Preparing for post-quantum TLS

Douglas Stebila

UNIVERSITY OF WATERLOO

https://www.douglas.stebila.ca/research/presentations/

NSERC CRSNG

European Space Agency

esa.int

→ THE EUROPEAN SPACE AGENCY

esa

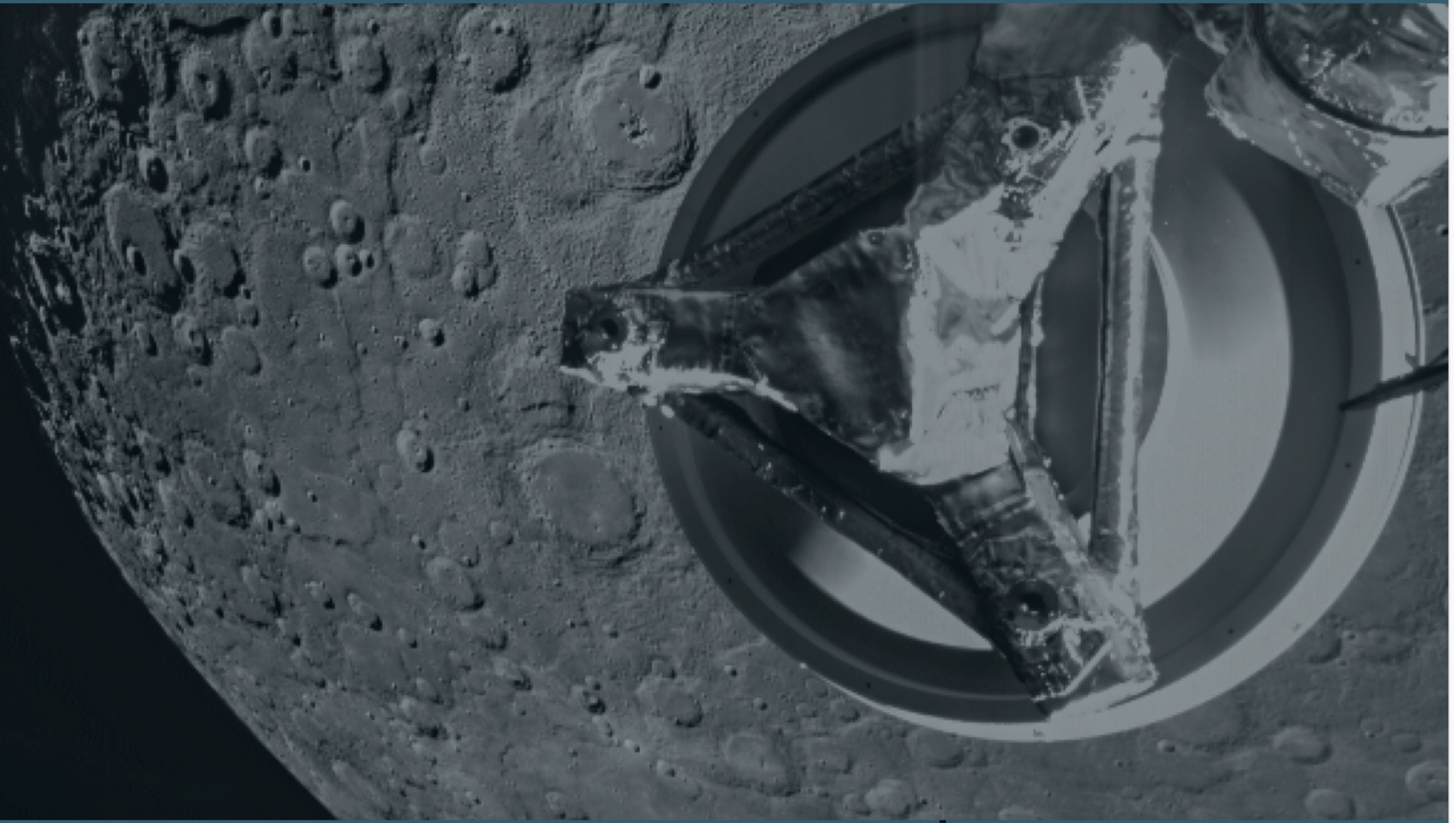
STORY

SCIENCE & EXPLORATION

Second helpings of Mercury

24/06/2022 20307 VIEWS 44 LIKES

READ →



ESA Vision

ESA Pillars

ESA Programmes

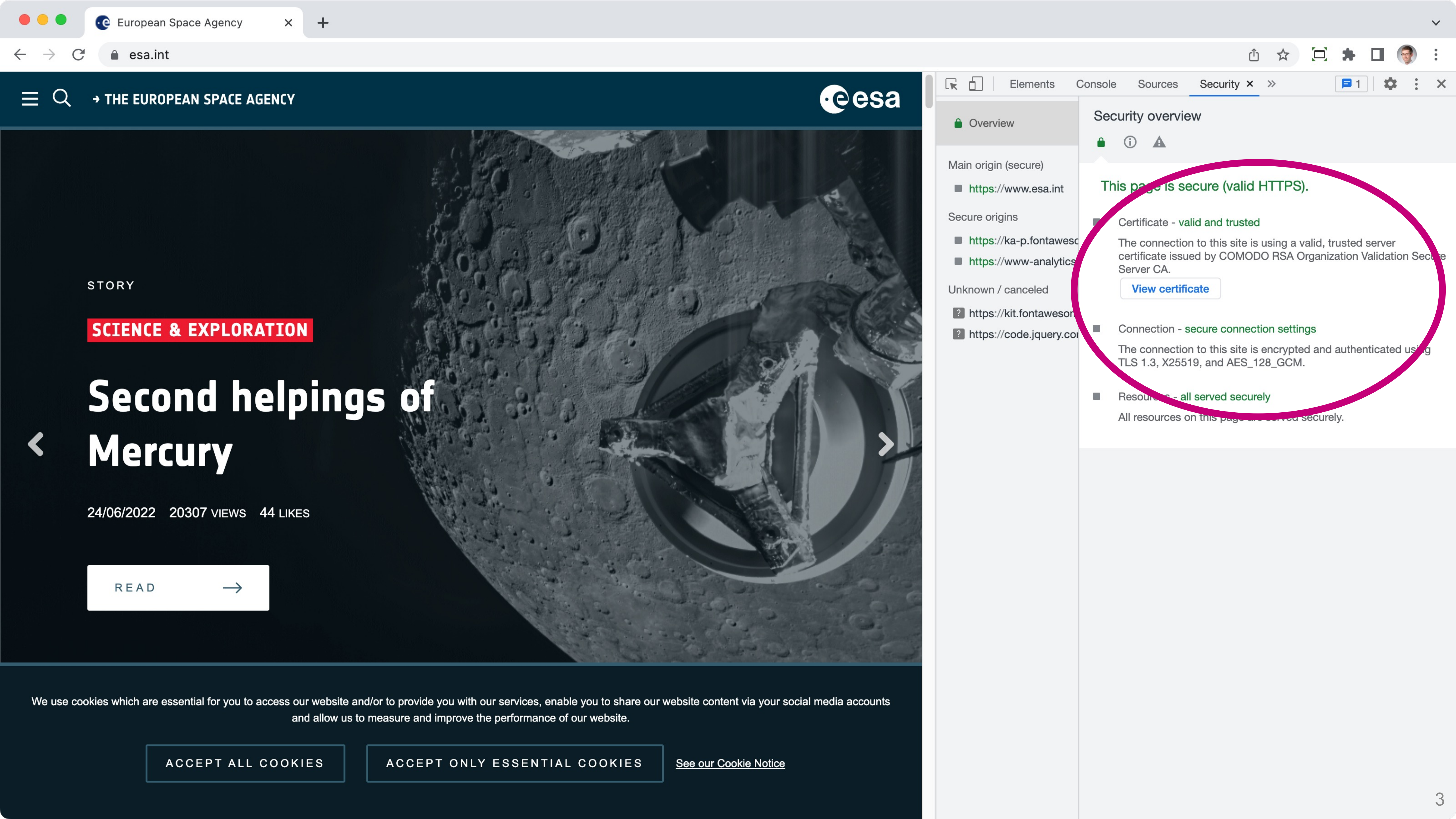
ESA and You

We use cookies which are essential for you to access our website and/or to provide you with our services, enable you to share our website content via your social media accounts and allow us to measure and improve the performance of our website.

