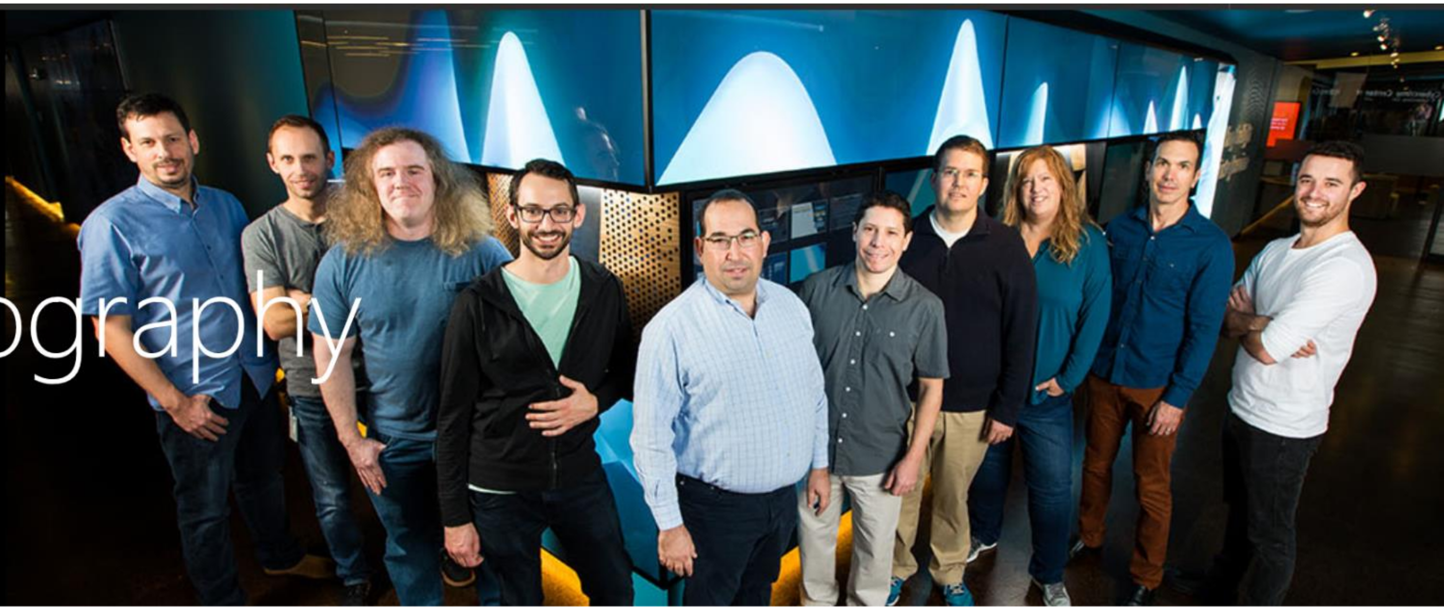
Integrating post-quantum crypto into real-life applications

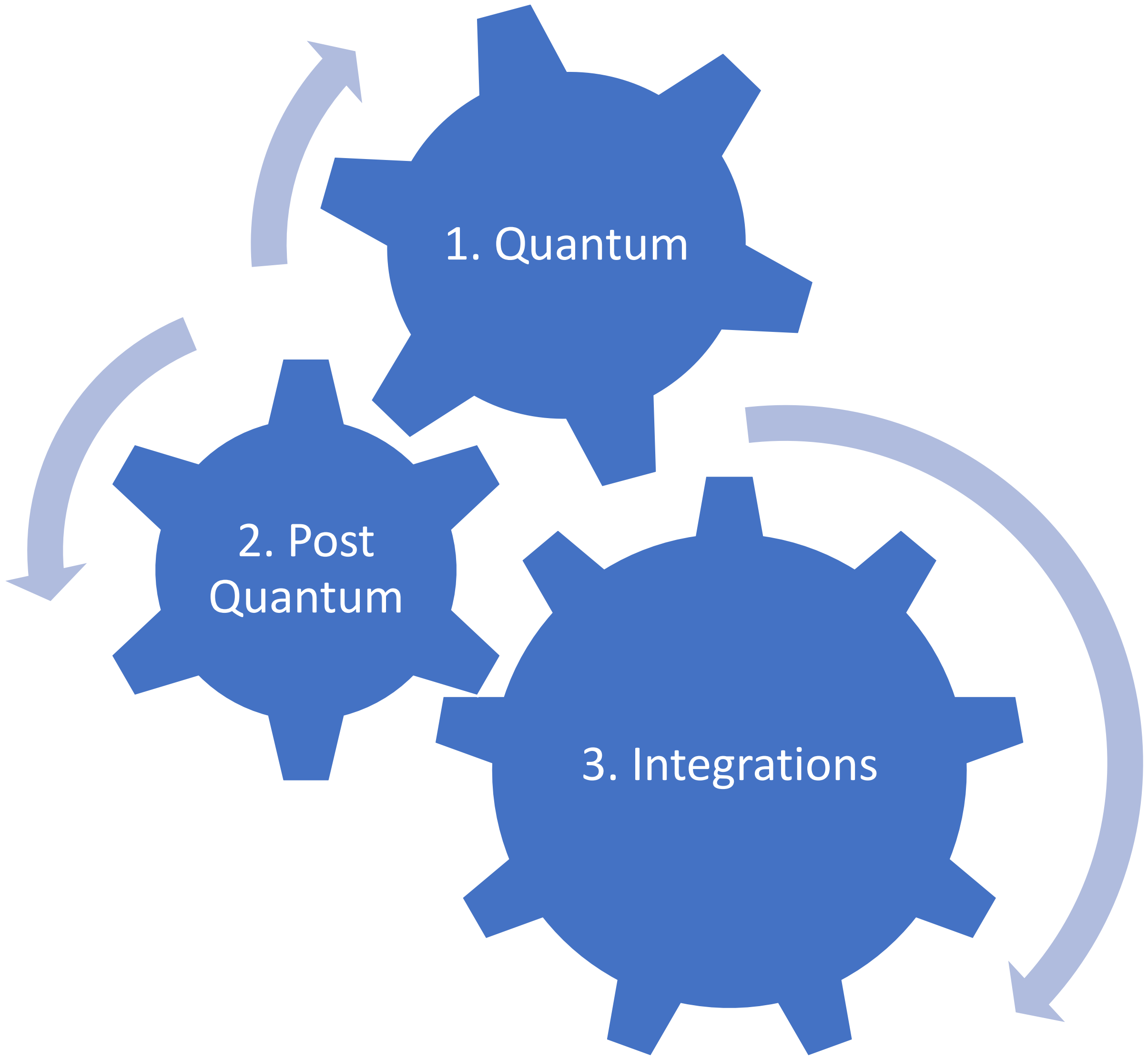
Christian Paquin
@chpaquin
Principal Program Manager
MSR Security & Crypto

CRYPTO + PRIVACY VILLAGE



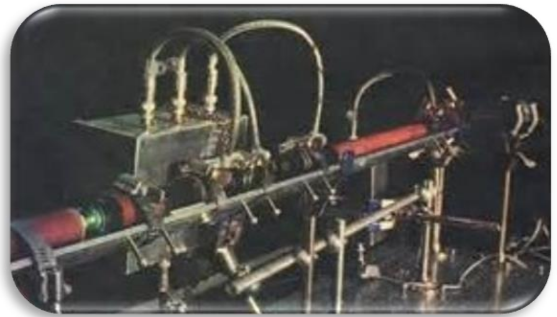
Post-quantum cryptography
Microsoft Research

Outline



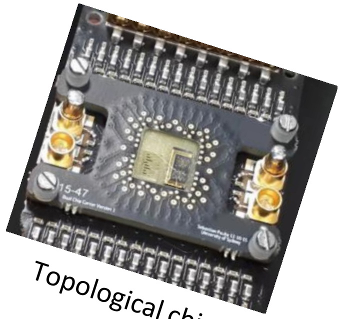
Quantum

“The quantum revolution is coming”



Université de Montréal

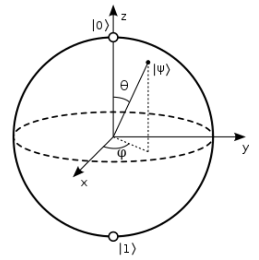
- I’ve been hearing this for 20 years…
  - Studying quantum crypto at UdM with the co-inventor of QKD
- But now, it’s getting serious
  - https://www.bing.com/news/search?q=quantum+computers
- My colleagues are building the full stack: from the chip to an SDK! https://www.microsoft.com/quantum/



