Funded by the European Union under Grant Agreement No. 101087529. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or European Research Executive Agency. Neither the European Union nor the granting authority can be held responsible for them.

Co-funded by the European Union

# Integrating Post-Quantum Algorithms In Practice

CHESS

Cyber-security Excellence Hub in Estonia and South Moravia

*Lasaris Lab*

*9.11.2023*

Petr Muzikant

Information Security Research Institute @ Cybernetica, Estonia

[This presentation includes clickable links]

CYBERNETICA

# Presentation outline

- **Introduction**
- **Our experience with PQC implementation**
- **Current state of PQC**
  - Libraries, ASN.1, JSON Web Algorithms, Hybrid modes
- **General remarks**
  - Preparations, technological constraints, implementation
- **Conclusions**

# Introduction

- **Quantum PC → ~~RSA, ECDSA, …~~ → PQC → new algorithms**
- **Standardization of PQC** (e.g. NIST)
- **Next step**:
  - PQ support in all system and architecture layers
  - Ensure functionality, compatibility, interoperability
- **My activities**:
  - Exploring current options and state-of-the-art
  - Focus on engineering aspects of PQ protocol implementations
  - Gather experience, remarks, and tips

# Post-Quantum Algorithms

- NIST standardization efforts (2016-now)
  - Round 4:
    - **Digital signatures:**
      - Dilithium, Falcon (lattice-based)
      - Sphincs+ (hash-based)
    - **Key Encapsulation Mechanisms:**
      - Kyber (lattice-based)
  - More algs.: TBA ("On ramp")
- other evaluation efforts (BSI, ENISA, ...) → possibly other algorithms

# Our experience with PQC implementation

PQ versions of Web-eID, CDOC2, ASICE, IVXV, TOPCOAT

# Project with direct implementation

- **PQ-Web-eID**
  - authentication framework for web applications
  - Estonian electronic ID cards + state web services
  - dig. signatures, ~~smart cards~~ → ESP32 programmable microcontroller
- **PQ-CDOC2**
  - Estonian standard for securing and exchanging data
  - KEMs, problems with TLS
- **PQ-ASiC-E**
  - (almost Estonian) standard for digitally signed container of data

# Project with problematic implementations

- **PQ-IVXV**
  - electronic voting system
  - preparation of infrastructure, PQ-OCSP, PQ-TSA
  - dig. signatures are OK, but vote encryption is problem
    - ~~elGamal~~ → completely new PQ protocol (lattice-based)
- **TOPCOAT**
  - threshold digital signature scheme
  - almost no existing implementations → completely new PQ protocol (lattice-based)

# Go-to libraries

- PQClean (C)
  - *Cleaned* aggregation of NIST-submitted algorithms
  - Source of source-code (i.e. not a library)
- libOQS (C)
  - + wrappers for C++, Python, Java, Go, .NET, and Rust
  - + applications built with libOQS (OpenSSL, OpenSSH, OpenVPN forks)
- BouncyCastle (Java), rustpq/pqcrypto (Rust), pqm4 (C, Cortex-M4), botan-pq (C++)
- custom wrappers

# PQ ASN.1 structures

- **No standards exist yet** → NIST requires raw bytes
- RFC drafts
  - private and public keys with alg. specific parameters
  - e.g. `DilithiumPrivateKey` (contains `nonce`, `tr`, `s1`, `s2`, `t0`, etc.)
- **PQ Object Identifiers (OIDs)**:
  - OQS defined their own (most commonly used so far)
  - BouncyCastle expanded with KEMs

# JSON Web Algorithms (RFC 7518)

- Usage: *JW Signature*
- Format: *(DIGSIG + HASH identification)*
- Example: *ES384 = ECDSA using P-384 curve and SHA-384*
- **No RFC drafts for PQ JWAs**
- RFC draft for **PQ JW Encodings**
  - *e.g. CRYDI5 = CRYSTALS-Dilithium parameter set 5*
  - *HASH? → SHA-512*
- RFC drafts for PQ JW Object Signing and Encryption (JOSE)
  - alg. specific parameters encoding