ACCEPT ALL COOKIES

ACCEPT ONLY ESSENTIAL COOKIES

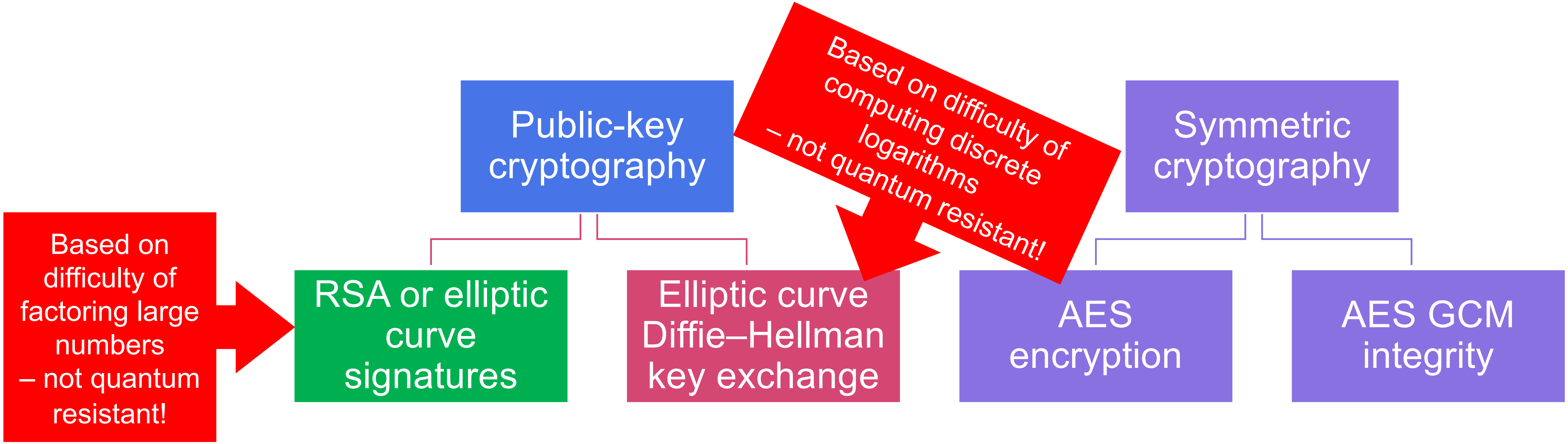
See our Cookie Notice


European Space Agency
esa.int
THE EUROPEAN SPACE AGENCY
esa
STORY
SCIENCE & EXPLORATION
Second helpings of Mercury
24/06/2022 20307 VIEWS 44 LIKES
READ
We use cookies which are essential for you to access our website and/or to provide you with our services, enable you to share our website content via your social media accounts and allow us to measure and improve the performance of our website.
ACCEPT ALL COOKIES
ACCEPT ONLY ESSENTIAL COOKIES
See our Cookie Notice
Elements
Console
Sources
Security
Overview
Main origin (secure)
https://www.esa.int
Secure origins
https://ka-p.fontawesc
https://www-analytics
Unknown / canceled
https://kit.fontawesor
https://code.jquery.cor
Security overview
This page is secure (valid HTTPS).
Certificate - valid and trusted
The connection to this site is using a valid, trusted server certificate issued by COMODO RSA Organization Validation Secure Server CA.
View certificate
Connection - secure connection settings
The connection to this site is encrypted and authenticated using TLS 1.3, X25519, and AES_128_GCM.
Resources - all served securely
All resources on this page are served securely.

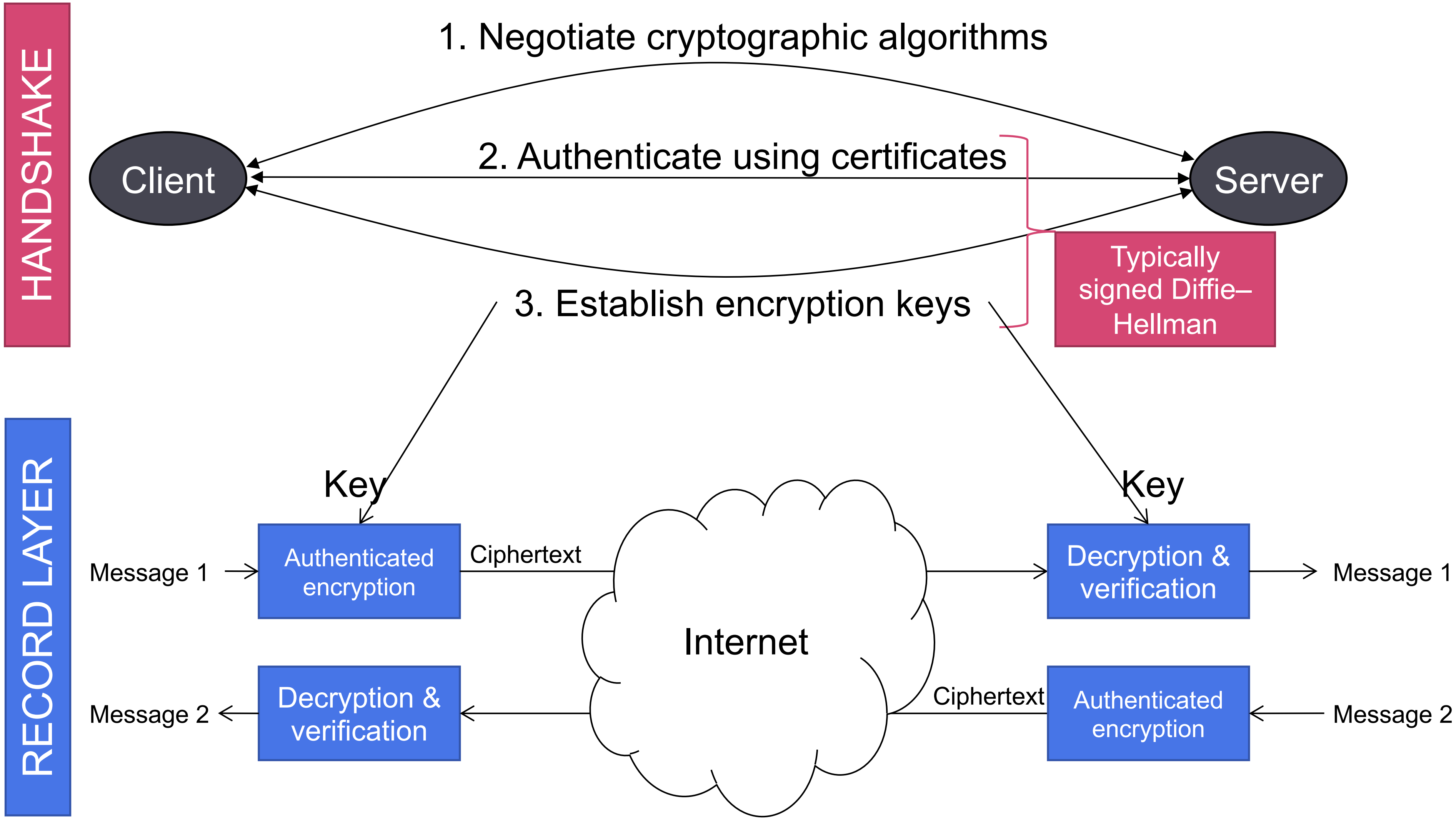
Cryptographic building blocks

Connection - secure connection settings

The connection to this site is encrypted and authenticated using TLS 1.3, X25519, and AES_256_GCM.



SSL/TLS Protocol



Four TLS 1.3 modes

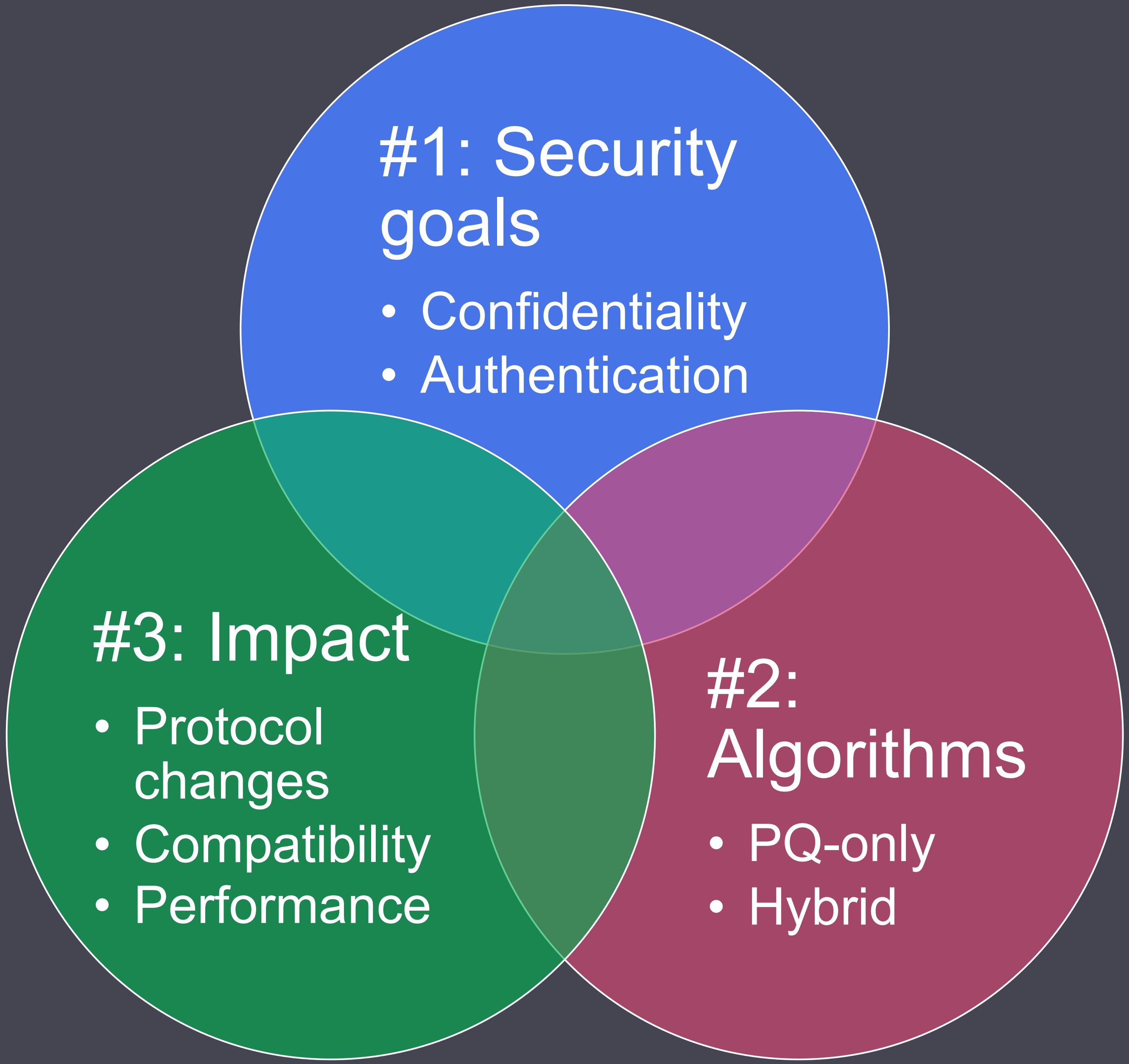
Signed Diffie–Hellman, server-only authentication

Signed Diffie–Hellman, mutual authentication

Pre-shared key (PSK) ← **Already PQ!**

Pre-shared key with ephemeral Diffie–Hellman (PSK-ECDHE)

Three dimensions of "post-quantum TLS"



What is "post-quantum TLS"?