Topological chip

Microsoft Quantum Development Kit

Quantum computers



- Computers operating using the laws of quantum physics
- A quantum bit, or *qubit*, can be in *superposition* of the classical states 0 and 1; i.e. it can be both values simultaneously providing intrinsic parallelism

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

- Measurement of a qubit yields a probabilistic classical value depending on the complex amplitudes α and β
  - Quantum algorithms must reinforce the desired computational states
- Qubits can be *entangled*, i.e. be in a shared state across space
  - $|\psi\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$ means either both 0 or 1 with equal probability
- Can be built with various physical particles
  - Electron, photon, anyon (topological)
- "Nobody understands quantum mechanics" – Richard Feynman

Charlie Marcus' lab in Copenhagen

The Quantum Menace

- Quantum computing brings great promises in many fields, but has dire consequences for cryptography
- Shor (1994)
  - Solves the factoring (RSA) and discreet log (DSA, DH, and EC variants) problems in polynomial time
    - Reduce to period finding
  - Affects most of the asymmetric cryptography in use today
- Grover (1996)
  - Speeds up "database search" and "function inversion" in $O(\sqrt{n})$
  - Improves brute force of symmetric cryptography such as hash functions (SHA) and block ciphers (AES)
  - Need to double the size of key/digest: AES128 → AES256

Tic toc…

- Michele Mosca (Waterloo):

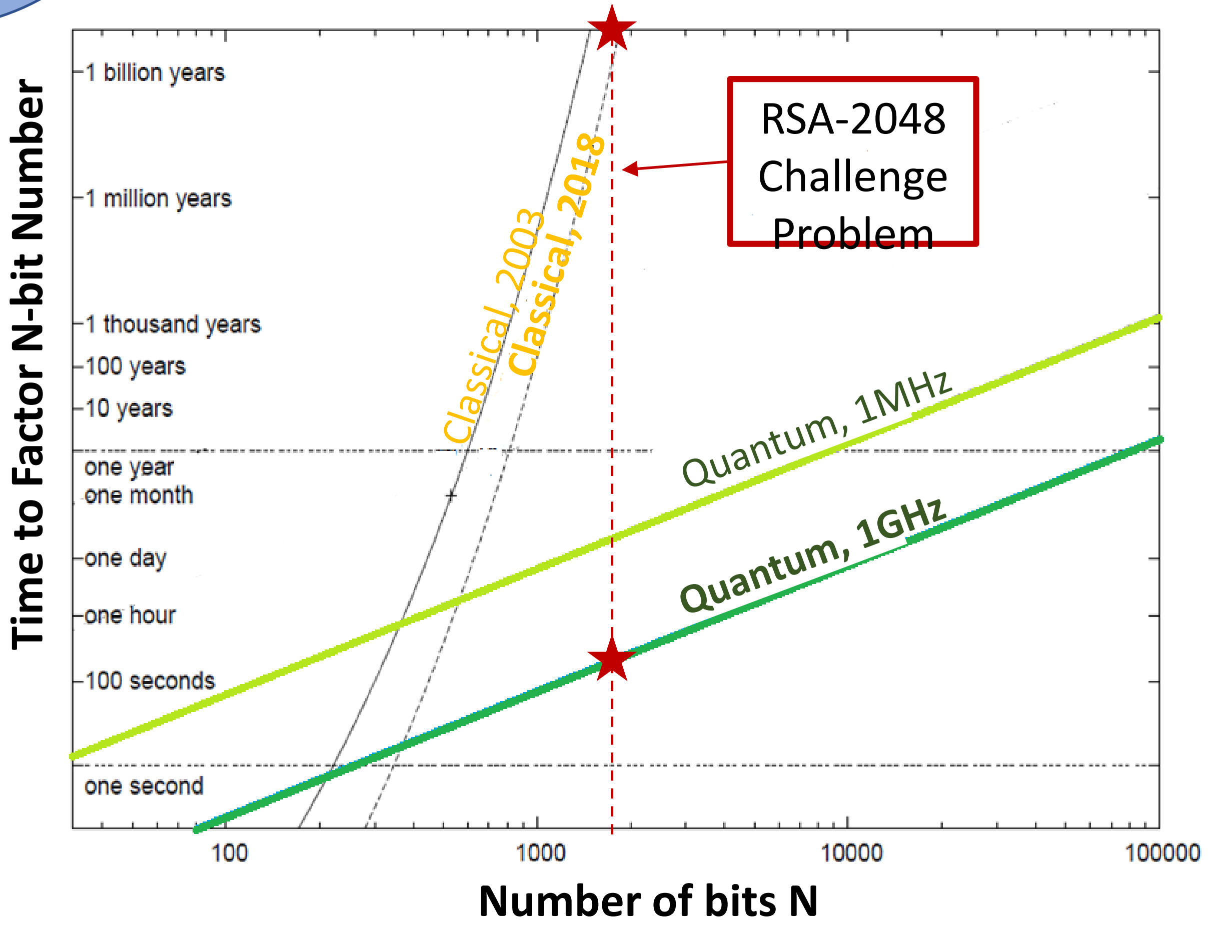
  “1/7 chance of breaking RSA-2048 by 2026, 1/2 chance by 2031” (2015)

  “1/6 chance within 10 years” (2017)

- Simon Benjamin (Oxford):

  “maybe 6-12 years if someone is willing to go Manhattan project”

- My colleagues estimate 2030



We need *quantum-safe* alternatives soon: *post-quantum cryptography!*

Post-quantum

Many reasons to start thinking post-quantum today

- Long-lived secrets/signatures are in danger
  - Capture now, decrypt later
- Need to understand impact on
  - Standards (TLS, SSH, IKE, PKI, S/MIME, …)
  - Products and services
    - Longer key/message/sig sizes
    - Slower running times
    - Code agility
- Early deployment of hybrid scenarios
  - Today’s assurance + safety net against QC

NIST competition

- The National Institute of Standards and Technologies (NIST) started the process to specify Post-Quantum Cryptography