# Hybrid mode (PQ + classic crypto)

- Longevity of data confidentiality + protection against emerging threats in PQC
- **Concatenation, sequential** modes
  - *Ghinea et al.*
    - both have security issues
    - novel method to improve unforgeability of ECDSA+PQ signatures
- RFC Draft for hybrid **KEM in TLS1.3** follows **concatenation**
- **Cloudflare and Google Chrome** follow RFC draft using **concatenation** (X25519 + Kyber)

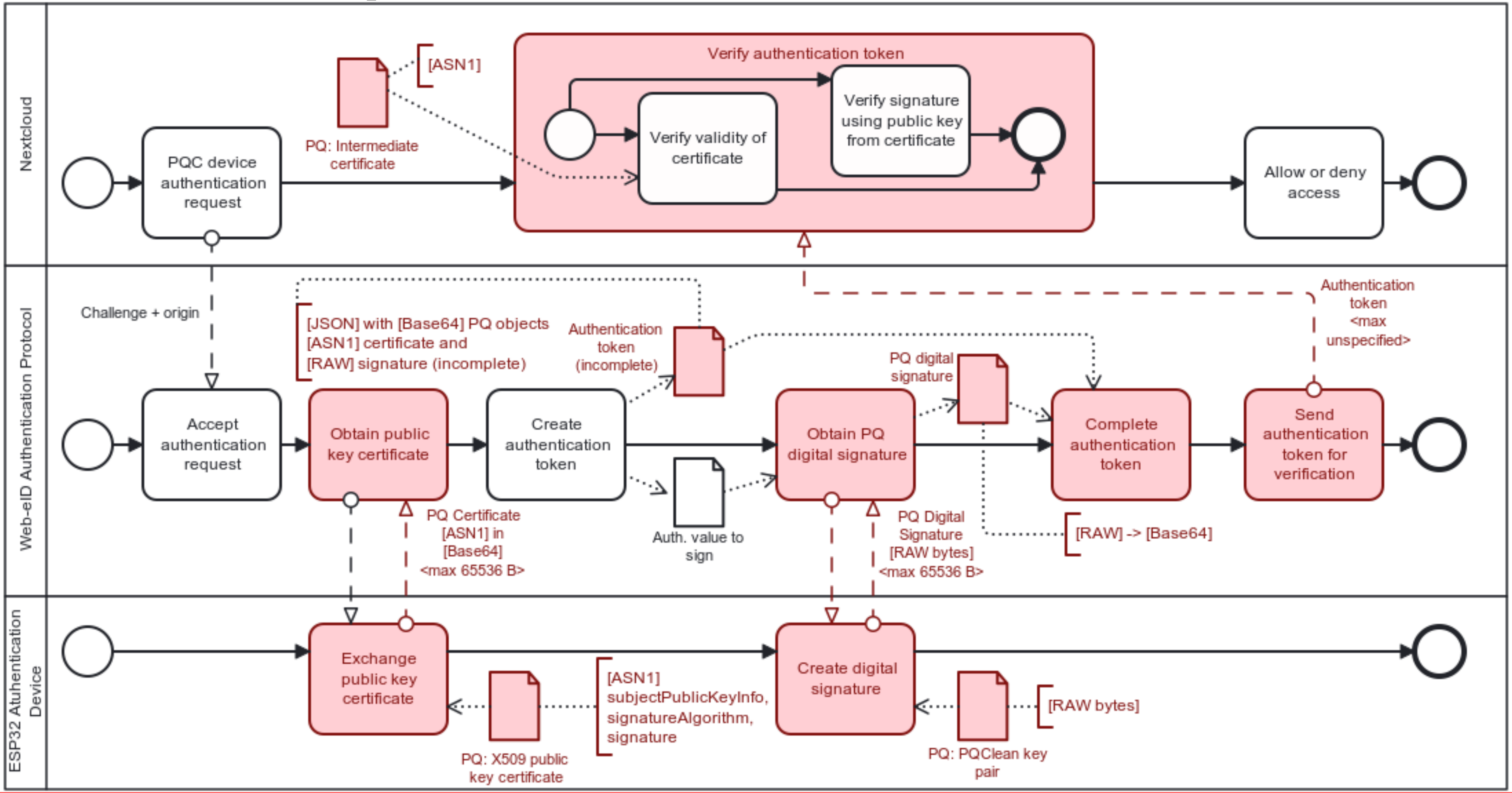
# General remarks

Preparations, technological constraints, implementation

# Identifying relevant locations

- **Identify all PKI objects through their lifetime**
- **Beware of MTUs**
  - bigger objects, variable sizes (Falcon)
- **Beware of changing data formats**
  - ASN1, Base64, PEM, JOSE, other...

# BPMN example






CYBERNETICA

# Technological constraints

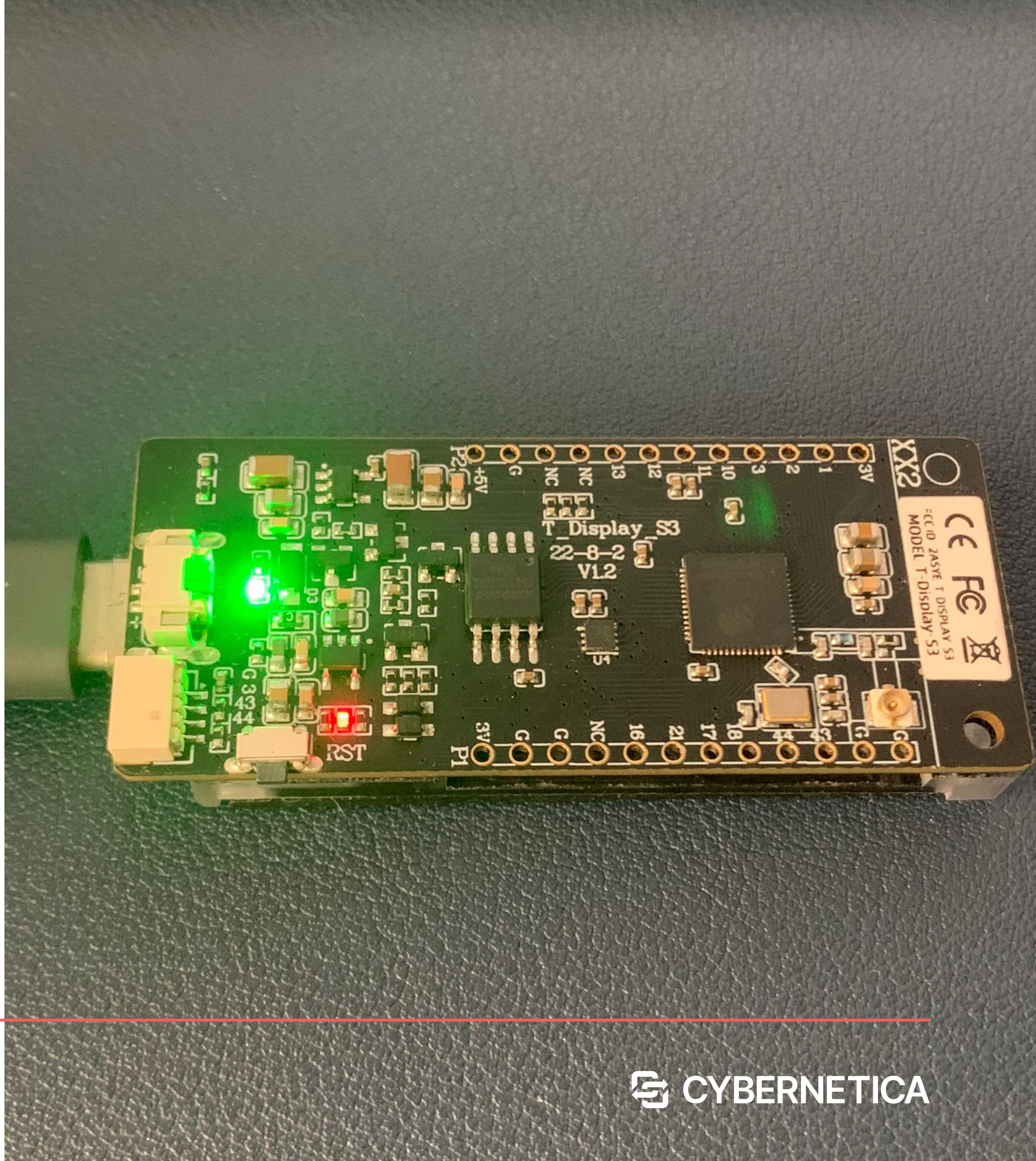
- Assess current boundaries of the system
  - Increased **performance**, **memory**, and **storage overhead**
  - Limited devices and slow networks
- **Protocol adjustments**:
  - streaming public keys and signatures into memory
  - key encapsulation instead of digital signatures (credit cards)
  - allocate all objects in heap instead of limited stack memory (our case)

# Implementation

- Start at the **beginning of the data lifecycle**
- Implement PQC **one step at a time**
- Implementation is still **not straight-forward**
- **Extensions, adjustments, adaptations** of crypto libraries
- **Create your own wrappers** using SWIG
- Expect future changes - standardization is not over!

# Embedded devices

- Smart cards are not suitable → LilyGO T-Display-S3
- Problematic memory management:
  - Limited to 8 KB of stack RAM
  - *PQClean* allocates to a stack a lot
- Solved by adjusting *PQClean* code by using:
  - `malloc` and `free` functions
  - `std::unique_ptr` (C++ v11)



# libOQS extensions

- Available wrappers for C++, Python, Java, Go, .NET, and Rust
  - ~~PHP~~? → SWIG wrapper generator!
- C/C++ interface definition required
  - → *liboqs-php*, *liboqs-python*, *liboqs-go*
- Some remapping was required:
  - PHP `string` ↔ C++ `uint8_t*`
  - Python `bytes` ↔ C++ `uint8_t*`

# PQ in PHP

- **OpenSSL usage → OQS-OpenSSL**
  - (v1 forks discontinued)
  - v3 extension provider:
    - extends regular OpenSSL@3
    - PHP algorithms IDs hardcoded for DSA, DH, RSA, and EC
      - some built-in functions do not require algID (e.g. `openssl_verify()`)
  - my notes on OQS-OpenSSL in PHP
- **PHPSecLib → new PQC-PHPSecLib fork**
  - *OQS-OpenSSL* or *liboqs-php* (based on availability)



CYBERNETICA

# PQ in BouncyCastle (v1.74+)

- Not well documented
  - bc-java / core / src / main / java / org / bouncycastle / asn1 / bc / BCObjectIdentifiers.java
  - `org.bouncycastle.pqc.*` packages
- Works with actual algorithm parameters from ASN1 drafts
  - vs raw bytes in libOQS
  - e.g. `KyberPublicKeyParameters` has `t` and `rho`

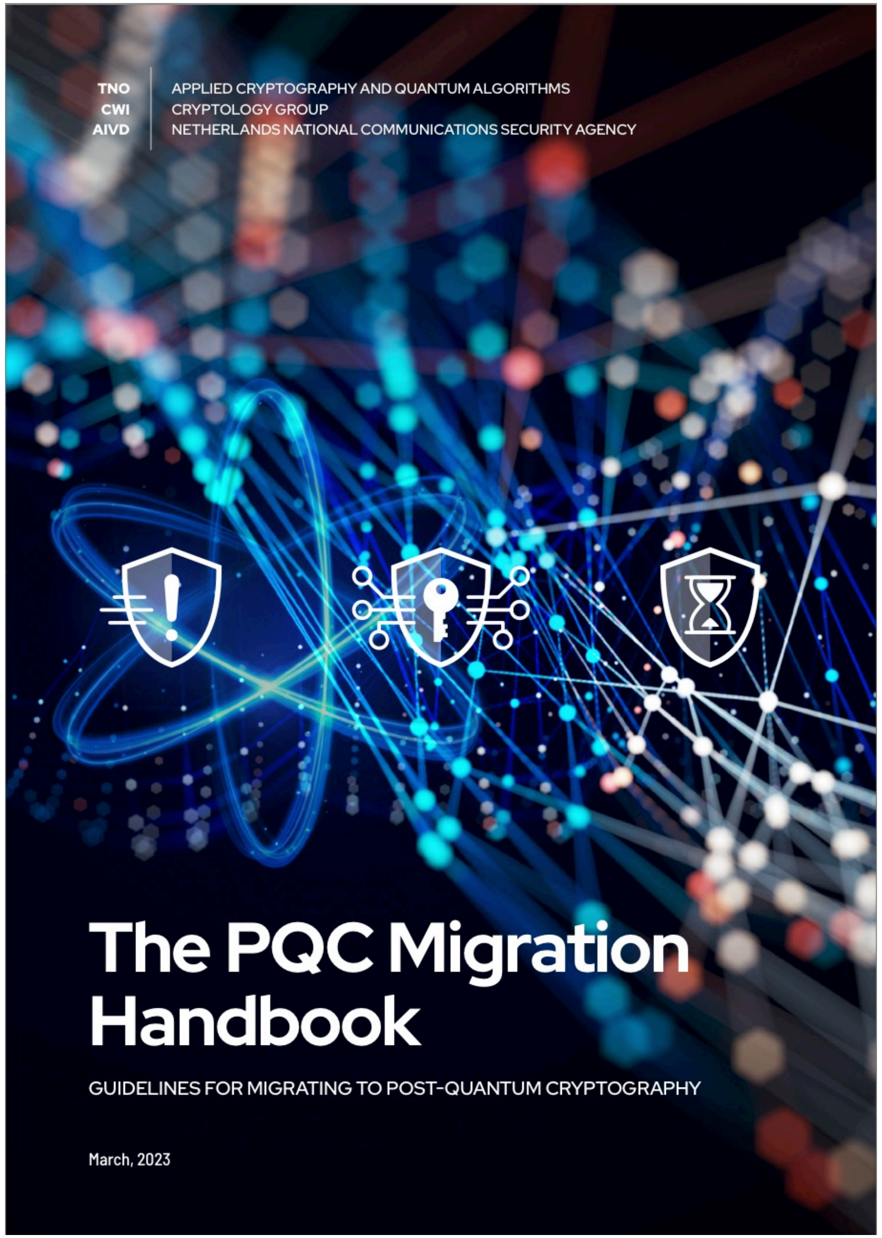
# PQ Java Keytool

- keytool = command for managing a keystore of cryptographic objects
- PQ BouncyCastle → PQ Java Keytool
- e.g. to generate .p12 with Dilithium keypair and self-signed certificate:

```
keytool \
    -providerpath bcprov-jdk18on-175.jar \
    -provider org.bouncycastle.pqc.jcajce.provider.BouncyCastlePQCProvider \
    -genkeypair \
    -keyalg Dilithium5 \
    -alias cdoc20-client-pqc-CA \
    -keystore cdoc20clientpqcCA.p12 \
    -storepass passwd \
    -sigalg Dilithium5 \
    -dname "CN=cdoc20-client-pqc-CA,OU=ISRI,O=CyberneticaAS,L=Brno,S=Czechia,C=CZ"
```

# This week's news

- **Post-Quantum Cryptography conference 2023** by PKI Consortium
  - Recording on YT
  - PQC Migration Handbook
  - Experiments on embedded devices are unrealistic
  - FIDO2 tokens are highly limited
  - PQC on mobile phones might require new HW co-processors
  - banking is XY years behind



# Conclusions

- **Implementing PQC today is...**
  - **...complicated**:
    - different libraries → different approaches and documentation level
    - computational constraints, adaptation and tweaking
    - standardization is not finished (and complete - MPC, ZKP, etc..)
  - **...doable:**
    - authentication framework, crypto systems for encryption, signing containers, etc.
  - **...worth it:**
    - long-term data protection, experience
  - **...helpful:**
    - big space for open-source PQ contributions, reduce confusion

# Thank you for listening!

**References:**

- links in presentation
- PQ authentication framework
- Notes on PQC in PHP
- write me an email!

https://cyber.ee/

info@cyber.ee

cybernetica

CyberneticaAS

cybernetica_ee

Cybernetica

Petr Muzikant, petr.muzikant@cyber.ee

CYBERNETICA