Pre-shared key (PSK) mode	Post-quantum key exchange	Classical+PQ key exchange	Post-quantum signatures	Classical+PQ signatures	Alternative protocol designs
• Already supported! • Still has the key distribution problem • No PQ forward secrecy	• Easiest to implement • Easy backwards compatibility • Needed soonest: harvest now & decrypt later with quantum computer	• "Hybrid" • Easy to implement • Possibly in demand during pre-FIPS-certification period	• On the web: requires coordination with certificate authorities • Less urgently needed: can't retroactively break channel authentication	• "Hybrid" or "Composite" • May not make sense in the context of a negotiated protocol like TLS	• Harder to implement; may require state machine or architecture changes

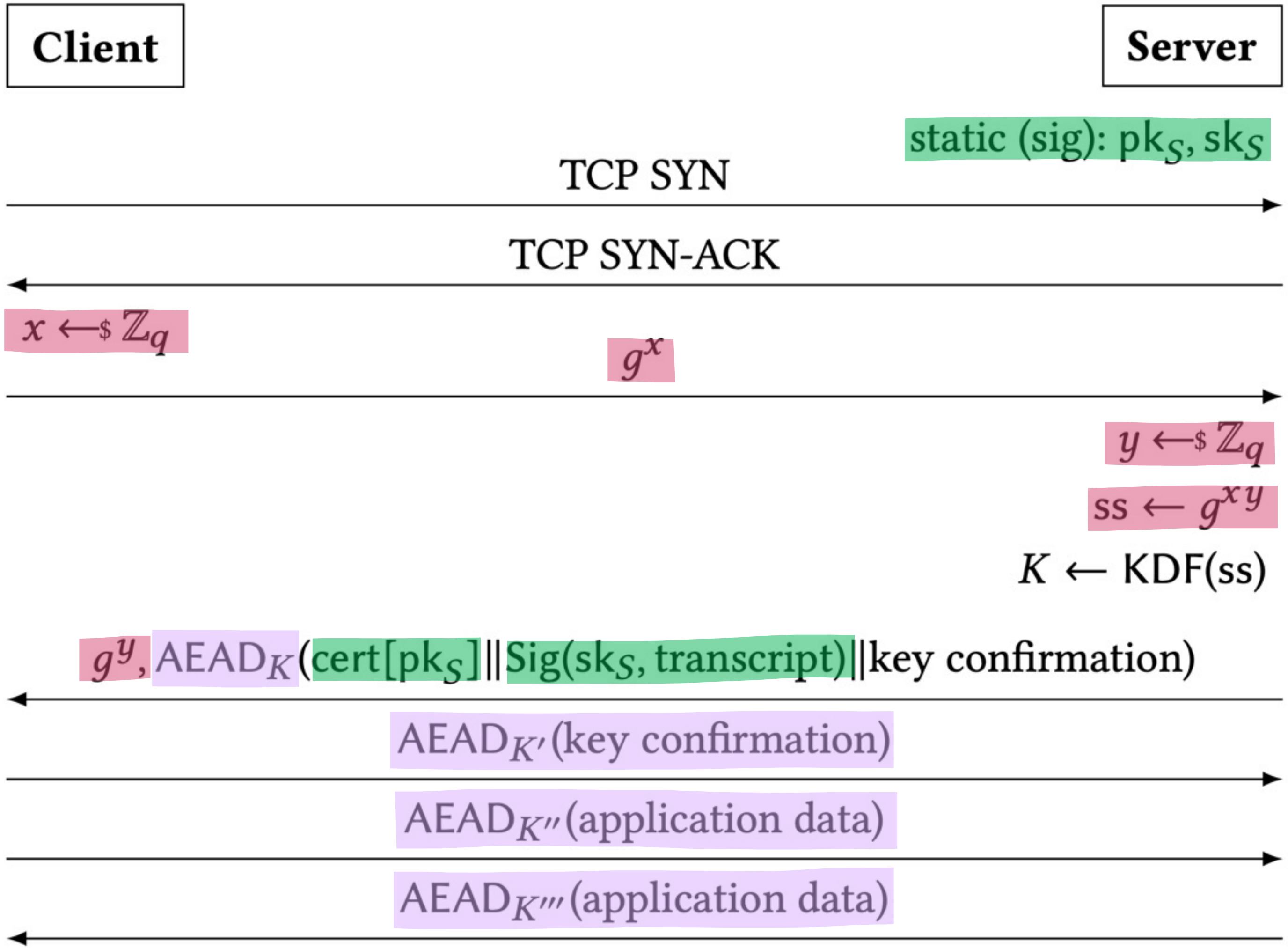
TLS 1.3 handshake

Diffie-Hellman key exchange

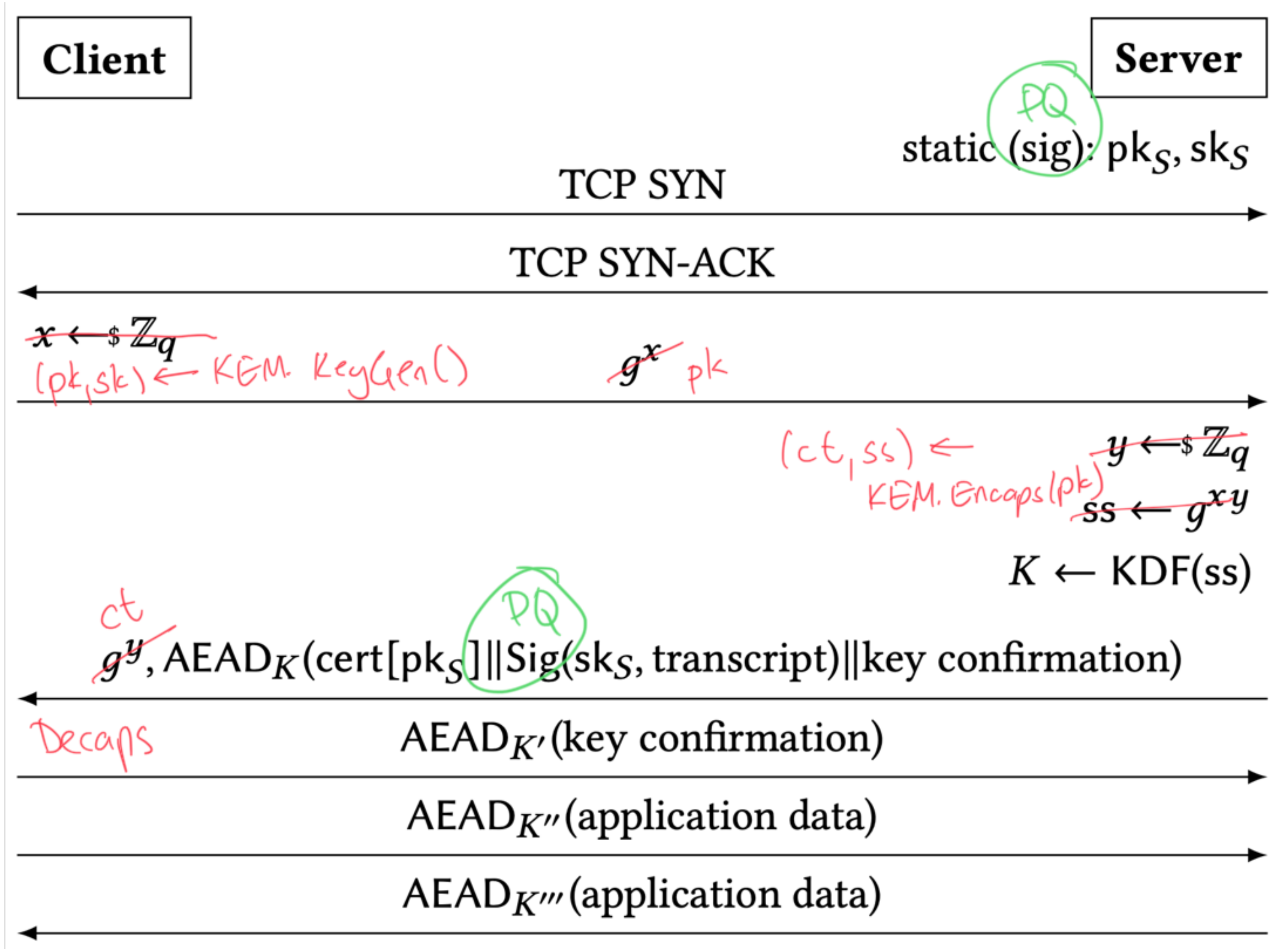
Digital signature

Signed Diffie–Hellman

Authenticated encryption

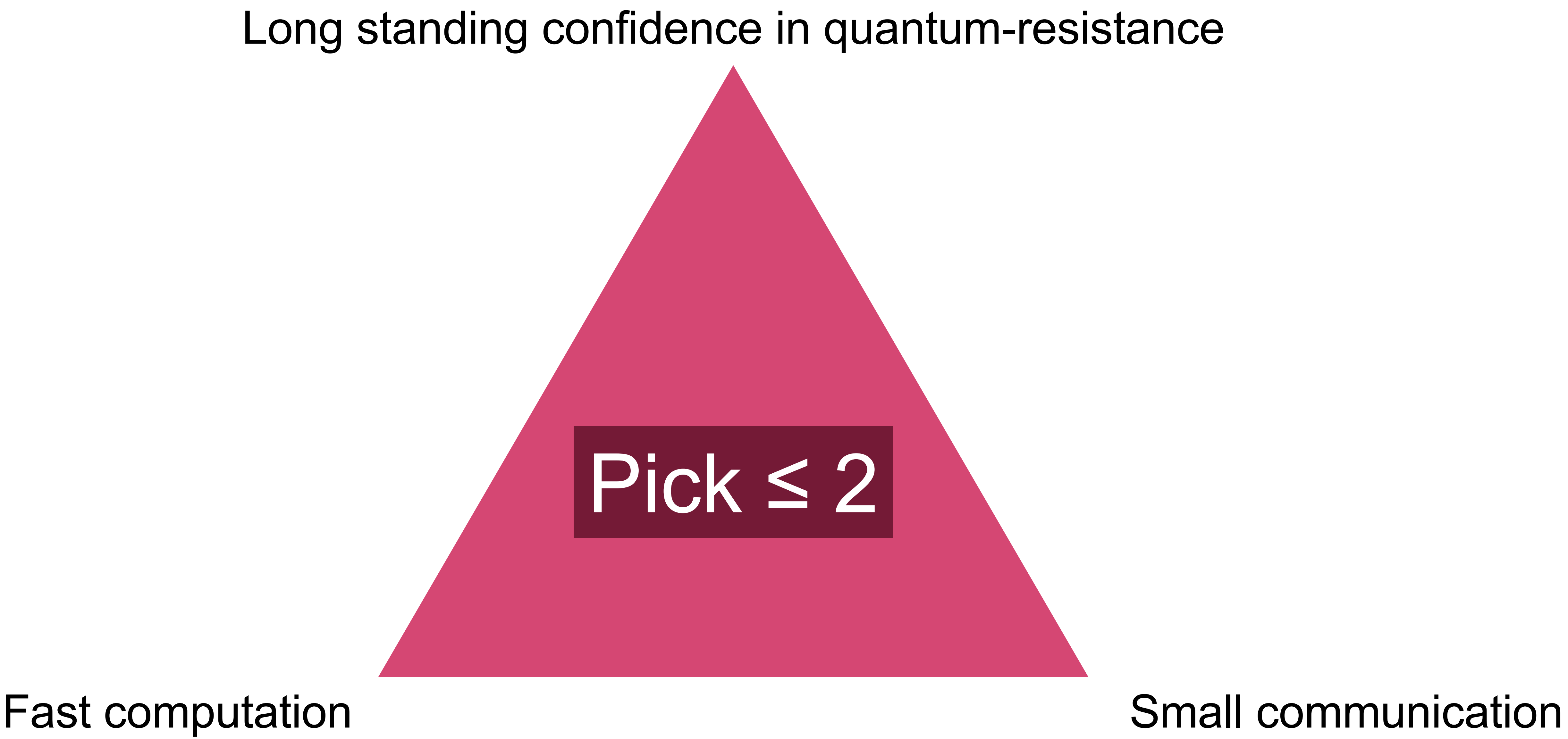


TLS 1.3 handshake



~~Signed Diffie–Hellman~~

Post-Quantum!!!


Long standing confidence in quantum-resistance
Pick ≤ 2
Fast computation
Small communication

Outline

Classical + PQ key exchange

Classical + PQ signatures

Performance

Classical + PQ key exchange

Douglas Stebila, Scott Fluhrer, Shay Gueron

https://datatracker.ietf.org/doc/html/draft-ietf-tls-hybrid-design-03

Why use two (or more) algorithms?

1. Reduce risk from break of one algorithm

2. Ease transition with improved backwards compatibility

3. Standards compliance during transition

Why use two (or more) algorithms?

1. Reduce risk from break of one algorithm

- Enable early adopters to get post-quantum security without abandoning security of existing algorithms
- Retain security as long as at least one algorithm is not broken
- Uncertainty re: long-term security of existing cryptographic assumptions
- Uncertainty re: newer cryptographic assumptions

2. Ease transition with improved backwards compatibility

3. Standards compliance during transition

Why use two (or more) algorithms?

1. Reduce risk from break of one algorithm

2. Ease transition with improved backwards compatibility

- Design backwards-compatible data structures with old algorithms that can be recognized by systems that haven't been upgraded, but new implementations will use new algorithms
- May not be necessary for negotiated protocols like TLS

3. Standards compliance during transition

Why use two (or more) algorithms?

1. Reduce risk from break of one algorithm

2. Ease transition with improved backwards compatibility

3. Standards compliance during transition

- Early adopters may want to use post-quantum before standards-compliant (FIPS-)certified implementations are available
- Possible to combine (in a certified way) keying material from FIPS-certified (non-PQ) implementation with non-certified keying material