Nov 2017
- Round 1 submissions

Apr 2018
- 1st NIST PQC workshop

2018/2019
- Round 2

Aug 2019
- 2nd NIST PQC workshop

2020/2021
- Round 3

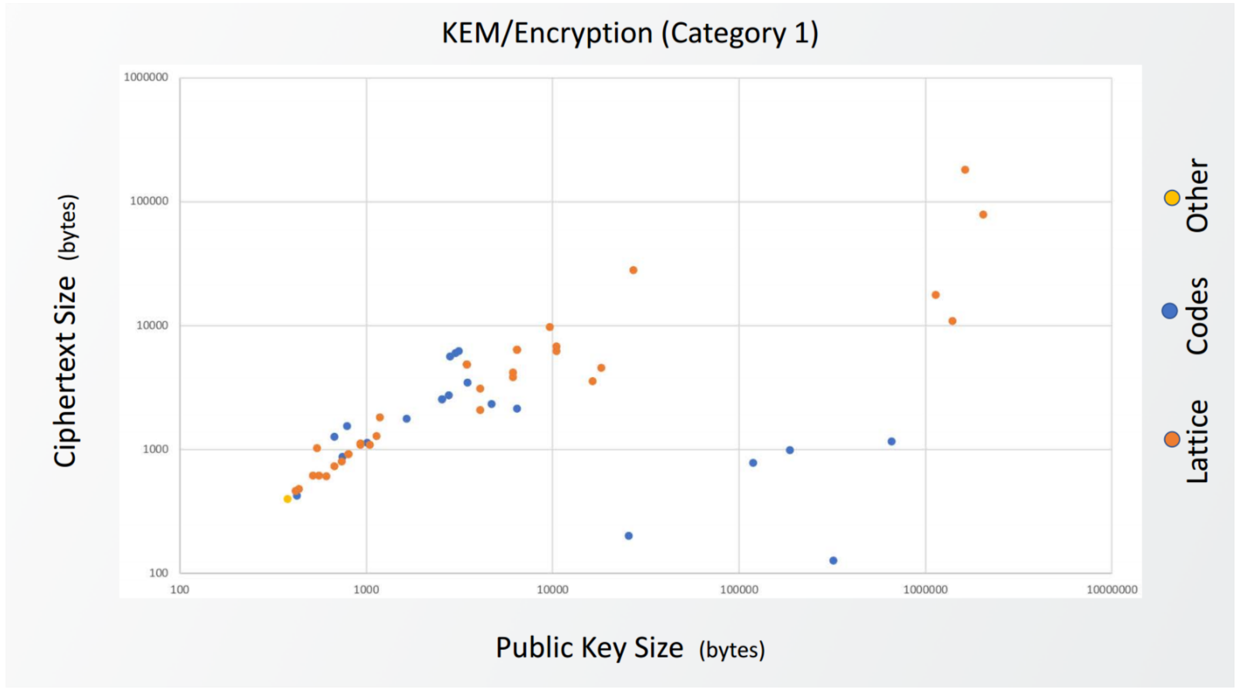
2022-2024
- Draft standards

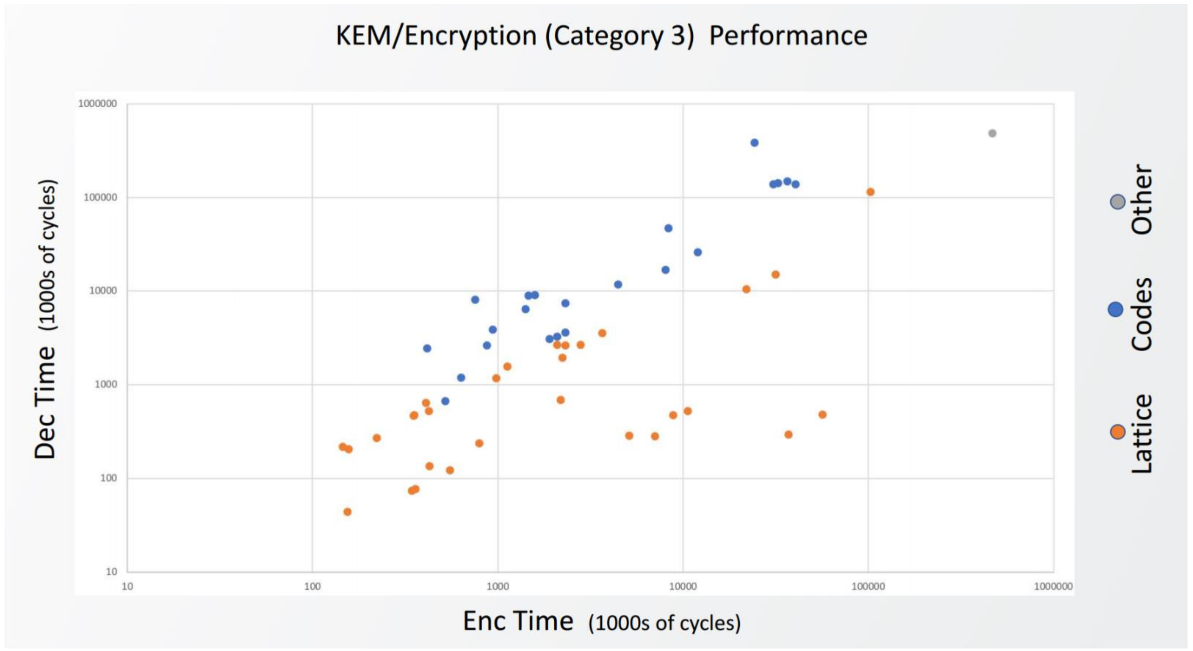
- Looking for signatures, encryption, and key establishments schemes
  - Five levels, corresponding to breaking AES-128/192/256 and SHA-256/384
- 64 submissions remaining (from 69 valid submissions)
  - 19 signature schemes, 45 KEM/encryption schemes
- https://csrc.nist.gov/projects/post-quantum-cryptography

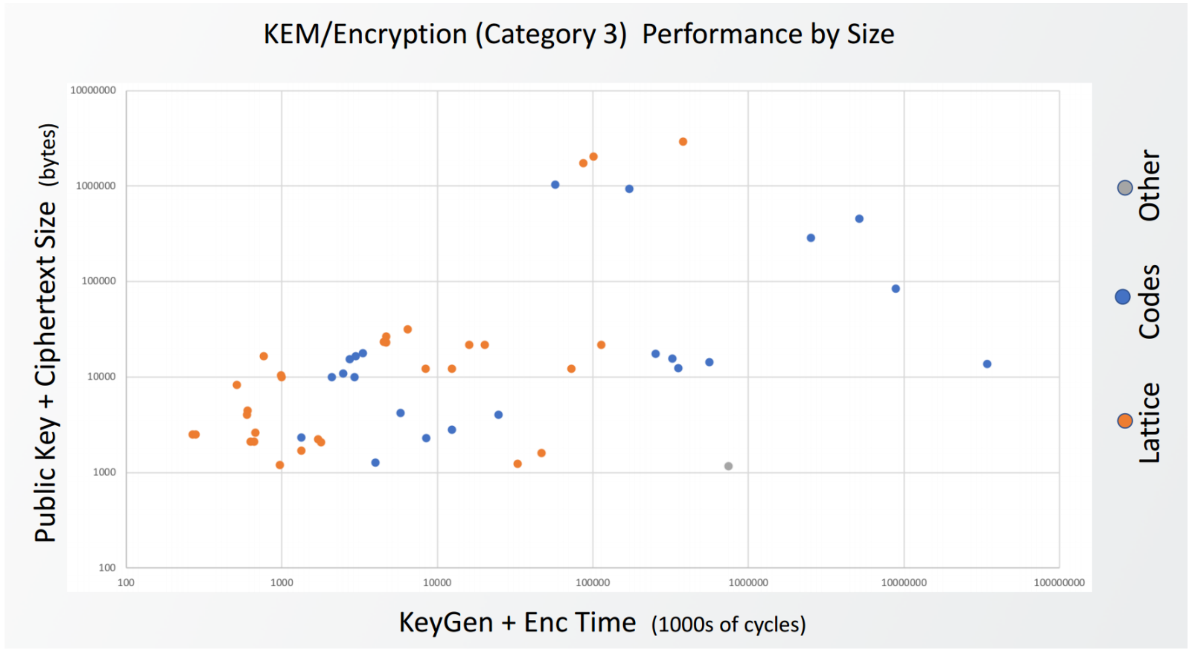
Families of PQC

- Lattice-based systems (26)
  - Encryption/signature based on lattices (NTRU in ’96)
  - Learning With Error (LWE, 2005), or its less secure but more efficient Ring version (R-LWE: Peikert → BCNS → NewHope)
- Code-based (19)
  - Encryption/signature based on error-correcting codes (McEliece, Niederreiter)
  - As old as public-key crypto
- Multivariate-based systems (9)
  - Encryption/signature based on multivariate polynomials over a finite field
  - Developed in 90’s
- Hash-based systems (3)
  - Signatures based on hash functions (Lamport, Merkle)
  - As old as public-key crypto
  - Early standardization candidates: LMS, XMSS
- Others (7)
  - SIDH/SIKE: based on isogenies on elliptic curves
  - Picnic: based on symmetric ciphers and ZK proofs

