# A Quantum of QUIC: Dissecting Cryptography with Post-Quantum Insights

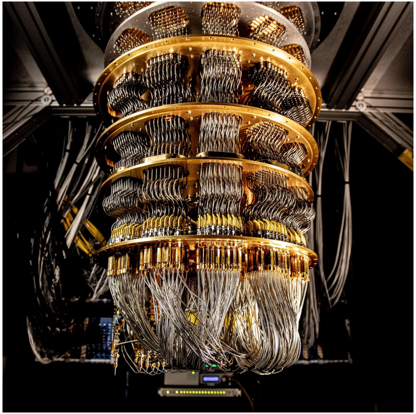
**Marcel Kempf**, Nikolas Gauder, Benedikt Jaeger, Johannes Zirngibl, Georg Carle

Tuesday 15th October, 2024

Munich Internet Research Retreat Raitenhaslach 2024

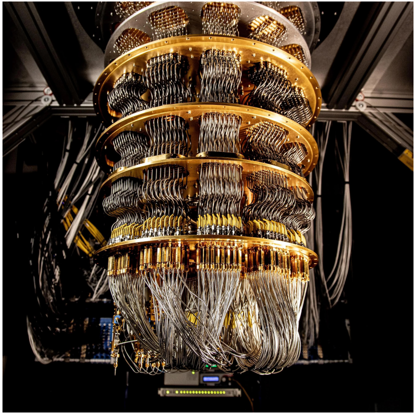
Chair of Network Architectures and Services
School of Computation, Information, and Technology
Technical University of Munich

# Motivation



# Motivation

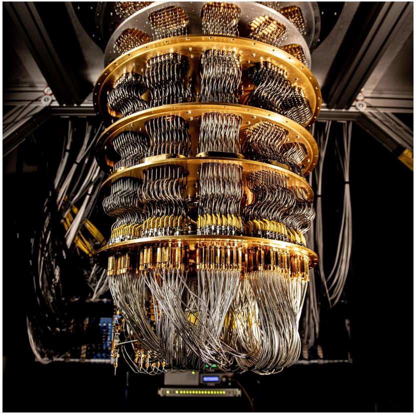
- Traditional asymmetric cryptography will be broken by quantum computers!
  - Post-quantum cryptography (PQC) algorithms exist
  - NIST competition: winners will be standardized



© Erik Lucero, Google Quantum AI

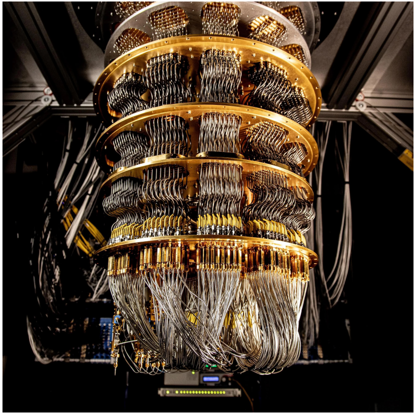
# Motivation

- Traditional asymmetric cryptography will be broken by quantum computers!
  - Post-quantum cryptography (PQC) algorithms exist
  - NIST competition: winners will be standardized
- PQC also comes with drawbacks
  - Significantly larger keys
  - More messages to exchange



# Motivation

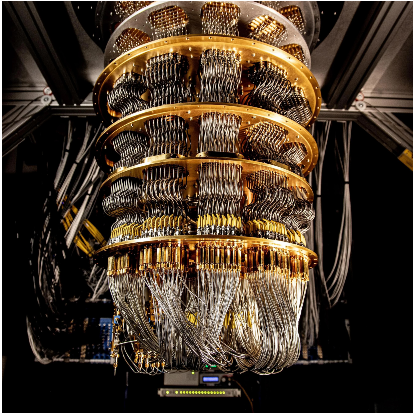
- Traditional asymmetric cryptography will be broken by quantum computers!
  - Post-quantum cryptography (PQC) algorithms exist
  - NIST competition: winners will be standardized
- PQC also comes with drawbacks
  - Significantly larger keys
  - More messages to exchange
- And why should we care now?
  - "Store now, decrypt later" attacks
  - Understand PQC in practice



© Erik Lucero, Google Quantum AI

# Motivation

- Traditional asymmetric cryptography will be broken by quantum computers!
  - Post-quantum cryptography (PQC) algorithms exist
  - NIST competition: winners will be standardized
- PQC also comes with drawbacks
  - Significantly larger keys
  - More messages to exchange
- And why should we care now?
  - "Store now, decrypt later" attacks
  - Understand PQC in practice
- QUIC is the new general-purpose transport protocol
  - QUIC includes TLS 1.3 and enforces encryption
  - Encryption and authentication, packet & header protection



© Erik Lucero, Google Quantum AI

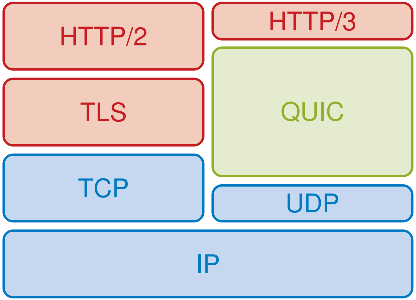
# Research Goals and Questions

- How much does QUIC's cryptography affect the performance?
- How big is the impact of the different security features?
- How does this change with PQC?
- Does the integration of PQC into QUIC lead to problems?

# Background

## What is QUIC?

- New protocol designed as replacement for TCP / TLS
- Standardized in May 2021 by the IETF as RFC 9000
- Implemented in user space on top of UDP
- Several implementations exist
- Includes multiple TCP features **and TLS 1.3**

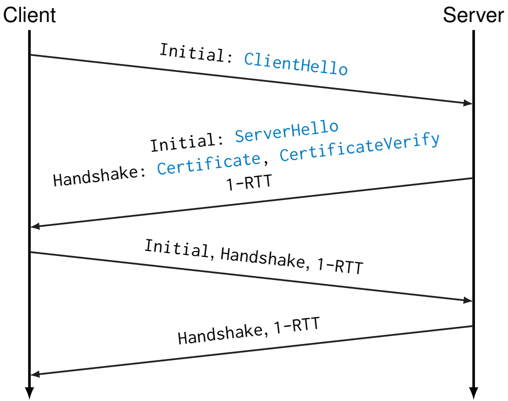



## Applications of QUIC:

- Transport protocol for HTTP/3
- MASQUE (*Apple iCloud Private Relay*, *Cloudflare WARP*)

# Background

## The QUIC Handshake




Asymmetric cryptography only in blue parts:

- `Client/ServerHello` affected by PQ key encapsulation mechanisms
- `Certificate(Verify)` affected by PQ signature schemes

# Related Work

| Paper | Content |
| --- | --- |
| [Jaeger23] | • Performance analysis of QUIC implementations on physical hardware<br>• Impact of cryptographic operation broadly analyzed |
| [Yang20] | • Performance analysis of QUIC implementations<br>• NIC offloading context |
| [Sosnowski23] | • Comparison of post-quantum cipher suites<br>• Physical hardware + network emulation<br>• TCP as transport protocol |
| This paper | • Deep performance analysis of cryptography in QUIC<br>• Measurements on physical hardware & links<br>• Symmetric and asymmetric cryptography covered |


[Jaeger23] B. Jaeger et al. **"QUIC on the Highway: Evaluating Performance on High-rate Links"**, IFIP Networking 2023

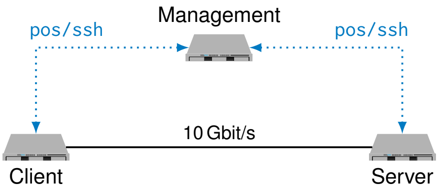
[Yang20] X. Yang et al. **"Making QUIC Quicker With NIC Offload"**, EPIQ 2020

[Sosnowski23] M. Sosnowski et al. **"The Performance of Post-Quantum TLS 1.3"**, CoNEXT 2023

# Framework for QUIC Performance Measurements

## Modified QUIC Interop Runner

- Dedicated physical hosts for client and server
- Experiment orchestration via pos [1]
- Collect CPU, OS, and NIC metrics using various tools
- Flexibility, Portability, Reproducibility




## Custom TLS Libraries

- NOOP cipher implemented into *OpenSSL* and *BoringSSL*
  - No encryption/decryption happening
  - Only `memcpy()` operation
- Fork of BoringSSL [2] for post-quantum ciphers


[1] S. Gallenmüller et al., “The pos Framework: A Methodology and Toolchain for Reproducible Network Experiments, CoNEXT 2021