Terminology

- “Hybrid”
- “Composite”
- “Dual algorithms”
- “Robust combiner” [HKNRR05]

[HKNRR05] Harnik, Kilian, Naor, Reingold, Rosen. Eurocrypt 2005.

IETF draft: Hybrid key exchange in TLS 1.3

Goals

Define data structures for negotiation, communication, and shared secret calculation for hybrid key exchange

Non-goals

- Hybrid/composite certificates or digital signatures
- Selecting which post-quantum algorithms to use in TLS

Mechanism

Main idea:
Each desired combination of traditional + post-quantum algorithm will be a new (opaque) key exchange “group”

- **Negotiation**: new named groups for each desired combination will need to be standardized
- **Key shares**: concatenate key shares for each constituent algorithm
- **Shared secret calculation**: concatenate shared secrets for each constituent algorithm and use as input to key schedule

IETF draft: Hybrid key exchange in TLS 1.3

Current status

- May 2022: Working group last call
- In progress: Minor revisions from WGLC
- Then: Park until NIST Round 3 concludes and CFRG has reviewed algorithms

Classical + PQ signatures

LAMPS working group

- “Limited Additional Mechanisms for PKIX and S/MIME”
  - PKIX: Public key infrastructure a.k.a. X.509 certificates
  - S/MIME: Secure email (encrypted/signed)

- LAMPS charter now includes milestones related to PQ
  - PQ algorithms in PKIX/X.509 and S/MIME
  - Hybrid key establishment
  - Dual signatures

https://datatracker.ietf.org/wg/lamps/about/

IETF drafts: pq-composite-keys, -sigs

Led by Mike Ounsworth from Entrust
and Massimiliano Pala from CableLabs

(I’m not involved – just including here FYI)

https://datatracker.ietf.org/doc/html/draft-ounsworth-pq-composite-keys-02
https://datatracker.ietf.org/doc/html/draft-ounsworth-pq-composite-sigs-07

IETF drafts: pq-composite-keys, -sigs

Main question

How to represent algorithm identifiers and keys

Option #1: Generic composite

Single algorithm id representing “composite”, then an additional field containing list of algorithms

- Good for prototyping
- Allow for high degree of agility
- Allows ≥ 2 algorithms

Option #2: Explicit composite

New algorithm id for each combination of algorithms

- Less new processing logic
- Lower degree of agility

https://datatracker.ietf.org/doc/html/draft-ounsworth-pq-composite-keys-02
https://datatracker.ietf.org/doc/html/draft-ounsworth-pq-composite-sigs-07

Composite AND versus Composite OR

In an asynchronous setting:

How is a credential with two public keys/signatures meant to be used?

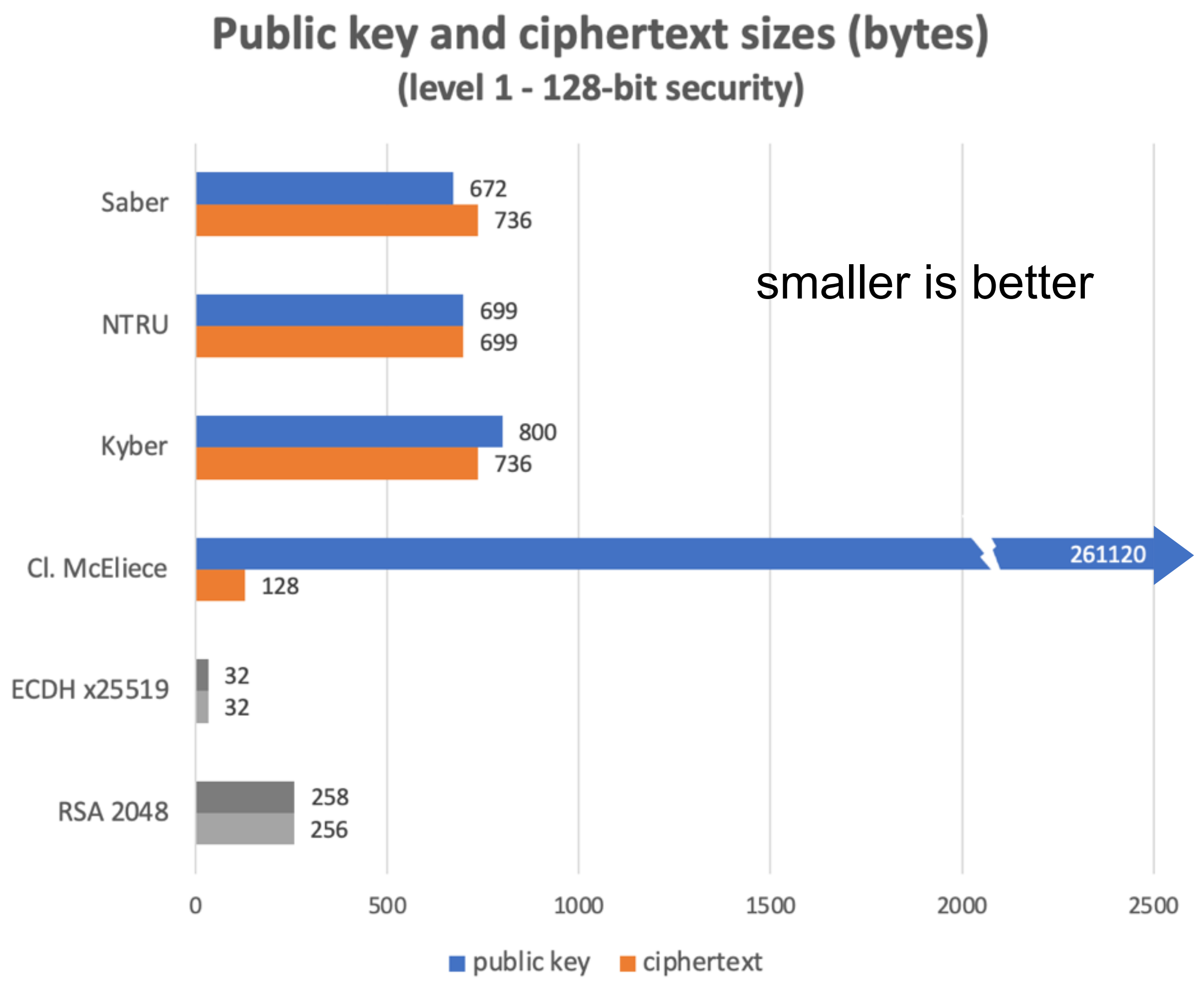
- Must both algorithms be used? (Composite AND)
- Is either algorithm okay? (Composite OR)
    - Must take countermeasures to avoid stripping/separating context
    - Risks of ambiguity