Encryption perf

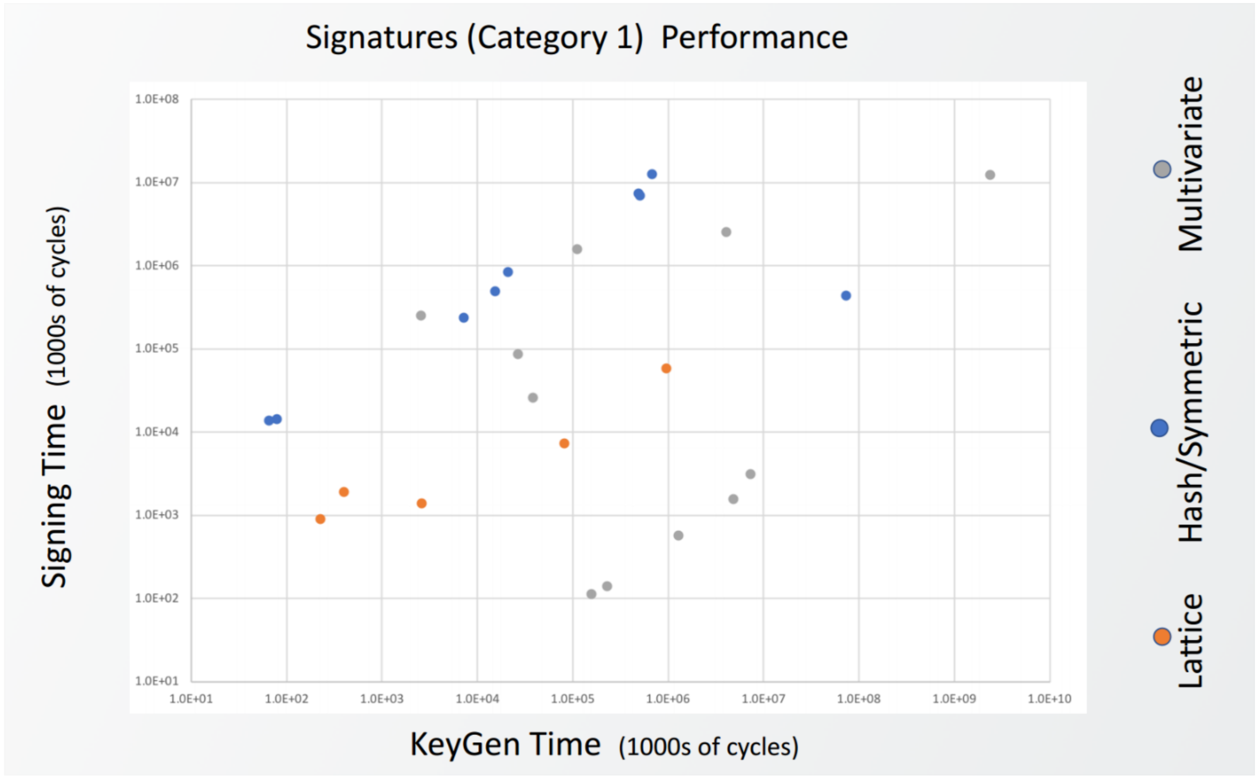


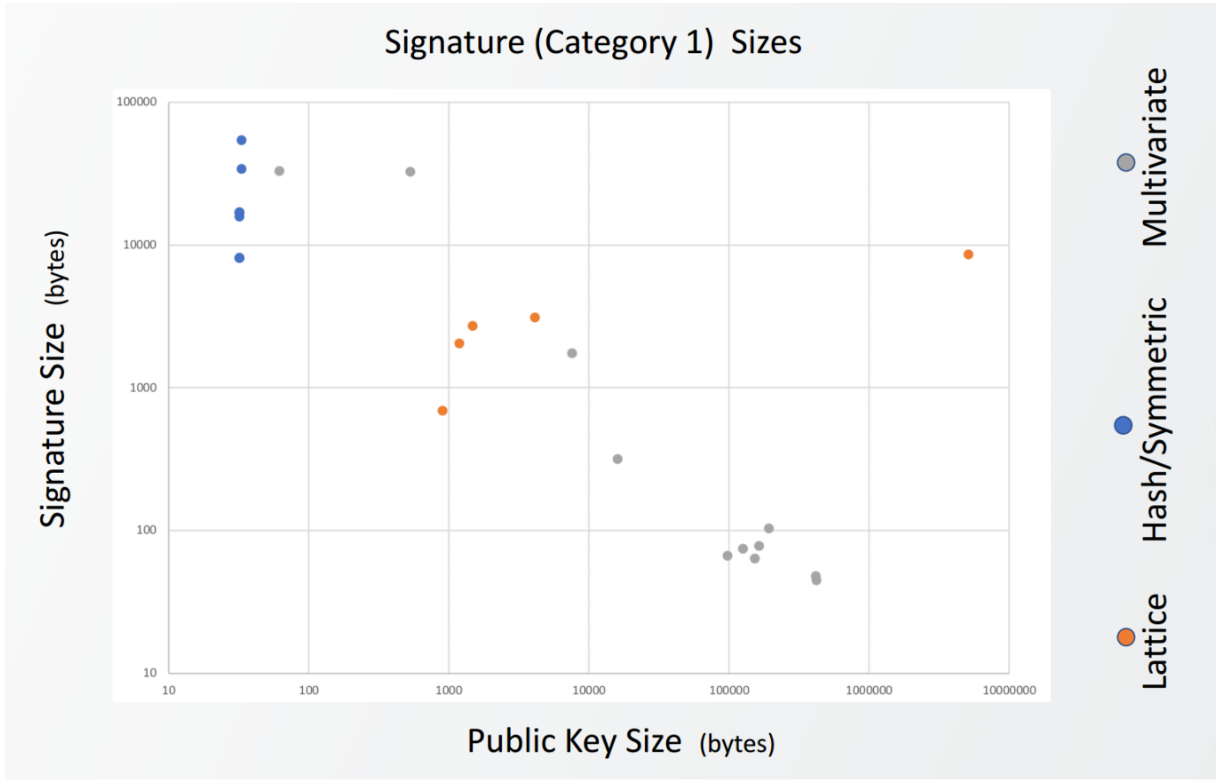




Data from NIST

Signature perf





Data from NIST

MSR’s collaborations

- FRODO (KEM)
  - Learning With Error (LWE) problem
  - 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/
- qTesla (sig)
  - Ring Learning with Error problem
  - https://qtesla.org

Integrations

OPEN QUANTUM SAFE

- Created to simplify integration of PQC into applications
- Multi-org dev team

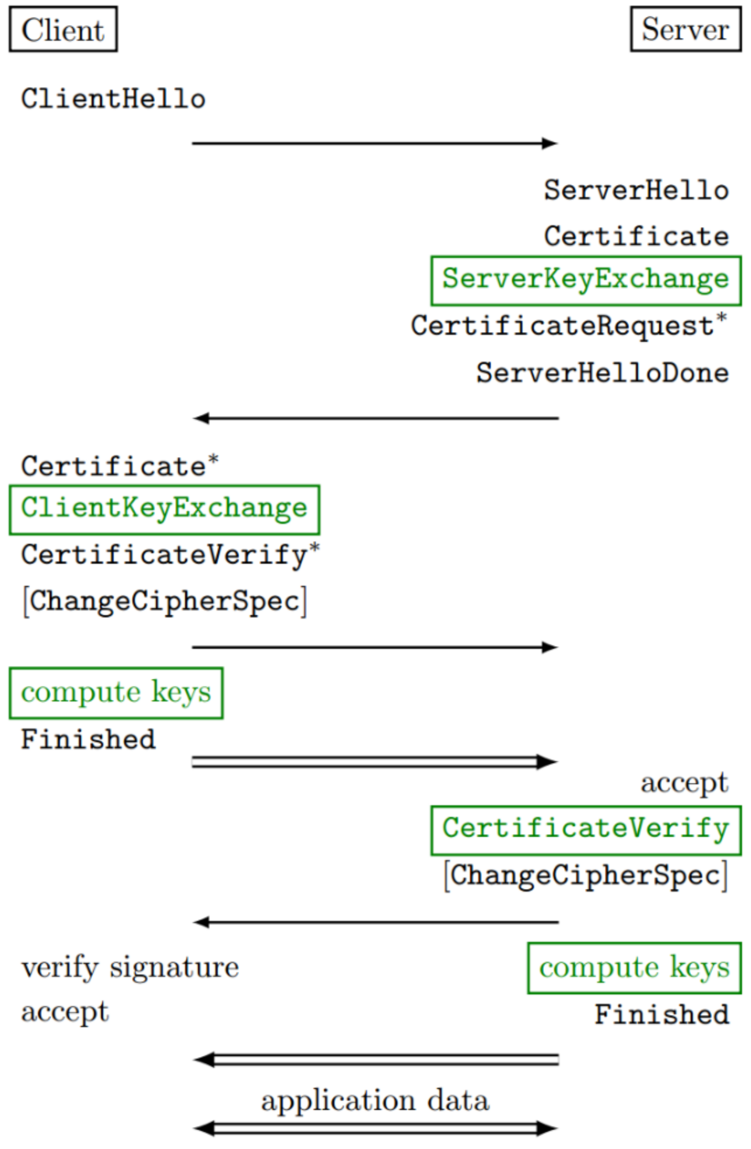
McMaster University   UNIVERSITY OF WATERLOO

Microsoft   evolution   amazon web services™   SRI International

- Master branch (for integration) and NIST branch (for experimentation)
- Shipped integrations with OpenSSL, OpenSSH
  - More in the pipeline
- https://openquantumsafe.org/