[2] https://github.com/open-quantum-safe/boringssl

# Framework for QUIC Performance Measurements

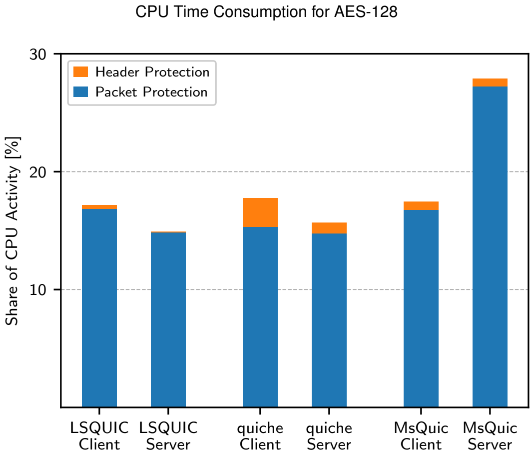
## Evaluated Implementations

| Name | Language | Developer | TLS Library | No-Crypto Mode |
|---|---|---|---|---|
| LSQUIC | C | LiteSpeed Technologies | BoringSSL | × |
| quiche | Rust | Cloudflare | BoringSSL | × |
| MsQuic | C | Microsoft | OpenSSL | ✓ |

# Evaluation

## Symmetric Cryptography in QUIC – CPU Time Consumption

- Packet protection and header protection
- CPU profiling with *perf*
- AES, ChaCha20 and NOOP cipher analyzed
  - ChaCha20: 9 % to 16 % slower than AES
  - NOOP: 10 % to 20 % faster than AES

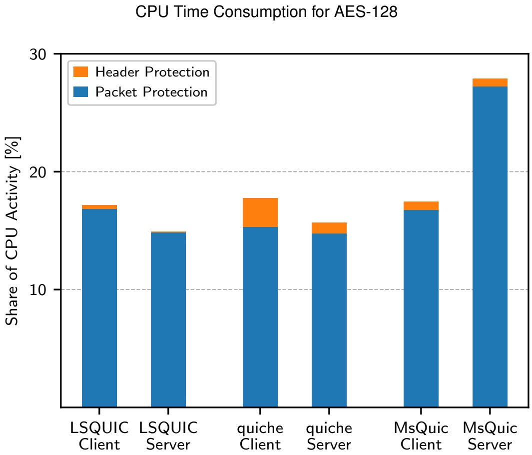


## Symmetric Cryptography in QUIC – CPU Time Consumption

- Packet protection and header protection
- CPU profiling with *perf*
- AES, ChaCha20 and NOOP cipher analyzed
    - ChaCha20: 9 % to 16 % slower than AES
    - NOOP: 10 % to 20 % faster than AES

**Take-away Result:**

- QUIC's header protection is basically free, especially with AES
- Performance impact of AES key size is negligible



## PQC – Key Encapsulation Mechanisms

- Used to transmit keys for symmetric cryptography
- Kyber, BIKE and HQC analyzed
  - Kyber selected for standardization by NIST
  - Kyber was renamed to ML-KEM
- RSA-2048 certificate used for measurements
- Elliptic curve Diffie-Hellman Exchange (ECDHE) as baseline (**bold**)

# Evaluation

## PQC – Key Encapsulation Mechanisms

- Used to transmit keys for symmetric cryptography
- Kyber, BIKE and HQC analyzed
  - Kyber selected for standardization by NIST
  - Kyber was renamed to ML-KEM
- RSA-2048 certificate used for measurements
- Elliptic curve Diffie-Hellman Exchange (ECDHE) as baseline (**bold**)

| | Algorithm | TTFB [ms] | |
|---|---|---|---|
| | | LSQUIC | quiche |
| I | **X25519** | 3.91 | 3.57 |
| | Kyber512 | 4.08 | 3.39 |
| | BIKE-L1 | 6.59 | 5.86 |
| | HQC-128 | 5.57 | 4.21 |
| | **P-256** | 3.90 | 3.49 |
| | P-256 + Kyber512 | 4.43 | 3.74 |
| | P-256 + BIKE-L1 | 6.95 | 6.27 |
| | P-256 + HQC-128 | 5.99 | 4.52 |
| III | Kyber768 | 4.23 | 3.78 |
| | BIKE-L3 | 11.75 | 10.49 |
| | HQC-192 | 7.57 | 4.81 |
| | **P-384** | 7.36 | 6.76 |
| | P-384 + Kyber768 | 8.99 | 8.67 |
| | ... | | |