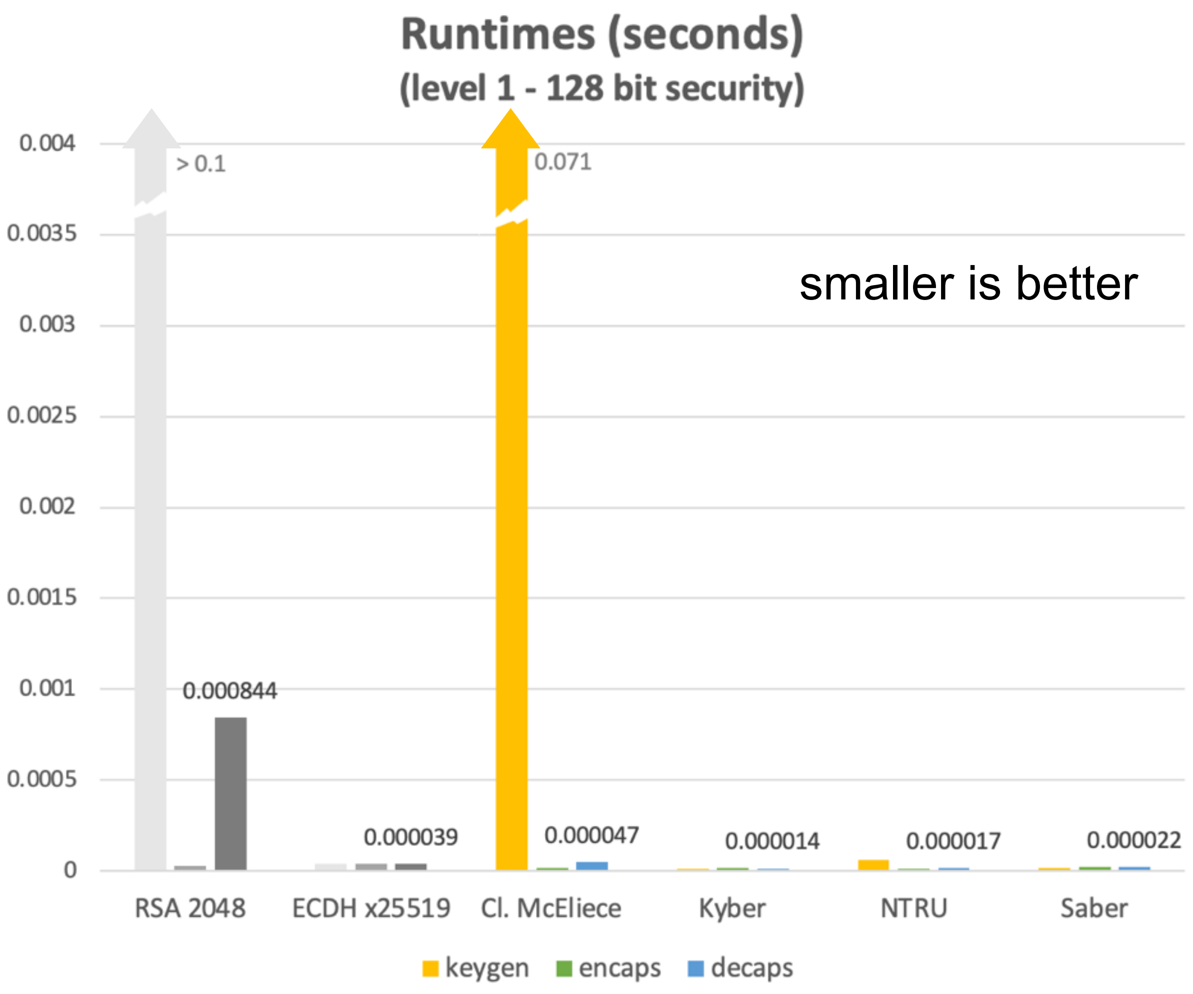
TLS performance

Open Quantum Safe benchmarking. https://openquantumsafe.org/benchmarking/

Christian Paquin, Douglas Stebila, Goutam Tamvada.
PQCrypto 2020. https://eprint.iacr.org/2019/1447

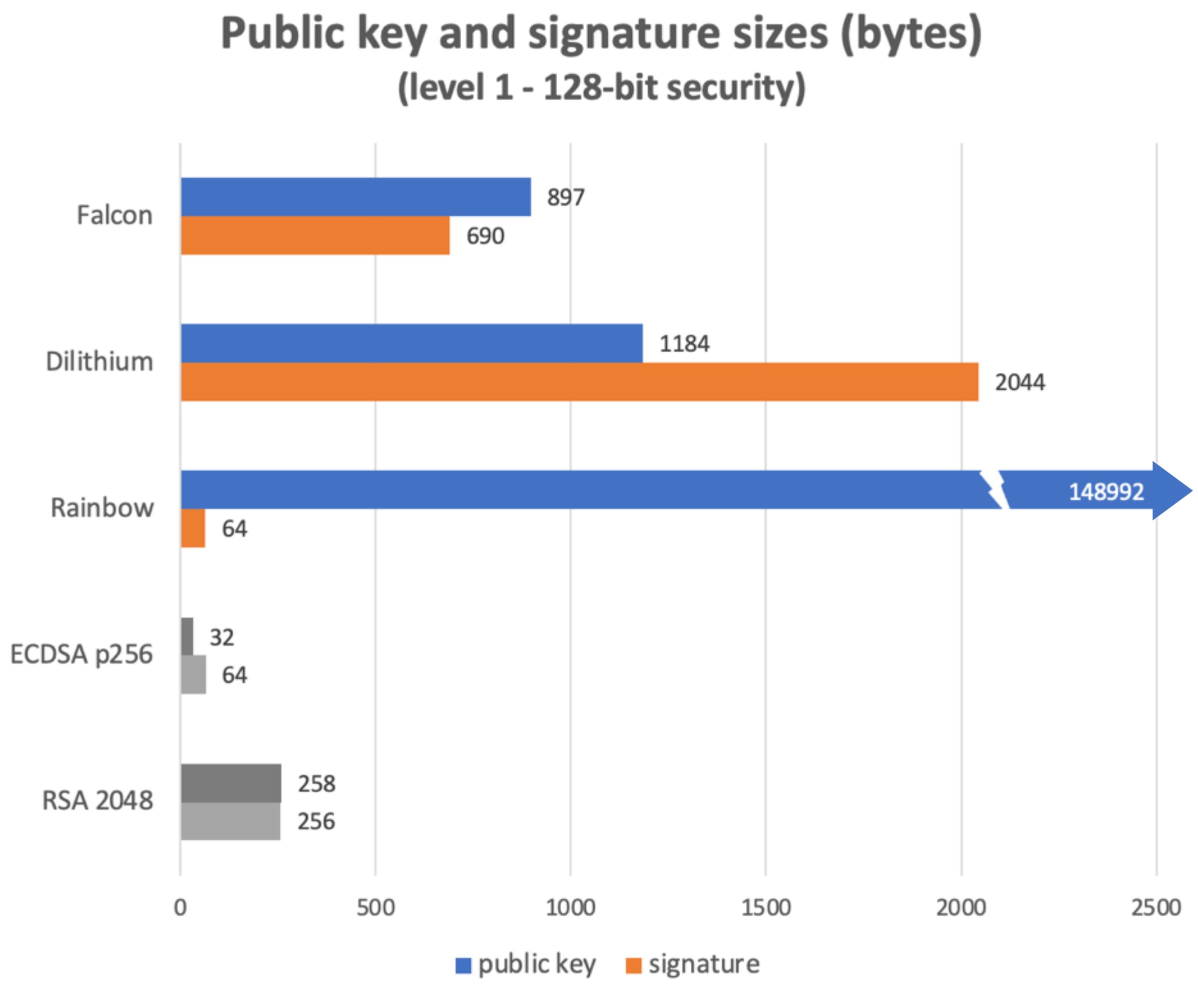
Base performance – Round 3 KEM Finalists