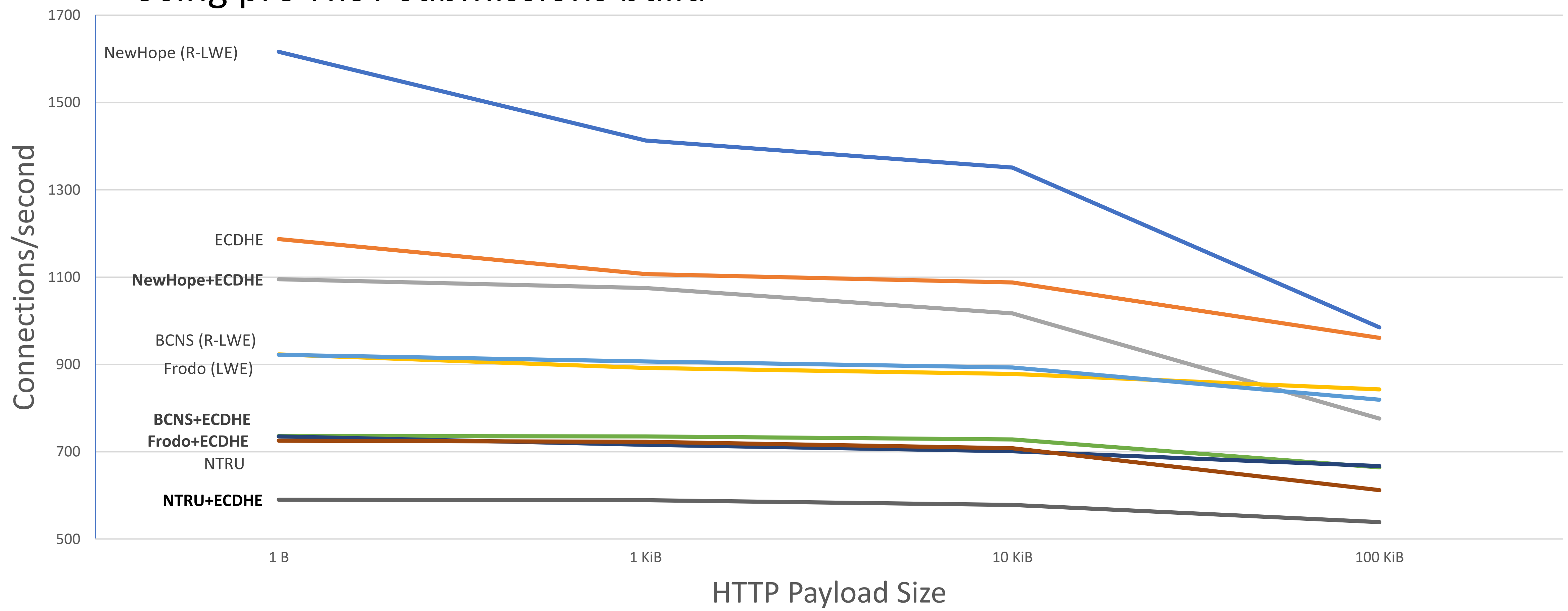
TLS 1.2 integration

- Added OQS key exchange (KEX) and authentication algs to OpenSSL 1.0.2
  - libcrypto: modified signature and X.509
  - libssl: modified TLS handler
- Defined new cipher suites
  - PQ or hybrid Key Exchange (KEX), e.g.
    - OQSKEX-SIDH-PICNIC-AES256-GCM-SHA384, OQSKEX-SIDH-<u>ECDHE</u>-PICNIC-AES256-GCM-SHA384
    - Pre-master secret := ECDH secret || PQ secret
  - Classical or PQ auth (Picnic)
    - Challenge: sig size limit of $2^{16} - 1$ bytes
- Tested with Apache 2.4.25
- https://github.com/open-quantum-safe/openssl
  - Branch: OpenSSL_1_0_2-stable



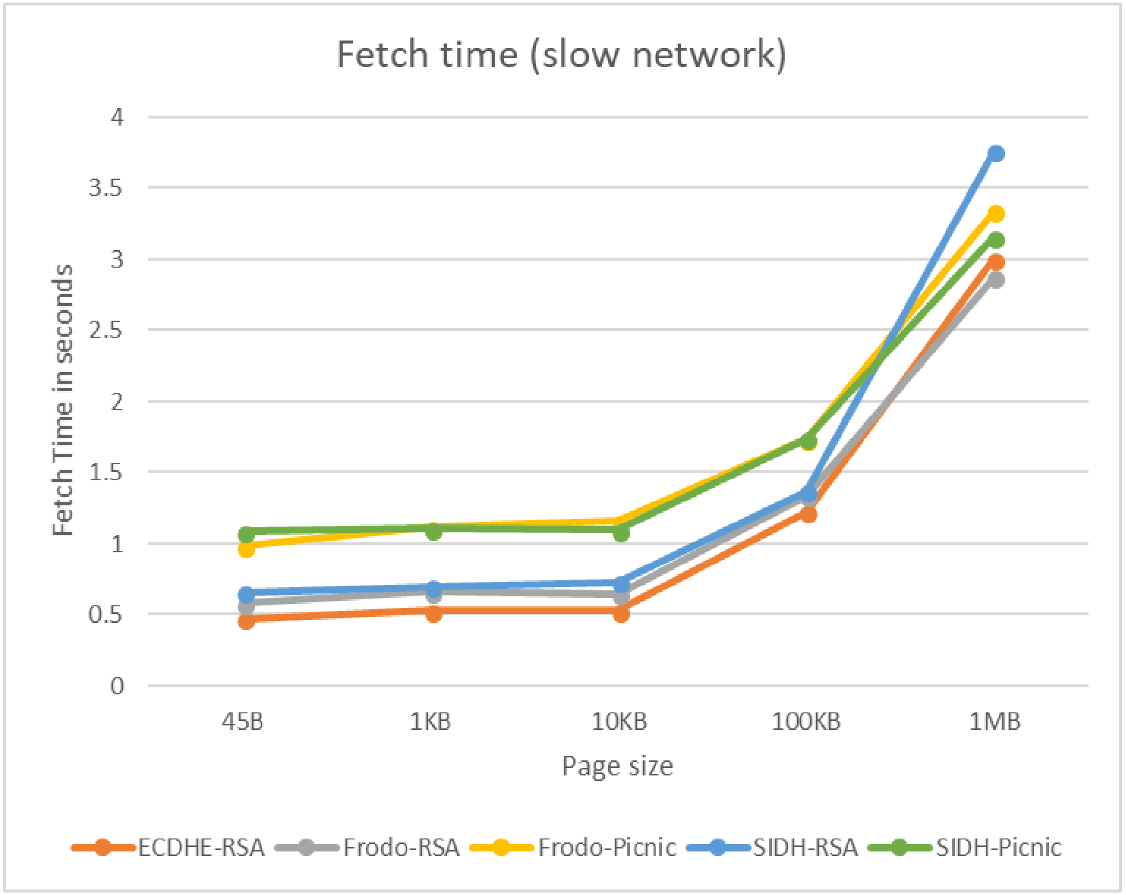
TLS 1.2 KEX performance

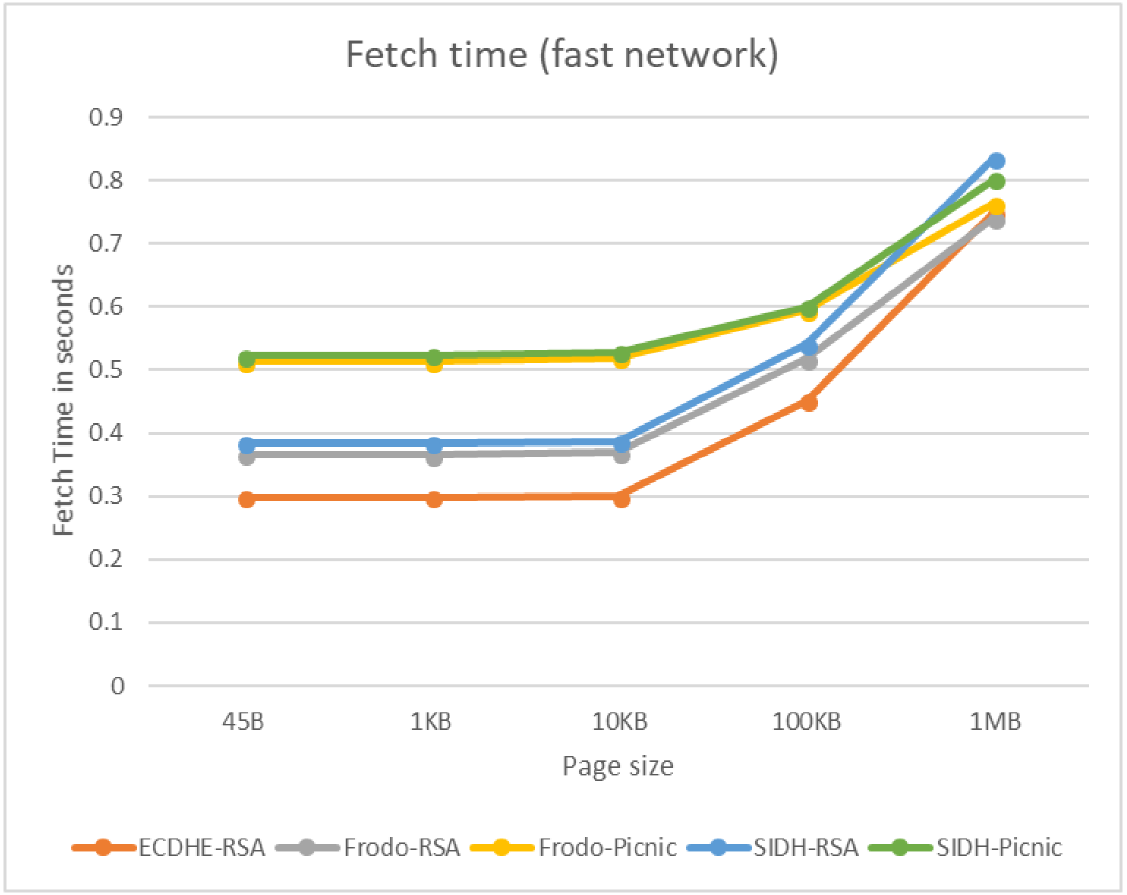
- Measurements from OQS-enabled Apache server and test client
  - Using pre-NIST submissions build