## PQC – Key Encapsulation Mechanisms

- Used to transmit keys for symmetric cryptography
- Kyber, BIKE and HQC analyzed
  - Kyber selected for standardization by NIST
  - Kyber was renamed to ML-KEM
- RSA-2048 certificate used for measurements
- Elliptic curve Diffie-Hellman Exchange (ECDHE) as baseline (**bold**)

| | Algorithm | TTFB [ms] | |
|---|---|---|---|
| | | LSQUIC | quiche |
| I | **X25519** | 3.91 | 3.57 |
| | Kyber512 | 4.08 | 3.39 |
| | BIKE-L1 | 6.59 | 5.86 |
| | HQC-128 | 5.57 | 4.21 |
| | **P-256** | 3.90 | 3.49 |
| | P-256 + Kyber512 | 4.43 | 3.74 |
| | P-256 + BIKE-L1 | 6.95 | 6.27 |
| | P-256 + HQC-128 | 5.99 | 4.52 |
| III | Kyber768 | 4.23 | 3.78 |
| | BIKE-L3 | 11.75 | 10.49 |
| | HQC-192 | 7.57 | 4.81 |
| | **P-384** | 7.36 | 6.76 |
| | P-384 + Kyber768 | 8.99 | 8.67 |
| | ... | | |

## PQC – Key Encapsulation Mechanisms

- Used to transmit keys for symmetric cryptography
- Kyber, BIKE and HQC analyzed
  - Kyber selected for standardization by NIST
  - Kyber was renamed to ML-KEM
- RSA-2048 certificate used for measurements
- Elliptic curve Diffie-Hellman Exchange (ECDHE) as baseline (**bold**)

| | Algorithm | TTFB [ms] | |
|---|---|---|---|
| | | LSQUIC | quiche |
| I | **X25519** | 3.91 | 3.57 |
| | Kyber512 | 4.08 | 3.39 |
| | BIKE-L1 | 6.59 | 5.86 |
| | HQC-128 | 5.57 | 4.21 |
| | **P-256** | 3.90 | 3.49 |
| | P-256 + Kyber512 | 4.43 | 3.74 |
| | P-256 + BIKE-L1 | 6.95 | 6.27 |
| | P-256 + HQC-128 | 5.99 | 4.52 |
| III | Kyber768 | 4.23 | 3.78 |
| | BIKE-L3 | 11.75 | 10.49 |
| | HQC-192 | 7.57 | 4.81 |
| | **P-384** | 7.36 | 6.76 |
| | P-384 + Kyber768 | 8.99 | 8.67 |
| | ... | | |

## PQC – Key Encapsulation Mechanisms

- Used to transmit keys for symmetric cryptography
- Kyber, BIKE and HQC analyzed
  - Kyber selected for standardization by NIST
  - Kyber was renamed to ML-KEM
- RSA-2048 certificate used for measurements
- Elliptic curve Diffie-Hellman Exchange (ECDHE) as baseline (**bold**)

| | Algorithm | TTFB [ms] | |
|---|---|---|---|
| | | LSQUIC | quiche |
| I | **X25519** | 3.91 | 3.57 |
| | Kyber512 | 4.08 | 3.39 |
| | BIKE-L1 | 6.59 | 5.86 |
| | HQC-128 | 5.57 | 4.21 |
| | **P-256** | 3.90 | 3.49 |
| | P-256 + Kyber512 | 4.43 | 3.74 |
| | P-256 + BIKE-L1 | 6.95 | 6.27 |
| | P-256 + HQC-128 | 5.99 | 4.52 |
| III | Kyber768 | 4.23 | 3.78 |
| | BIKE-L3 | 11.75 | 10.49 |
| | HQC-192 | 7.57 | 4.81 |
| | **P-384** | 7.36 | 6.76 |
| | P-384 + Kyber768 | 8.99 | 8.67 |
| | ... | | |

# Evaluation

## PQC – Key Encapsulation Mechanisms

- Used to transmit keys for symmetric cryptography
- Kyber, BIKE and HQC analyzed
  - Kyber selected for standardization by NIST
  - Kyber was renamed to ML-KEM
- RSA-2048 certificate used for measurements
- Elliptic curve Diffie-Hellman Exchange (ECDHE) as baseline (**bold**)

| | Algorithm | TTFB [ms] | |
|---|---|---|---|
| | | LSQUIC | quiche |
| I | **X25519** | 3.91 | 3.57 |
| | Kyber512 | 4.08 | 3.39 |
| | BIKE-L1 | 6.59 | 5.86 |
| | HQC-128 | 5.57 | 4.21 |
| | **P-256** | 3.90 | 3.49 |
| | P-256 + Kyber512 | 4.43 | 3.74 |
| | P-256 + BIKE-L1 | 6.95 | 6.27 |
| | P-256 + HQC-128 | 5.99 | 4.52 |
| III | Kyber768 | 4.23 | 3.78 |
| | BIKE-L3 | 11.75 | 10.49 |
| | HQC-192 | 7.57 | 4.81 |
| | **P-384** | 7.36 | 6.76 |
| | P-384 + Kyber768 | 8.99 | 8.67 |
| | ... | | |