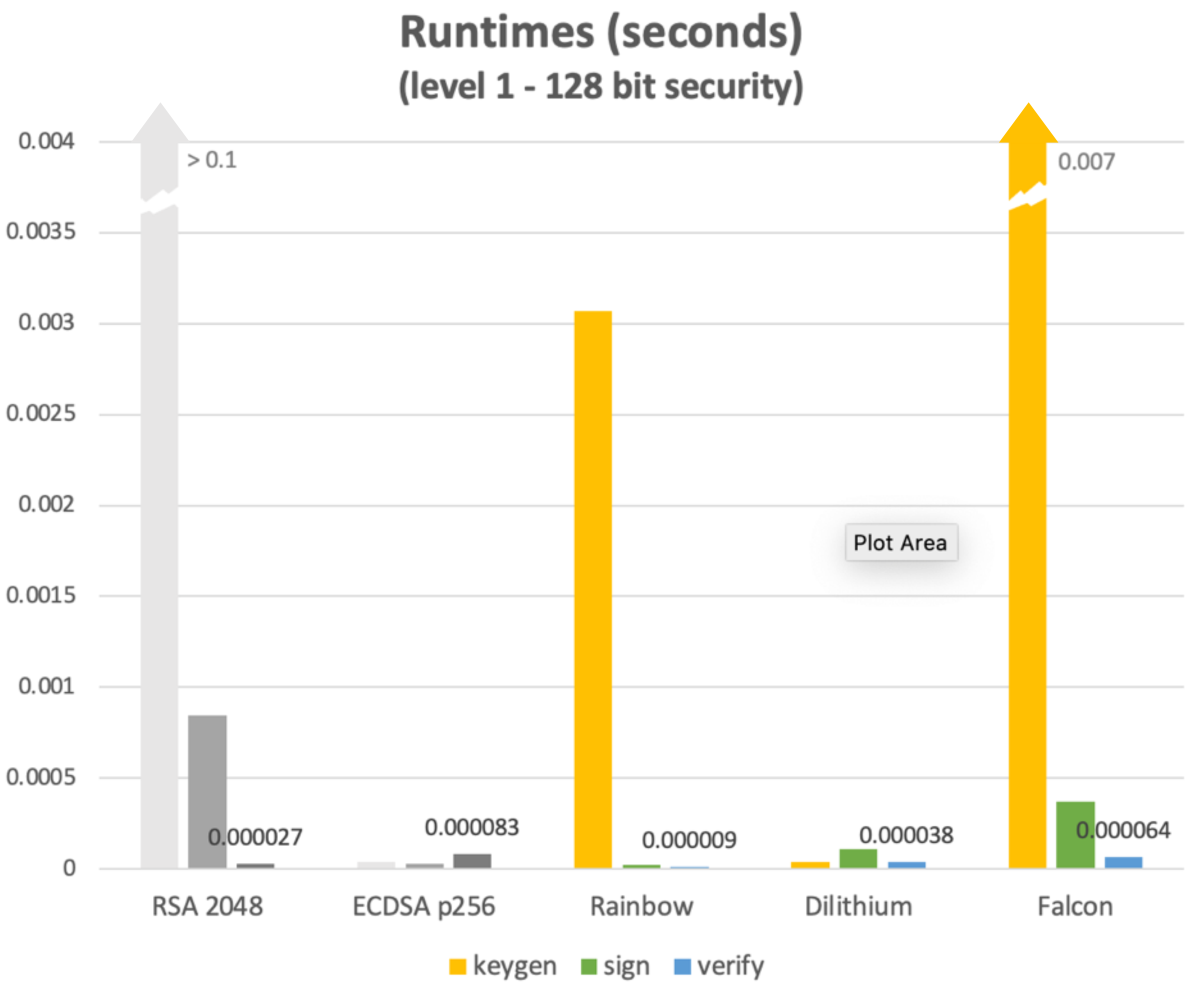




Based on Round 2 submission documents; AVX2 runtimes normalized

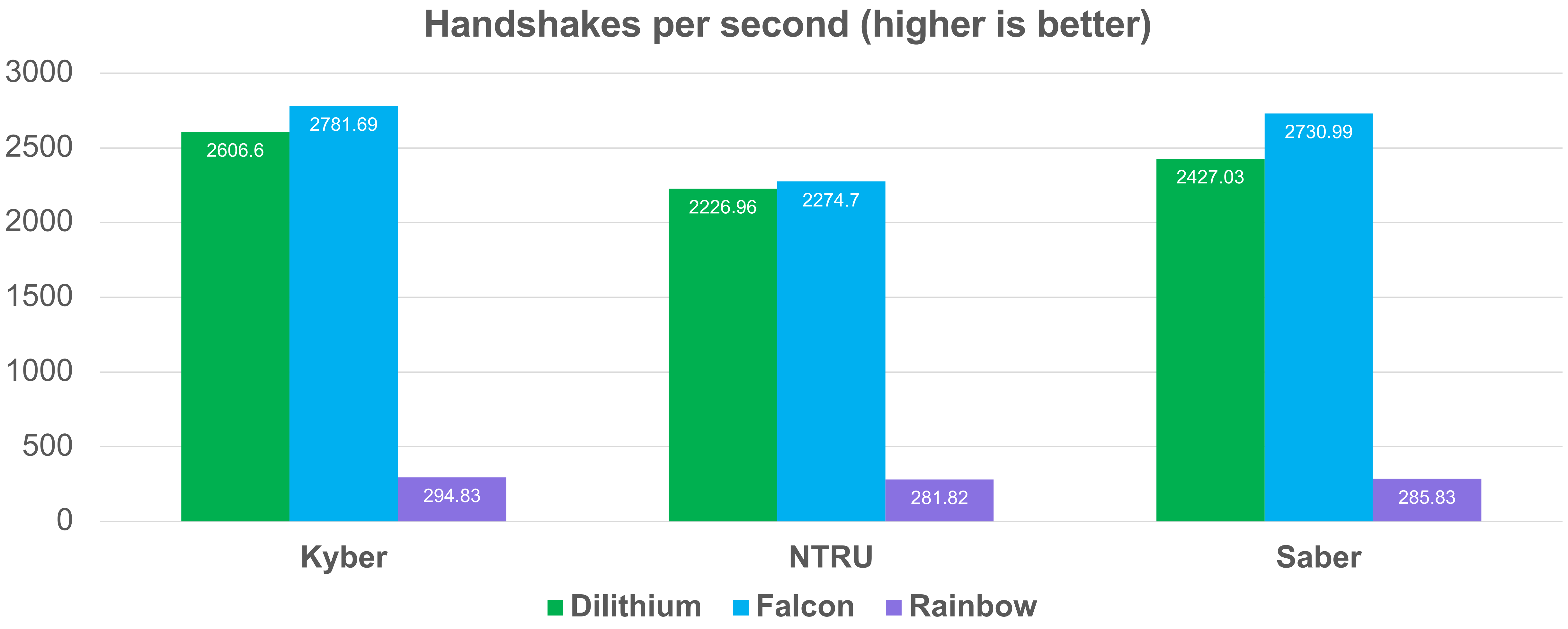
Base performance – Round 3 Signature Finalists





Based on Round 2 submission documents; AVX2 runtimes normalized

TLS performance – ideal conditions

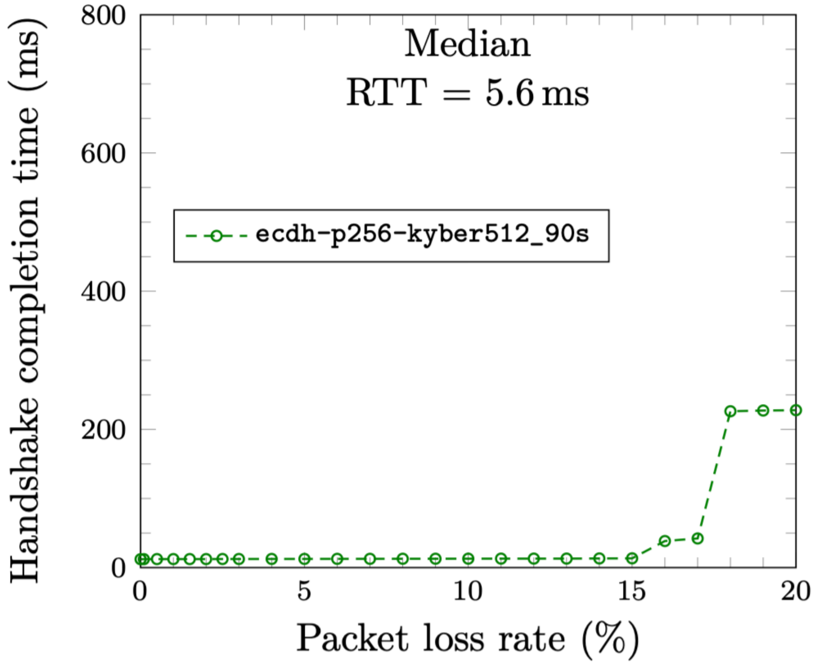


OQS benchmarking 2022/06/25 – x86_64 “performance” build – https://openquantumsafe.org/benchmarking/