TLS 1.2 Auth (Picnic) performance

- Fetch time for various pages on slow/fast network

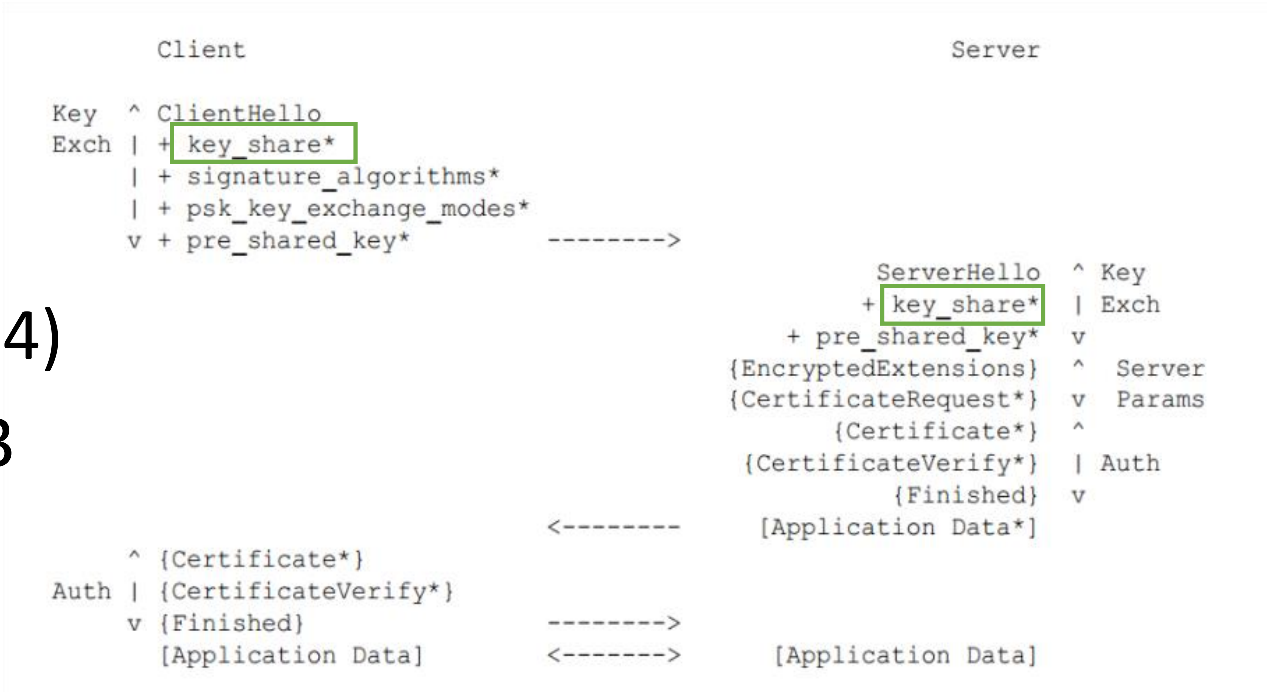




c.f. NIST Picnic submission

TLS 1.3 integration

- Added OQS key exchange (KEX) and auth to OpenSSL 1.1.1 (beta4)
- Defined new “curves” for TLS 1.3
  - PQ or hybrid Key Exchange (KEX)
- Tested with nginx 1.5.0
- We need extensions to enable PQC in TLS 1.3
- Details on OQS’s page
  - https://github.com/open-quantum-safe/openssl/wiki/PQC-integration-into-TLS-1.3
  - Branch: OQS-master



```
          Client                                           Server

   Key  ^ ClientHello
   Exch | + key_share*
        | + signature_algorithms*
        | + psk_key_exchange_modes*
        v + pre_shared_key*       -------->
                                                      ServerHello  ^ Key
                                                     + key_share*  | Exch
                                                + pre_shared_key*  v
                                            {EncryptedExtensions}  ^  Server
                                            {CertificateRequest*}  v  Params
                                                   {Certificate*}  ^
                                             {CertificateVerify*}  | Auth
                                                       {Finished}  v
                                  <--------  [Application Data*]
        ^ {Certificate*}
   Auth | {CertificateVerify*}
        v {Finished}              -------->
          [Application Data]      <------->  [Application Data]
```

TLS 1.3 Demo

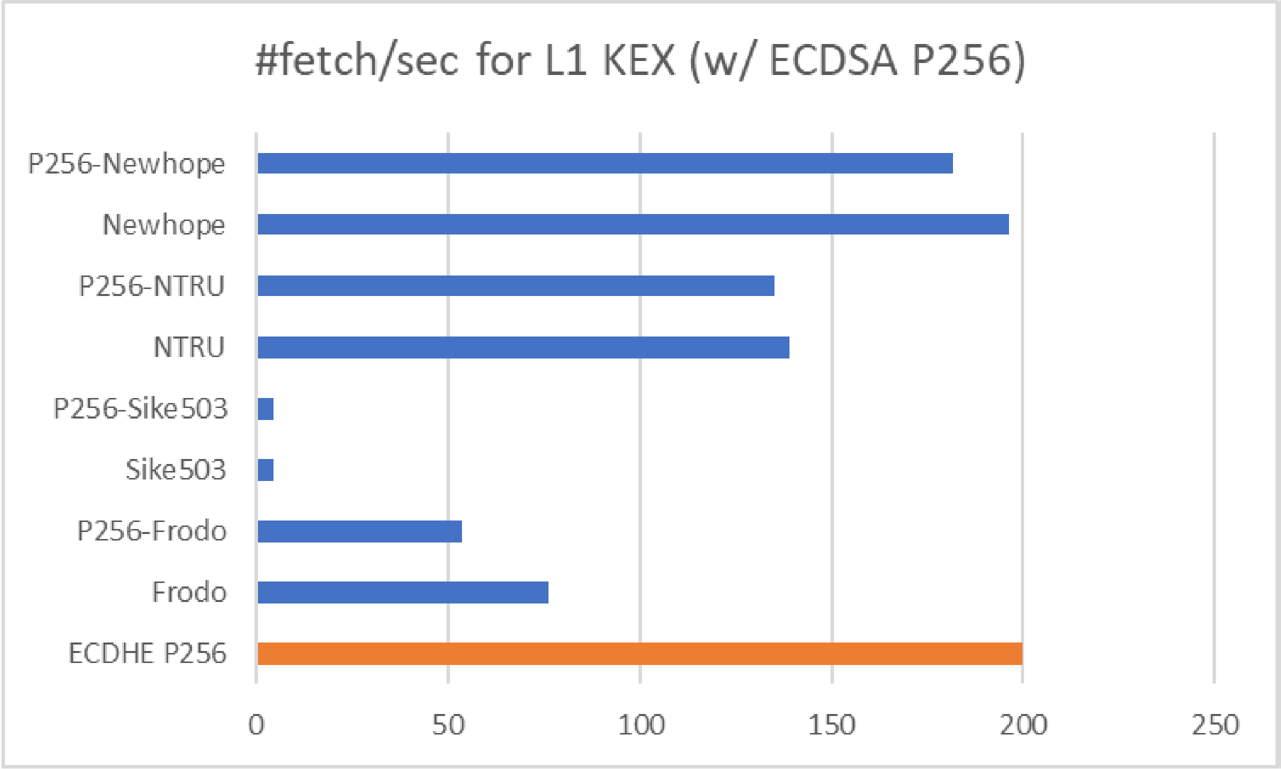
DEMO

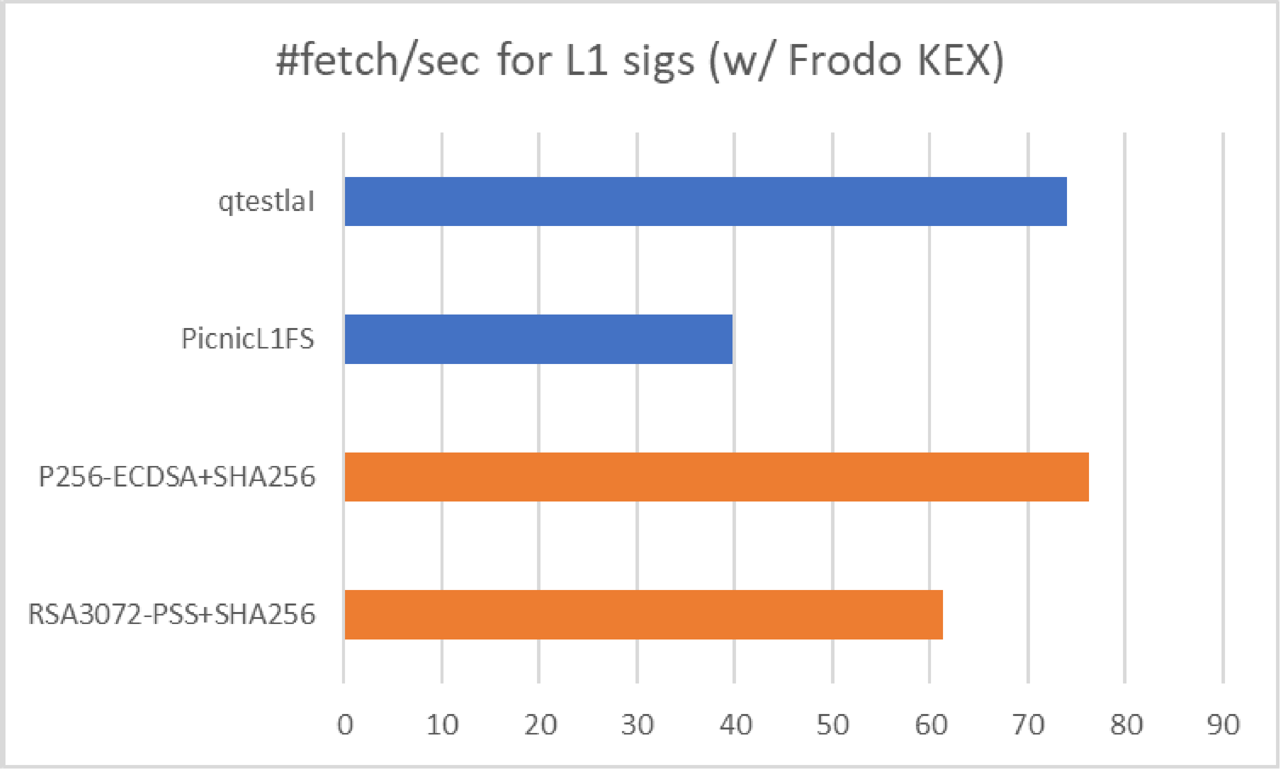
Hybrid scenarios

- TLS 1.3 KEX, two approaches
  - Naïve: define combo schemes and concatenate the data (currently implemented)
  - Multiple key shares (classical and PQC) both updating the master secret
    - State machine already supports hybrid keys, for PSK + ECDHE
    - PQC proposals: draft-whyte-qsh-tls13-06, draft-schanck-tls-additional-keyshare-00
- PKI, need to convey a classical and PQC signature
  - Hybrid signature scheme
  - Convey two certs
  - TLS PQC cert extension
  - X.509 extension for an extra PQC key
  - Bindel, Herath, McKague, Stebila; Transitioning to QR PKI

TLS 1.3 Perf

Measurements with client/server on localhost (no network delay)



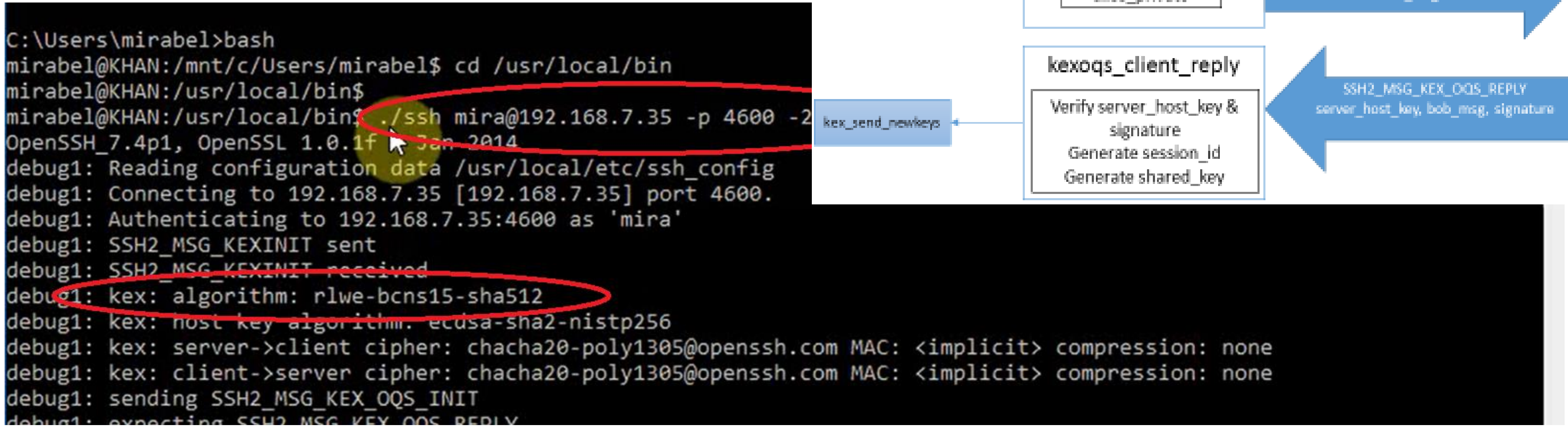


Classical 🟦 PQC 🟧

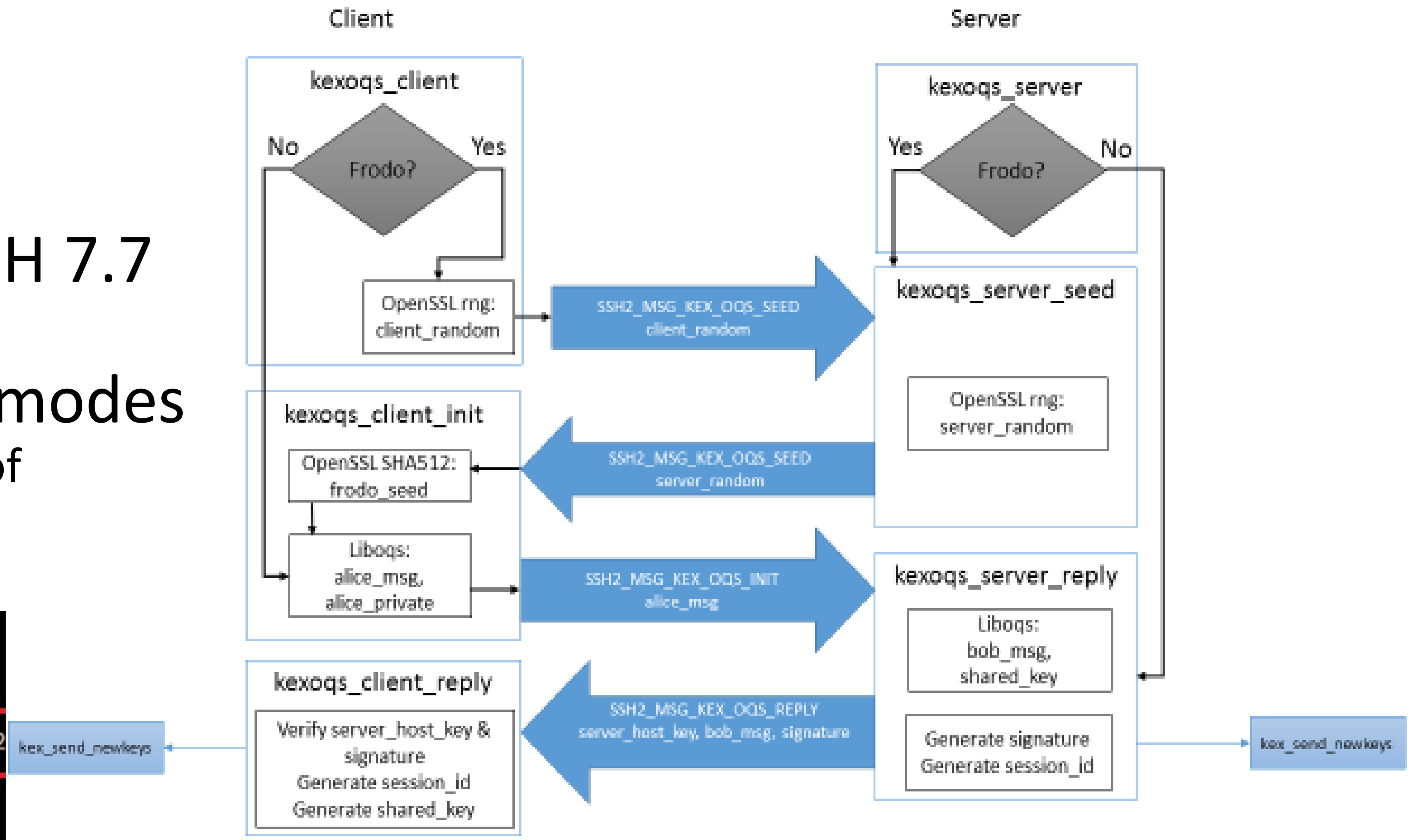
July 15th built of OQS/OpenSSL
Azure Standard D4s v3 VM, Ubuntu OS

SSH integration

- Integrated OQS in OpenSSH 7.7
  - KEX algs from master branch
- Supports PQC and hybrid modes
  - Shared secret = concatenation of classical & PQC shared secrets



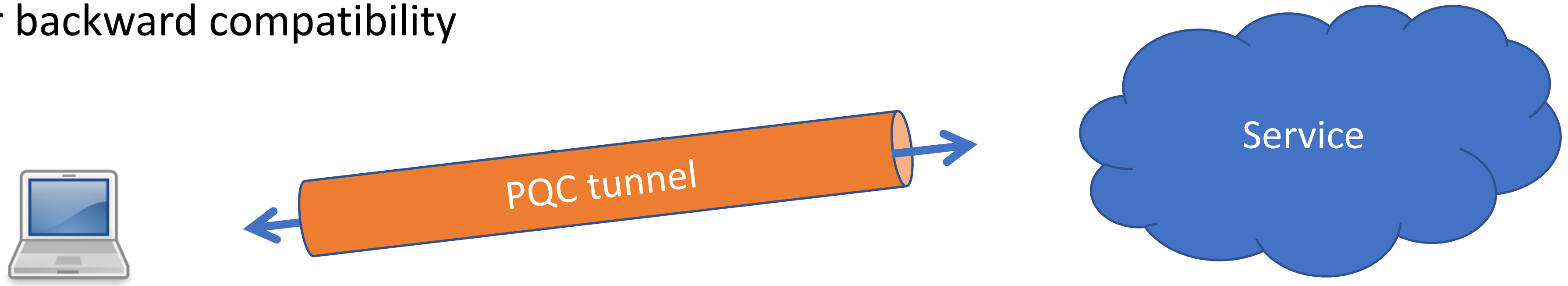
```
C:\Users\mirabel>bash
mirabel@KHAN:/mnt/c/Users/mirabel$ cd /usr/local/bin
mirabel@KHAN:/usr/local/bin$
mirabel@KHAN:/usr/local/bin$ ./ssh mira@192.168.7.35 -p 4600 -2
OpenSSH_7.4p1, OpenSSL 1.0.1f  Jan 2014
debug1: Reading configuration data /usr/local/etc/ssh_config
debug1: Connecting to 192.168.7.35 [192.168.7.35] port 4600.
debug1: Authenticating to 192.168.7.35:4600 as 'mira'
debug1: SSH2_MSG_KEXINIT sent
debug1: SSH2_MSG_KEXINIT received
debug1: kex: algorithm: rlwe-bcns15-sha512
debug1: kex: host key algorithm: ecdsa-sha2-nistp256
debug1: kex: server->client cipher: chacha20-poly1305@openssh.com MAC: <implicit> compression: none
debug1: kex: client->server cipher: chacha20-poly1305@openssh.com MAC: <implicit> compression: none
debug1: sending SSH2_MSG_KEX_OQS_INIT
debug1: expecting SSH2_MSG_KEX_OQS_REPLY
```


Client
Server
kexoqs_client
No
Frodo?
Yes
OpenSSL rng: client_random
SSH2_MSG_KEX_OQS_SEED client_random
kexoqs_server
Yes
Frodo?
No
kexoqs_server_seed
OpenSSL rng: server_random
kexoqs_client_init
OpenSSL SHA512: frodo_seed
SSH2_MSG_KEX_OQS_SEED server_random
Liboqs: alice_msg, alice_private
SSH2_MSG_KEX_OQS_INIT alice_msg
kexoqs_server_reply
Liboqs: bob_msg, shared_key
Generate signature Generate session_id
kexoqs_client_reply
Verify server_host_key & signature Generate session_id Generate shared_key
SSH2_MSG_KEX_OQS_REPLY server_host_key, bob_msg, signature
kex_send_newkeys
kex_send_newkeys

https://github.com/open-quantum-safe/openssh-portable

OpenVPN

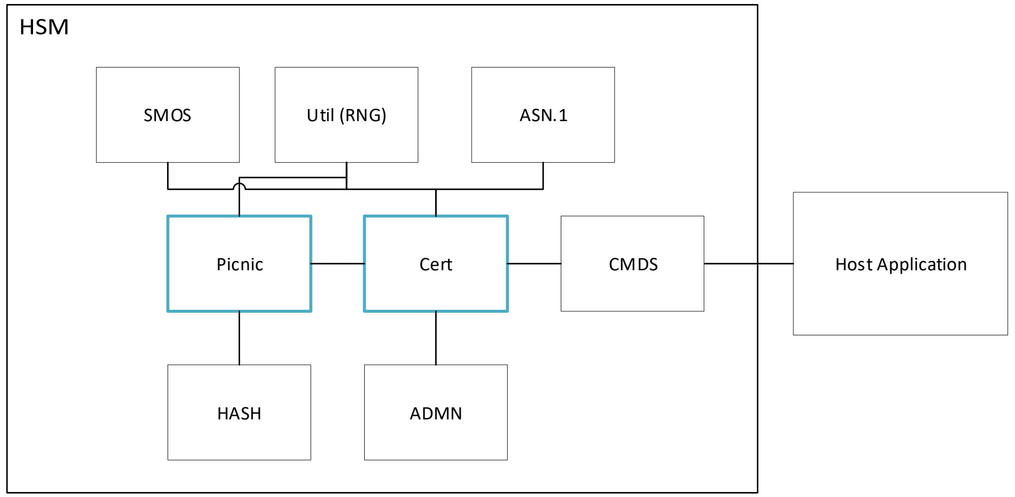
- Integration in OpenVPN 2.4.4
  - Uses OQS-OpenSSL to protect TLS key establishment
  - Uses RSA or Picnic auth
- Easy way to achieve PQC tunnel to the cloud even if applications haven’t been updated
  - Good for backward compatibility



- Tested with Raspberry Pi and Windows clients, and Azure Linux VM service
- https://github.com/Microsoft/PQCrypto-VPN

HSM integration

- Integrated Picnic into an Utimaco HSM (Security Server Se50 LAN v4)
- Experiment consisted of
  1. Picnic key generation and signing in HSM (using reference implementation)
  2. Generated self-signed root Picnic cert
  3. Issued end-user RSA certs using the Picnic cert
- https://microsoft.github.io/Picnic/



The road ahead

- Start planning transition to PQC
- Make sure your apps/services are crypto agile
- Consider deploying hybrid solutions for long-lived, high-value data
- Consider wrapping long-tail apps/services in a PQC-VPN tunnel

Questions?

cpaquin@microsoft.com