## Take-away Results:

- Kyber is the fastest KEM
  - Lattice-based $\rightarrow$ small key sizes
  - Even faster than ECDHE for NIST level III and V
- Hybrid approaches only marginally slower than pure post-quantum KEMs

## PQC – Problems with QUIC

- Some algorithms (especially hash-based signature schemes) have large signature sizes
- Handshake messages might spread over multiple packets
  - Amplification attack mitigation can cause an extra RTT
  - Implementation specific attack prevention might close the connection
- Some algorithms (especially code-based key encapsulation mechanisms) are computationally expensive
  - The client might need some milliseconds to process the `ServerHello`
  - The server might retransmit the `ServerHello`, assuming packet loss

# Conclusion

## Take Away Messages

- Hardware-accelerated AES is the fastest
- QUIC's header protection is basically free, especially with AES
- Integration of post-quantum cryptography is feasible
  - no major changes required thanks to *BoringSSL* fork
- PQ has promising candidates (Kyber, Dilithium) with comparable performance to traditional algorithms
- Large certificates lead to several issues in our experiments

## Framework

- Source code publicly available

Paper:

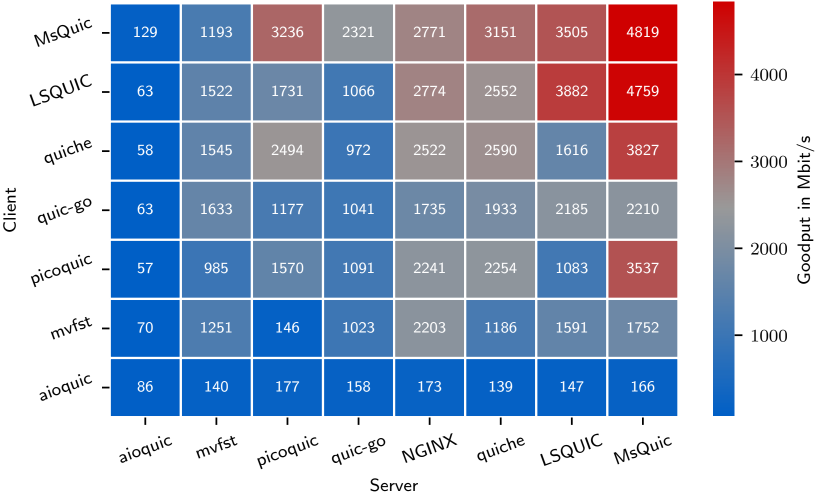
https://arxiv.org/pdf/2405.09264

Source Code:

https://github.com/tumi8/quic-crypto-paper

# Backup Slides

# QUIC Implementations - Performance Comparison

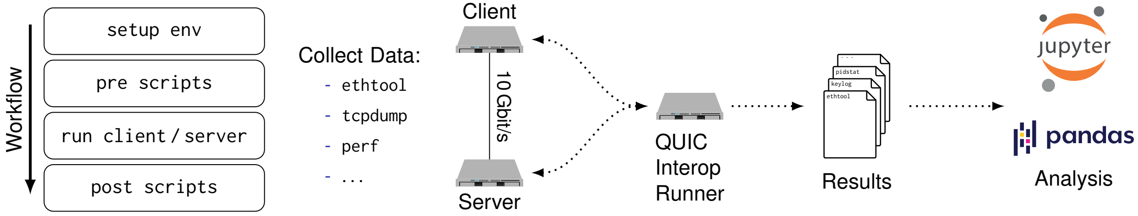


# NIST Levels

NIST's quantum security strength categories

| NIST Level | At least as hard to break as | Type of attack |
|---|---|---|
| I | AES-128 | Exhaustive key search |
| II | SHA-256 | Collision search |
| III | AES-192 | Exhaustive key search |
| IV | SHA-384 | Collision search |
| V | AES-256 | Exhaustive key search |

# Measurement and Analysis Workflow




## Measurement Workflow

1. Setup client and server host
2. Configure OS parameters
3. Start QUIC server and client
4. Reset OS parameters
5. Collect results

## Analysis Pipeline

- Tools have various output formats
- Collect and parse available result files
- Export via Pandas as CSV
- Results contain meta data like version hashes for reproducibility