TLS performance

Higher latency & packet loss

50th percentile

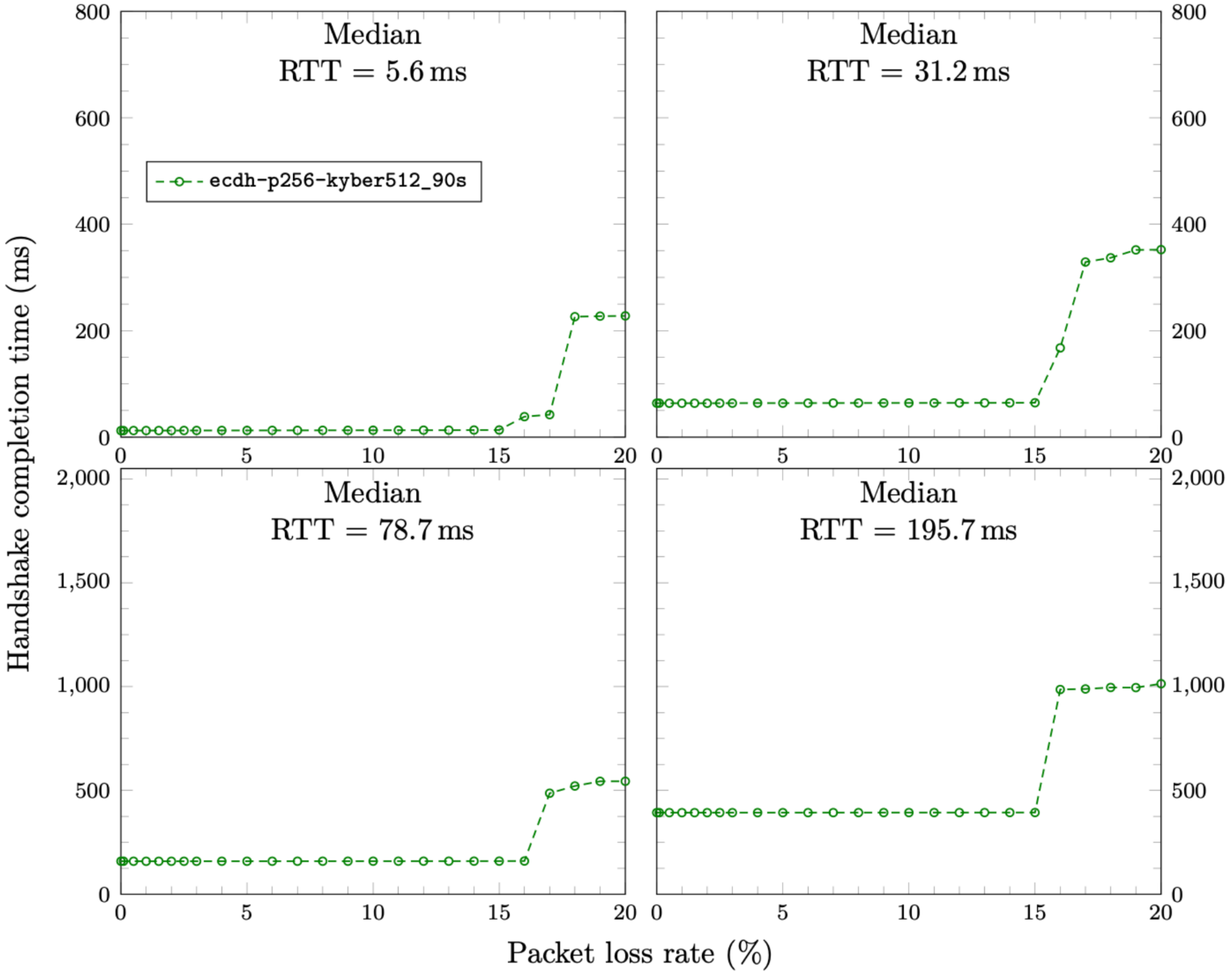


OQS-OpenSSL 1.1.1, x86_64, AVX2 enabled – https://eprint.iacr.org/2019/1447

TLS performance

Higher latency & packet loss

50th percentile

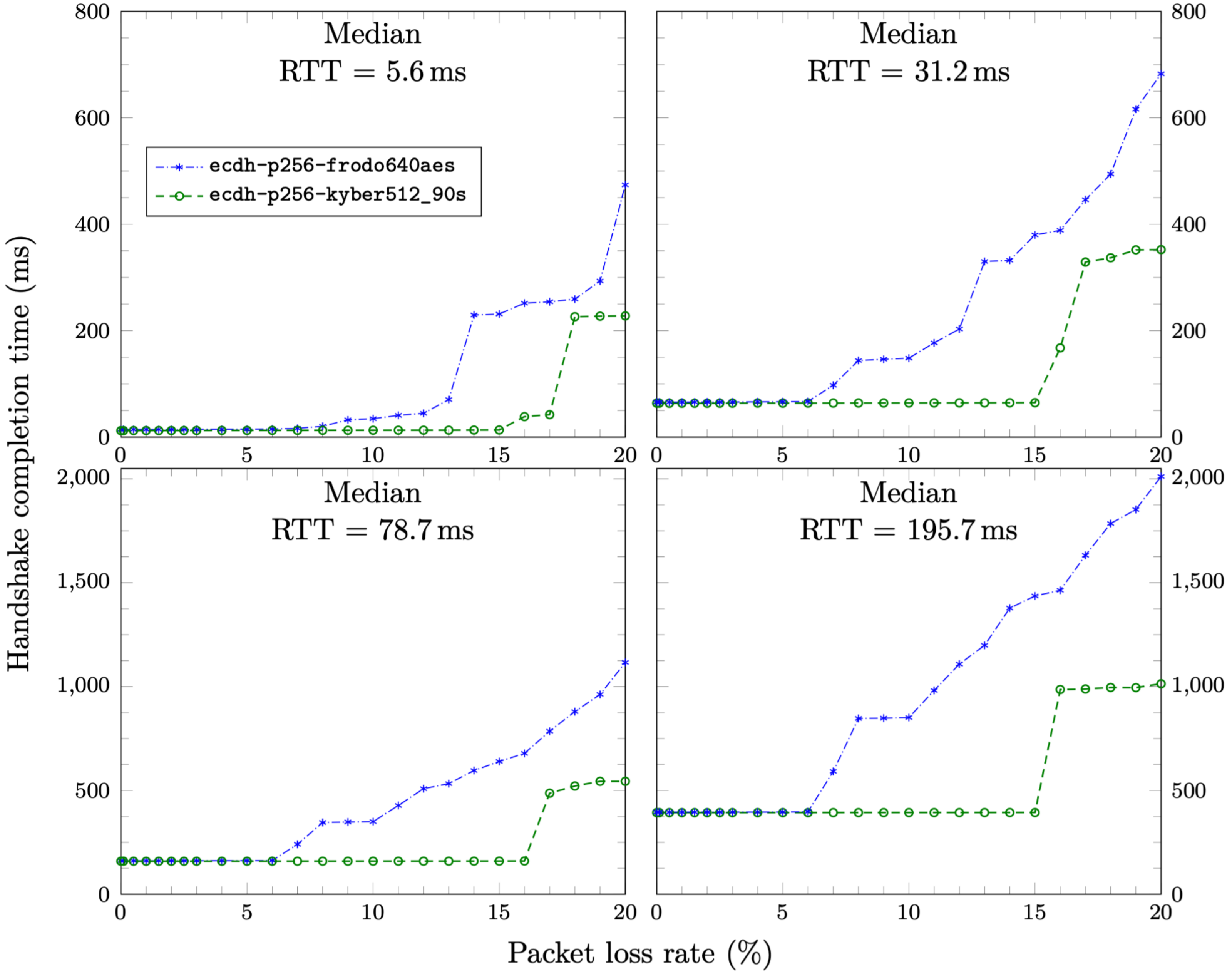


OQS-OpenSSL 1.1.1, x86_64, AVX2 enabled – https://eprint.iacr.org/2019/1447

TLS performance

Higher latency & packet loss

50th percentile

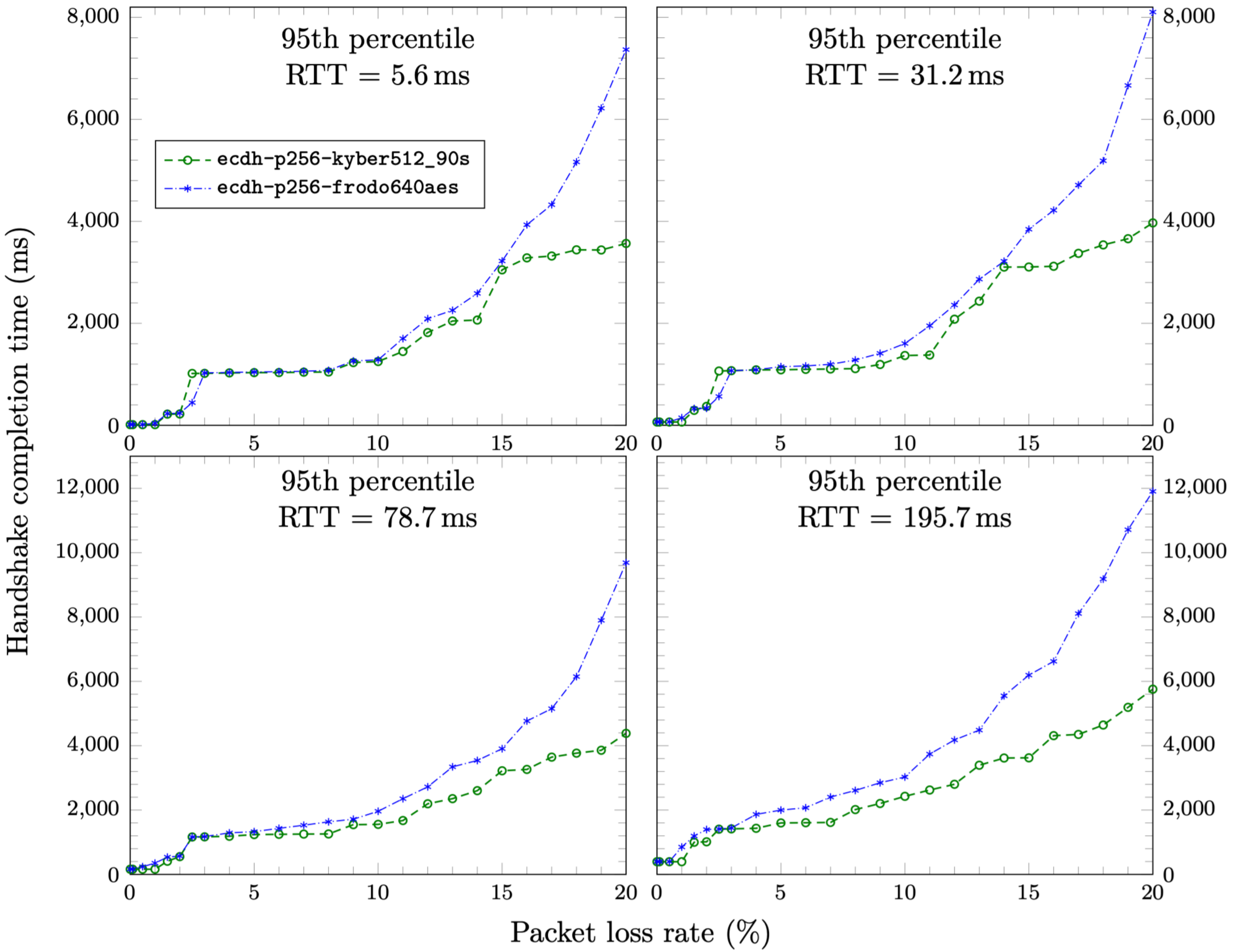


OQS-OpenSSL 1.1.1, x86_64, AVX2 enabled – https://eprint.iacr.org/2019/1447

TLS performance

Higher latency & packet loss

95th percentile



OQS-OpenSSL 1.1.1, x86_64, AVX2 enabled – https://eprint.iacr.org/2019/1447

TLS performance

On **fast, reliable network links**, the cost of public key cryptography dominates the median TLS establishment time, but does not substantially affect the 95th percentile establishment time

On **unreliable network links** (packet loss rates ≥ 3%), communication sizes come to govern handshake completion time

As application data sizes grow, the relative cost of TLS handshake establishment diminishes compared to application data transmission

OPEN QUANTUM SAFE

software for prototyping quantum-resistant cryptography

https://openquantumsafe.org https://github.com/open-quantum-safe

liboqs

- C library with common API for post-quantum signature schemes and key encapsulation mechanisms
- MIT License
- Builds on Windows, macOS, Linux; x86_64, ARM v8
- Includes all Round 3 finalists and alternate candidates
  - (except GeMSS)

https://openquantumsafe.org/liboqs/

TLS 1.3 implementations

	OQS-OpenSSL 1.1.1	OQS-OpenSSL 3 provider	OQS-BoringSSL
PQ key exchange in TLS 1.3	✓	✓	✓
Classical + PQ key exchange in TLS 1.3	✓	✓	✓
PQ certificates and signature authentication in TLS 1.3	✓	×	✓
Classical + PQ certificates and signature authentication in TLS 1.3	✓	×	×

Using draft-ietf-tls-hybrid-design for hybrid key exchange

Interoperability test server running at https://test.openquantumsafe.org

https://openquantumsafe.org/applications/tls/

Applications

- Demonstrator application integrations into:
  - Apache
  - nginx
  - haproxy
  - curl
  - Chromium
  - Wireshark

- In most cases required few/no modifications to work with updated OpenSSL

- Runnable Docker images available for download

https://openquantumsafe.org/applications/tls/#demo-integrations

Preparing for post-quantum TLS

Douglas Stebila
UNIVERSITY OF WATERLOO

https://www.douglas.stebila.ca/research/presentations/

What is post-quantum TLS?

- PSK mode
- PQ key exchange
- Classical + PQ key exchange
- PQ signatures
- Classical + PQ signatures
- Alternative protocol designs (KEMTLS)

Hybrid key exchange in TLS 1.3

https://datatracker.ietf.org/doc/html/draft-ietf-tls-hybrid-design-03

Composite certificates

https://datatracker.ietf.org/doc/html/draft-ounsworth-pq-composite-keys-02
https://datatracker.ietf.org/doc/html/draft-ounsworth-pq-composite-sigs-07

Performance

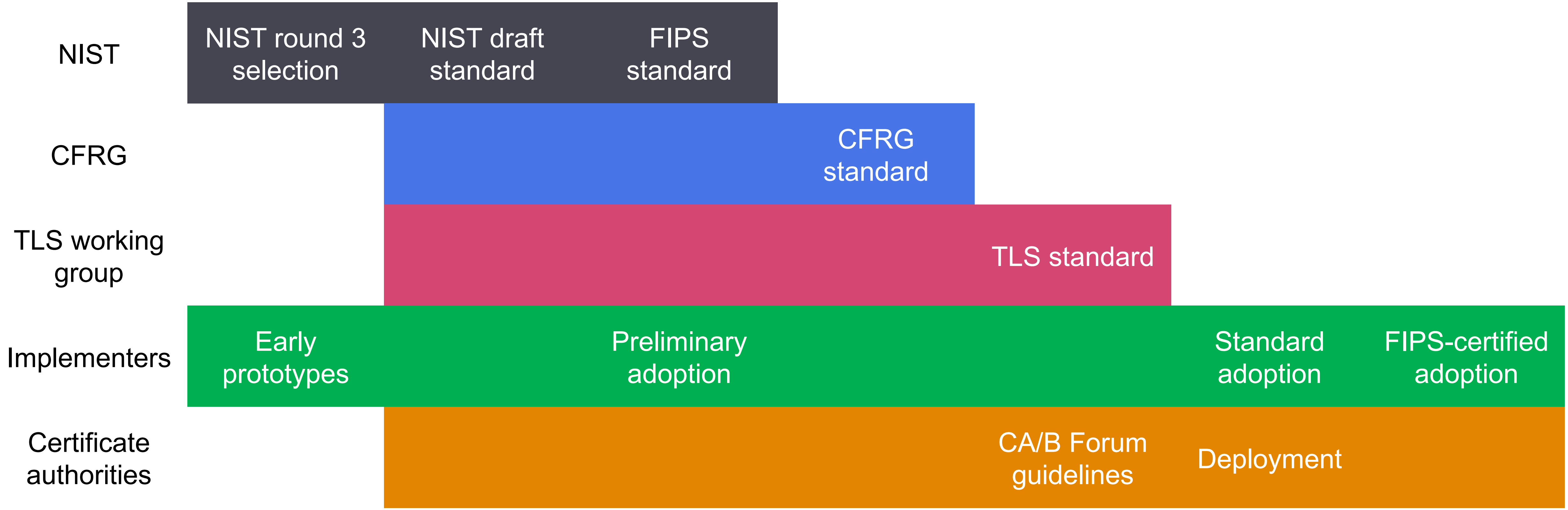
https://eprint.iacr.org/2019/1447
https://openquantumsafe.org/benchmarking/

Open Quantum Safe project

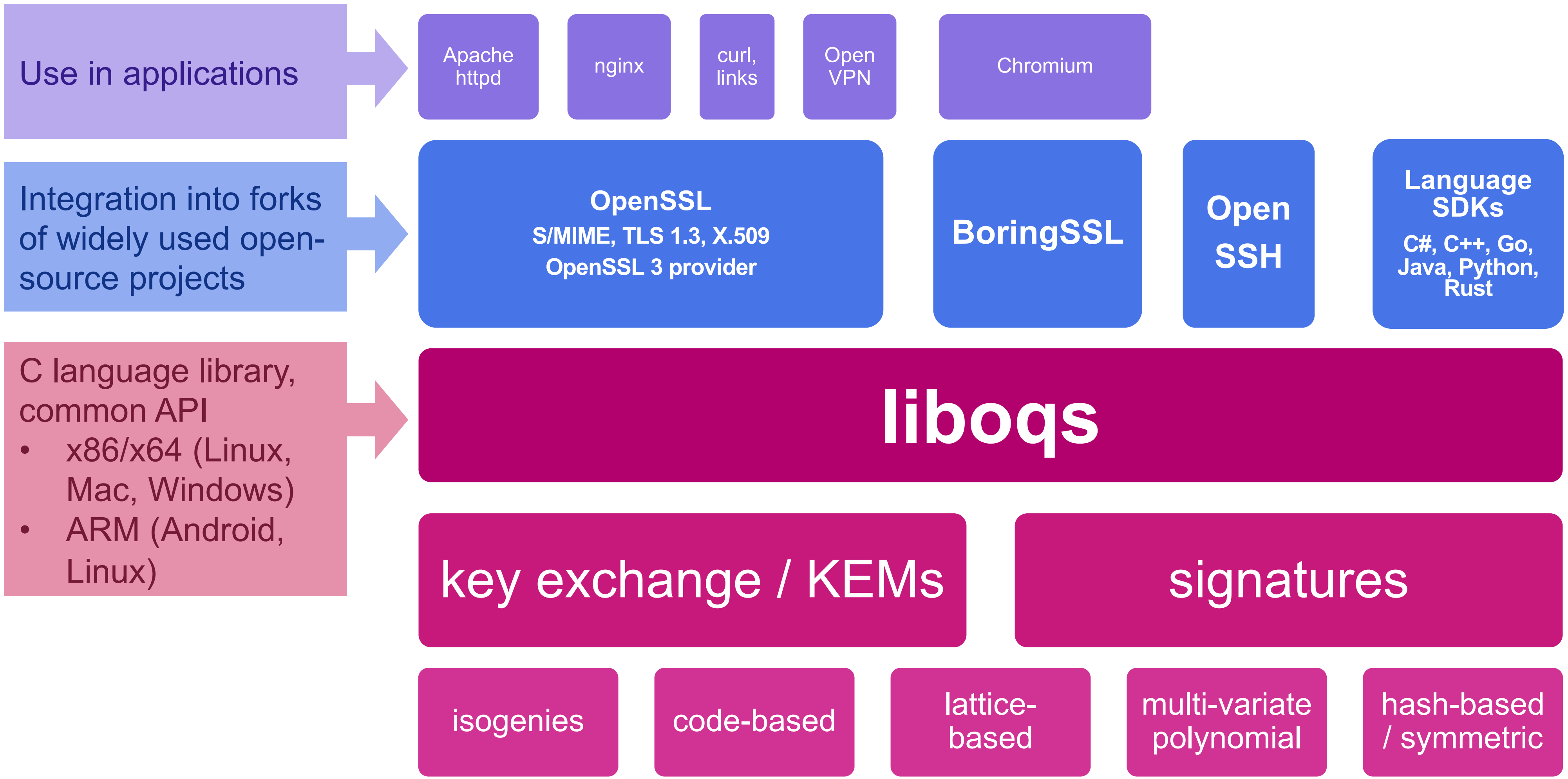
https://openquantumsafe.org • https://github.com/open-quantum-safe/

Appendix

Paths to standardization and adoption



Open Quantum Safe Project



Industry partners:

- Amazon Web Services
- evolutionQ
- IBM Research
- Microsoft Research

Additional contributors:

- Cisco
- Senetas
- PQClean project
- Individuals

Financial support:

- AWS
- Canadian Centre for Cyber Security
- NSERC
- Unitary Fund

https://openquantumsafe.org/ • https://github.com/open-quantum-safe/

Appendix:
Classical + PQ key exchange

Other design options

Negotiation

- 2 vs ≥2 algorithms
- More flexibility / granularity in algorithm selection
  - Extension for representing algorithm options and constraints

Key shares

- Separately list key shares for each algorithm
- Use extensions for extra key shares
- => More efficient communication

Shared secret calculation

- Apply KDF before inserting into key schedule
- XOR shares
- Insert into different parts of TLS key schedule

Securely combining keying material

Is it okay to use concatenation?

$ss = k_1 \| k_2$

$ss = H(k_1 \| k_2)$

Note concatenation is the primary hybrid method approved by NIST.

- Assume at least one of k_1 or k_2 is indistinguishable from random.
- If H is a random oracle, then ss is indistinguishable from random.
- If k_1 and k_2 are fixed length and H is a dual PRF in either half of its input, then ss is indistinguishable from random.

https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Cr2.pdf#page=10