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

Post-Quantum Cryptography for Engineers: Technical Overview

CHESS

Cyber-security Excellence Hub in Estonia and South Moravia

The Future Cryptography Conference

13.05.2024, Tallinn

Petr Muzikant

Information Security Research Institute @ Cybernetica AS, Estonia

[This presentation includes clickable links]

CYBERNETICA

Presentation Outline

1. **Introduction**
2. **Our Experience with PQC Implementation**
3. **How to Begin?**
    - Preparations, Technological Constraints, Implementation
4. **Where to Begin?**
    - PQ Algorithms, Cryptographic Libraries, Encodings, Hybrid modes
5. **Engineering Obstacles**
6. **Conclusions**

Introduction

- **Existing work:**
  - Timelines, "Migration Challenges", PQC Migration Handbook
- **The issue:**
  - How to actually migrate? Does a general security engineer have everything in their disposal for that?
- **Our work:**
  - Explore and support current FOSS state-of-the-art
  - Focus on engineering aspects of PQ implementations
  - Gather experience, problems, and remarks

Our Experience with PQC Implementation

- **e-Governance applications and frameworks**
  - Web-eID (Authentication)
  - CDOC2 (Encryption)
  - ASiC-E (Digital Signatures)
  - IVXV (e-Voting)
- **Supporting projects**
  - PQ library wrappers, extensions for crypto libraries
  - Lattice-based crypto development kit
  - PQ OCSP, TSA solutions

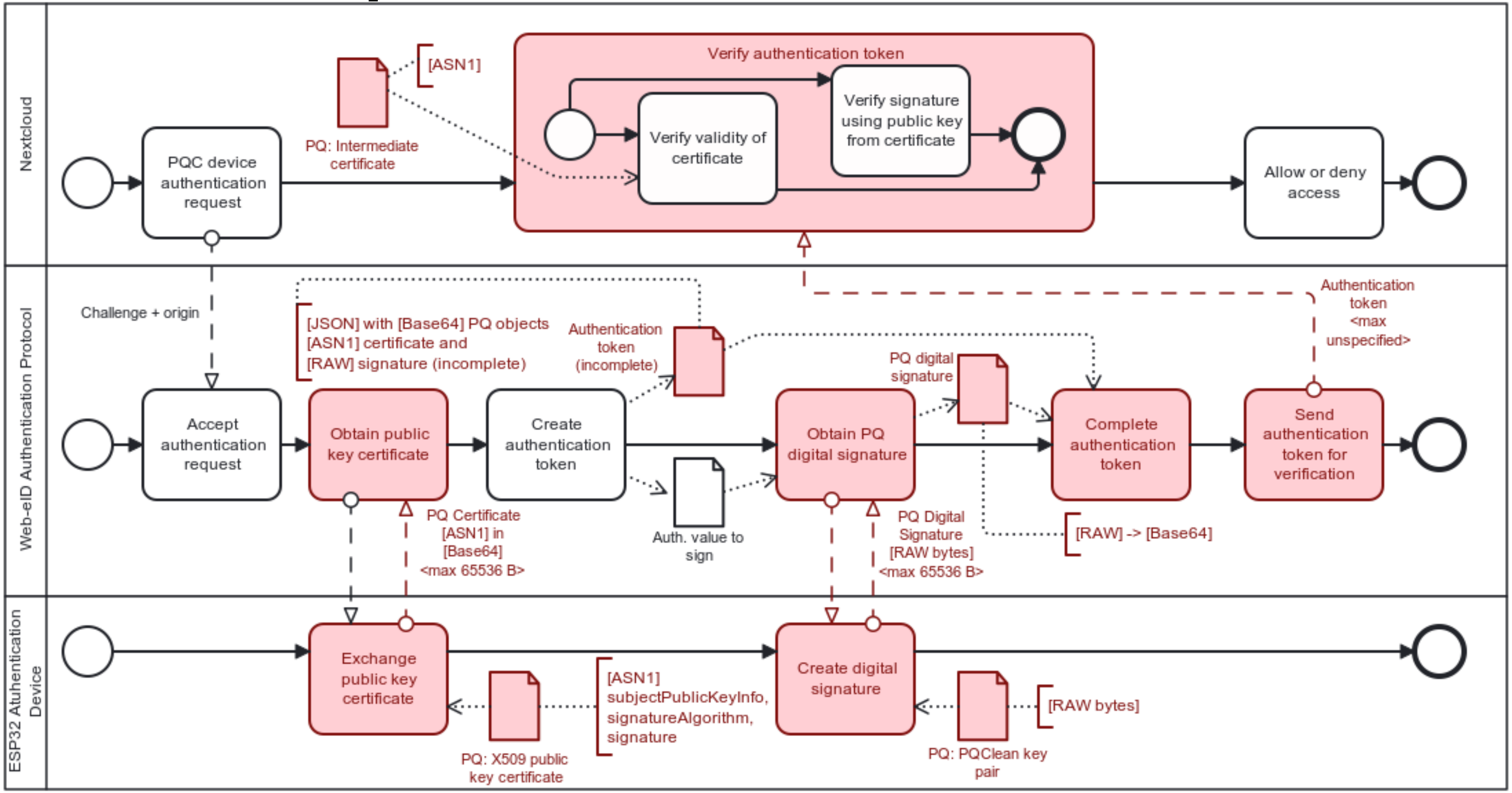
How to Begin?

Preparation, Technological Constraints, Implementation

Preparation

- **Identify all PKI objects and their lifetime in the system**
  - understand the extent of required changes
  - dig. signatures, key agreements, ...
- **Beware of MTUs**
  - PQ = bigger object sizes, sometimes even variable size (Falcon)
- **Beware of changing data formats**
  - ASN1, