Google

# Hybrid Post-Quantum Signatures in Hardware Security Keys


**Fabian Kaczmarczyck**

kaczmarczyck@google.com

Diana Ghinea, Fabian Kaczmarczyck, Jennifer Pullman, Julien Cretin, Stefan Kölbl, Rafael Misoczki, Jean-Michel Picod, Luca Invernizzi, Elie Bursztein


Google

Security and Privacy Research

# Team and Contributors

Diana Ghinea

Fabian Kaczmarczyck

Jennifer Pullman

Julien Cretin

Stefan Kölbl

Rafael Misoczki

Jean-Michel Picod

Luca Invernizzi

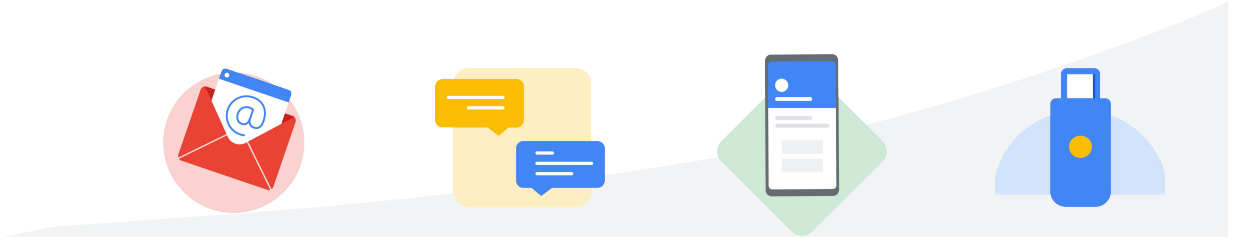
Elie Bursztein

# Security keys:
# The most secure
# two-factor authentication


Google

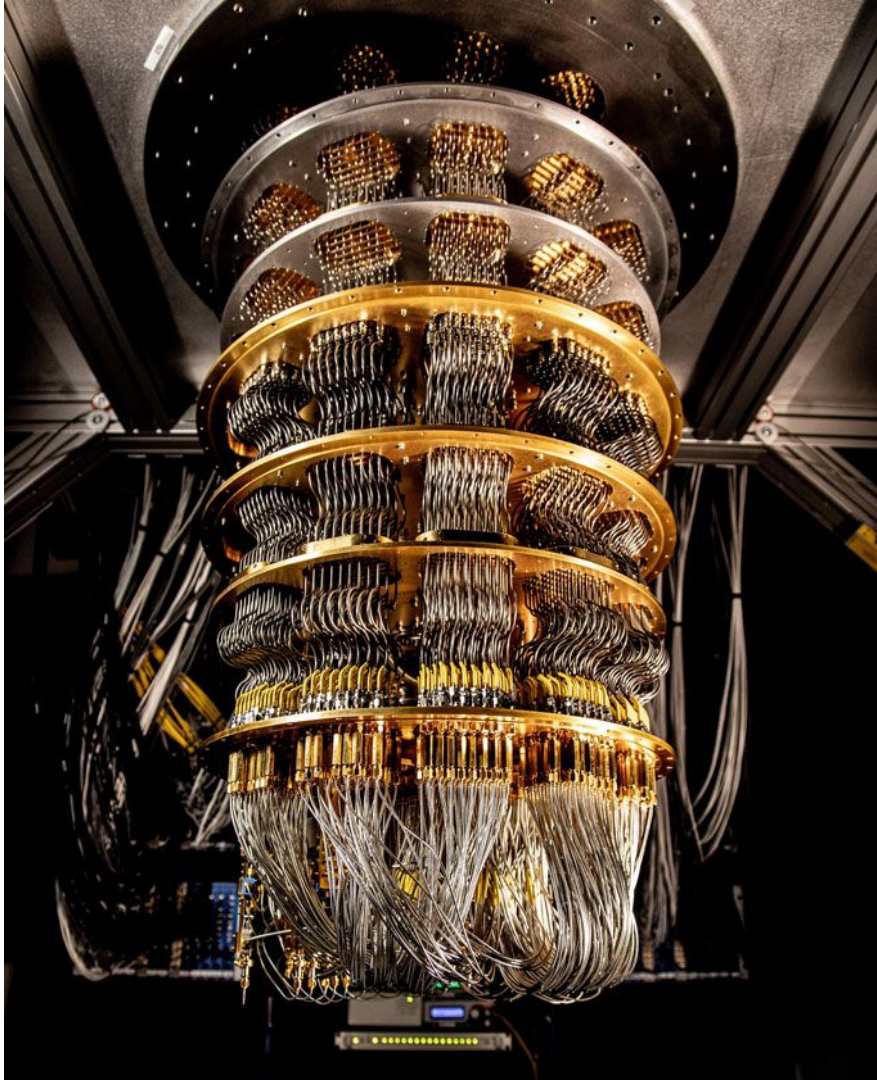
Security and Privacy Research

# Not all 2FA technologies are equal




| | Secondary email | SMS verification | Device prompt | Security key |
|---|---|---|---|---|
| Boutique phishing | 68% | 96% | 99% | 100% |
| Spear-phishing | 53-100% | 76% | 90% | 100% |

# Quantum computers and attacks are coming



Google

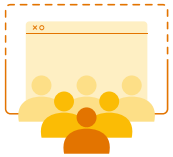
# Why Now?

## Users will need new security keys

Most security keys are not upgradable

## Web infrastructure needs to be updated

Rolling out of new cryptography to the whole web takes time



## User credentials must be recreated

After roll out all users need to re-register on each service

# The first open source security key with a post-quantum hybrid signature scheme

Google

Security and Privacy Research

# Agenda

FIDO Protocol

Hybrid Signature Scheme

Security Key Implementation

Evaluation

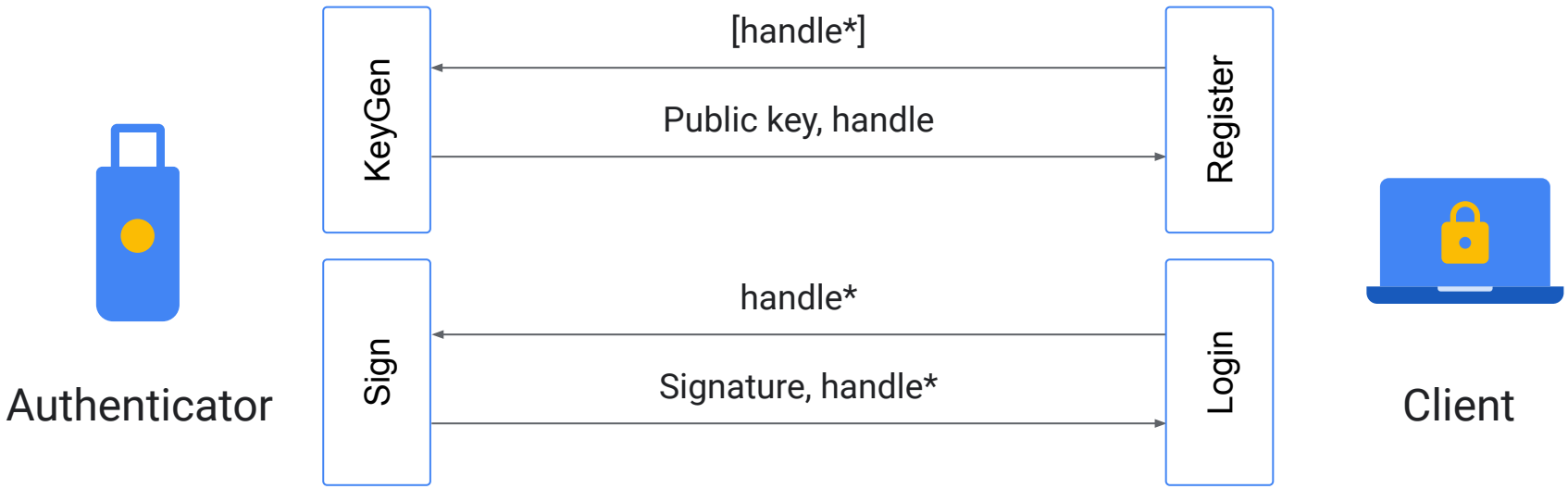
# FIDO Overview


Authenticator

Client

Relying Party

# CTAP




* wrapped private key

# Challenges

Protocol design:
Integrate PQC
into CTAP

PQC algorithm choice:
Dilithium and Falcon
are NIST winners

Embedded hardware
limitations:
memory, speed

# Hybrid Signature Scheme

Google

Security and Privacy Research

# What is a combiner?

A hybrid signature scheme, consisting of classical and quantum-secure algorithms.

# When is Hybrid Useful?

*Classical is still secure...*

1. **No cryptographically relevant quantum computers yet**
2. **Classical signatures withstand classical computers**

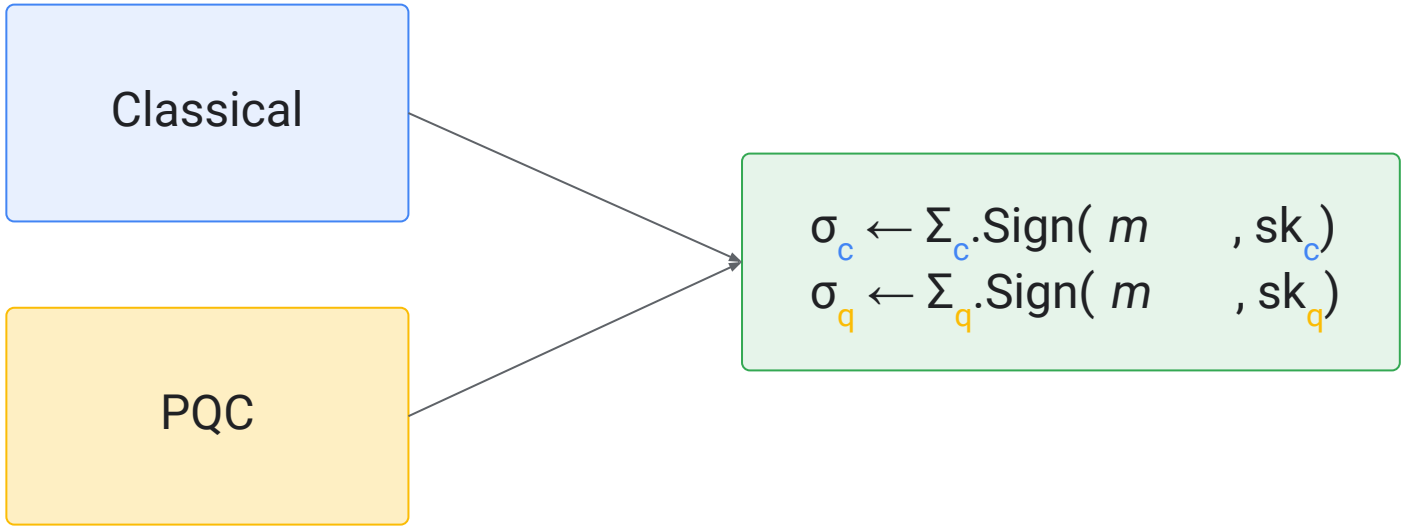
*... and necessary*

3. **Classical computers might break PQC**

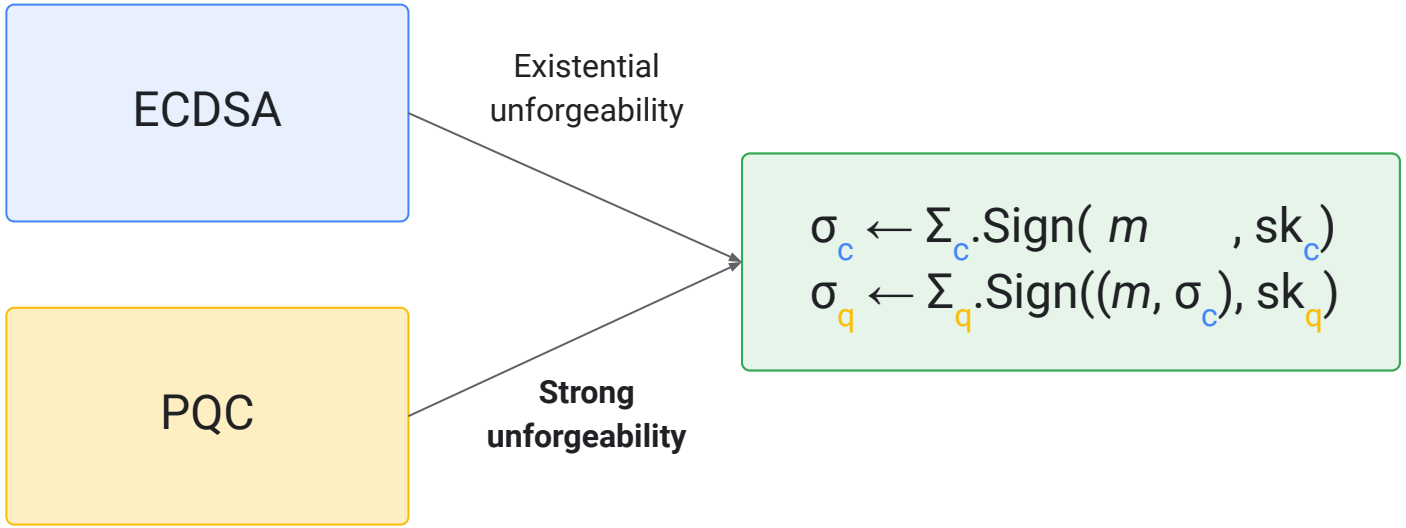
   i.e. see recent attack on Rainbow [Beullens]

# Goal:
# Maintain the security of both underlying schemes!

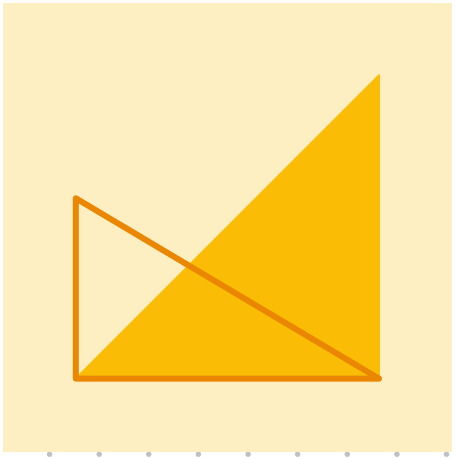
# Simple Combiner



# Strong Nesting



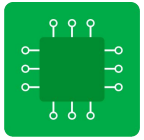
# Security Key Implementation



Google

Security and Privacy Research

Signature and key sizes



Memory

Runtime

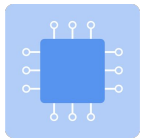
# Can we meet PQC resource requirements?

# PQC Algorithm Options


Dilithium
Modes: 2, 3, 5

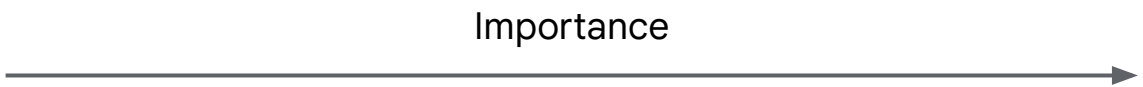
Falcon
Modes: 512, 1024

# Hardware & CTAP Requirements




| | Memory | Public key / Signature | Private key | Signing speed |
|---|---|---|---|---|
| **Limit** | 64 kB | 7609 B | << 7609 B | << 10 s |

# Hardware & CTAP Requirements

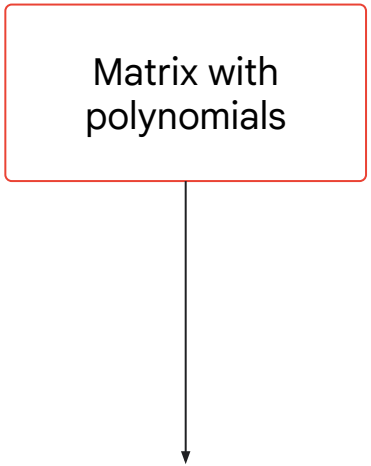



| | Memory | Public key / Signature | Private key | Signing speed |
|---|---|---|---|---|
| **Limit** | 64 kB | 7609 B | << 7609 B | << 10 s |

# Desktop Benchmarks (NIST)

Importance →

| | Memory | Public key / Signature | Private key | Signing speed |
|---|---|---|---|---|
| **Limit** | 64 kB | 7609 B | << 7609 B | << 10 s |
| **Dilithium5** | > 128 kB | 2592 B / 4595 B | 4864 B | 13k sign / s |
| **Falcon1024** | 40 kB | 1793 B / 1233 B | 2305 B | 3k sign / s |



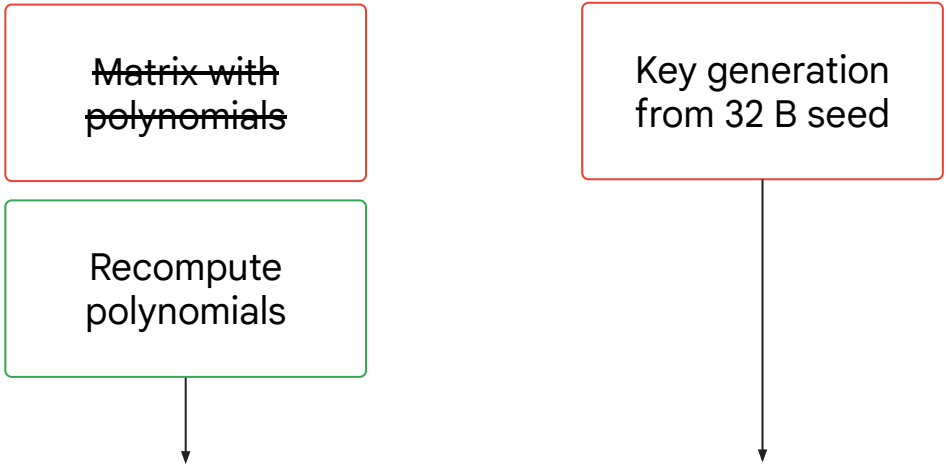

| | Memory | Public key / Signature | Private key | Signing speed |
|---|---|---|---|---|
| **Limit** | 64 kB | 7609 B | << 7609 B | << 10 s |
| **Dilithium5** | > 128 kB | 2592 B / 4595 B | 4864 B | ? |

~~Matrix with polynomials~~

Recompute polynomials



| | Memory | Public key / Signature | Private key | Signing speed |
|---|---|---|---|---|
| **Limit** | 64 kB | 7609 B | << 7609 B | << 10 s |
| **Dilithium5** | 19 kB | 2592 B / 4595 B | 4864 B | ? |




| | Memory | Public key / Signature | Private key | Signing speed |
|---|---|---|---|---|
| **Limit** | 64 kB | 7609 B | << 7609 B | << 10 s |
| **Dilithium5** | 19 kB | 2592 B / 4595 B | 4864 B | ? |

~~Matrix with polynomials~~

Recompute polynomials

~~Key generation from 32 B seed~~

Store seed, deterministically regenerate key

| | Memory | Public key / Signature | Private key | Signing speed |
|---|---|---|---|---|
| **Limit** | 64 kB | 7609 B | << 7609 B | << 10 s |
| **Dilithium5** | 19 kB | 2592 B / 4595 B | 32 B | ? |

~~Matrix with polynomials~~

Recompute polynomials

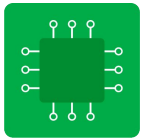
~~Key generation from 32 B seed~~

Store seed, deterministically regenerate key

| | Memory | Public key / Signature | Private key | Signing speed |
|---|---|---|---|---|
| **Limit** | 64 kB | 7609 B | << 7609 B | << 10 s |
| **Dilithium5** | 19 kB | 2592 B / 4595 B | 32 B | 3.3 s<br>2.3x |

for embedded

Binary size

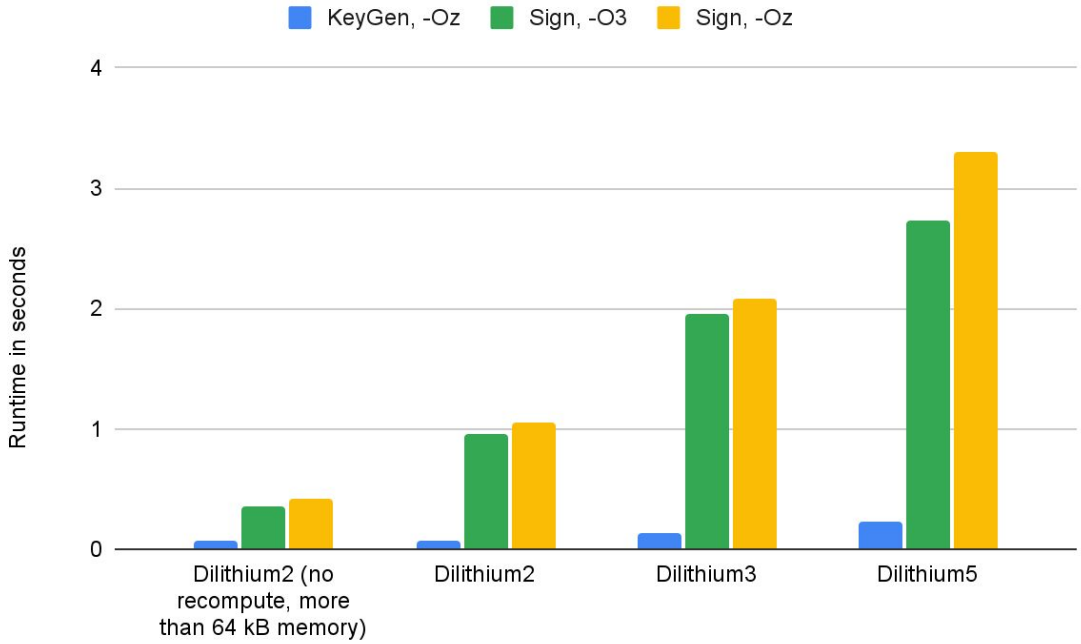


Memory

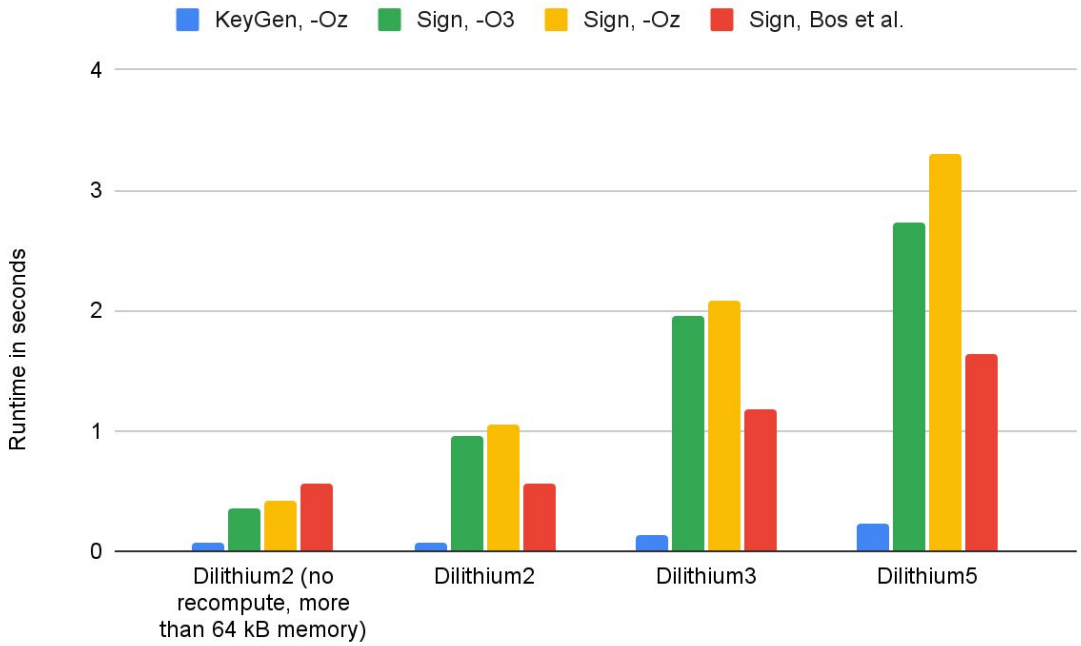
Speed

# Many possible trade-offs

# Speed Benchmark



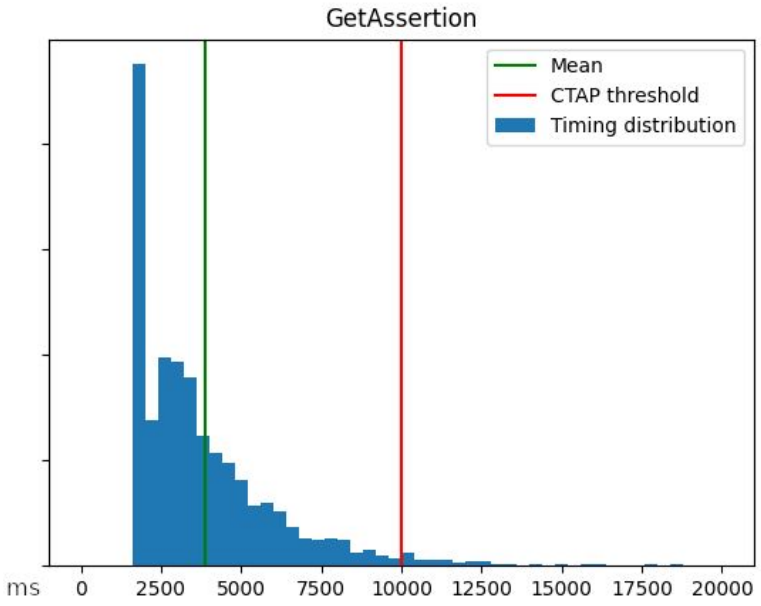
# Speed Benchmark



# Implementation Comparison

| | This work | Bos et al. |
|---|---|---|
| **OS** | TockOS | None |
| **Language** | Rust | C |
| **Configuration** | Flexible | Memory optimized |
| **Source** | Open | Closed |

# Signing Usability




Long-tail distribution

# Signing Usability

| Hybrid signing | < 1 s | < 2 s | < 10 s | Mean |
|---|---|---|---|---|
| Dilithium2 (no recompute) | 85% | 98% | 100% | 0.7 s |
| Dilithium2 | 43% | 80% | 100% | 1.4 s |
| Dilithium3 | 20% | 54% | 99% | 2.4 s |
| Dilithium5 | 0% | 31% | 98% | 3.4 s |

Dilithium is usable, but slow. Good UX needs hardware acceleration.

# Try our open source research framework:

github.com/google/OpenSK

*Tag for this work: hybrid-pqc*