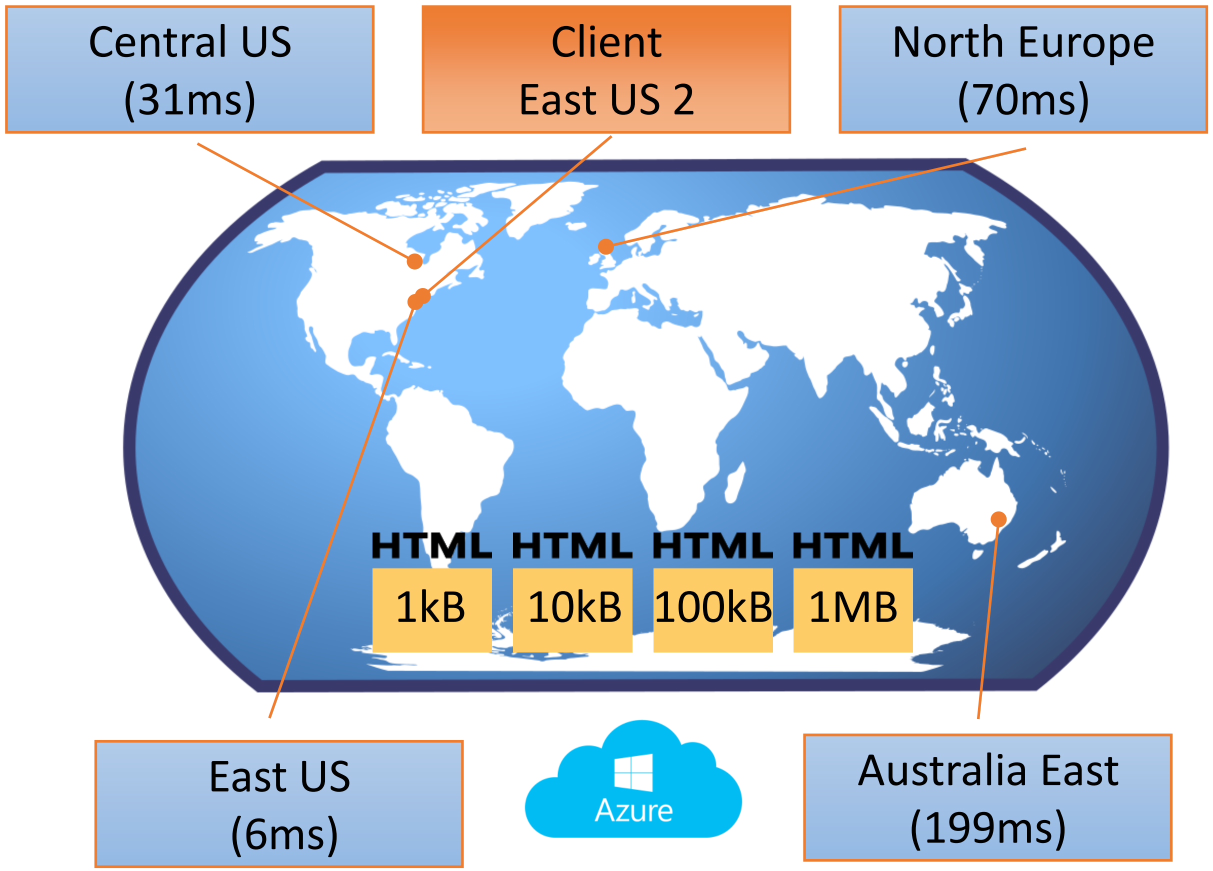
Linux (Ubuntu 18.04) - Azure D64s v3 VM – 64 vCPUs

" Packet loss rate > 3-5%: larger fragmented artefacts need to be retransmitted "



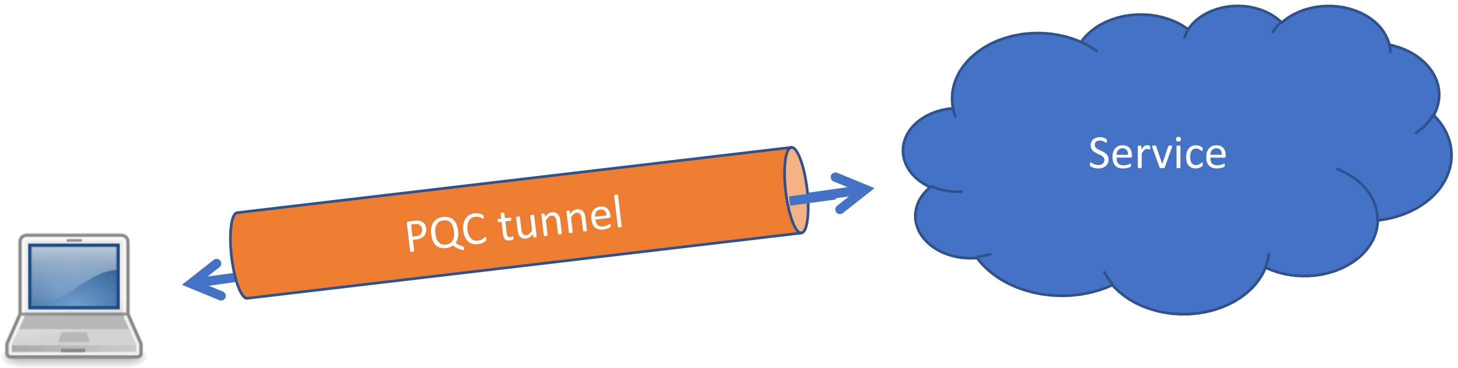
Benchmarking results #2





"The overhead of slower TLS connection establishment diminishes as a proportion of the overall page load time"

PQ VPN tunnels



- OpenVPN 2.4.8 integration
  - Uses OQS's OpenSSL fork
  - Easy legacy app tunneling
  - https://www.microsoft.com/en-us/research/project/post-quantum-crypto-vpn/

- Project Natick PQC VPN experiment
  - Natick is an underwater datacenter module off the coast of Scotland
  - We run a PQ VPN from Redmond
    - Uses ECDHE-P256 + SIKEp434 hybrid
    - Rekeying every hour
  - https://cloudblogs.microsoft.com/quantum/2020/02/26/cryptography-quantum-computers/

Quantum computers are coming…

Let’s make our crypto is *quantum safe!*

https://www.microsoft.com/en-us/research/project/post-quantum-cryptography/

1. Make sure your app/services are crypto agile
2. Start experimenting with PQC
3. Consider early migration to hybrid PQC

cpaquin@microsoft.com
@chpaquin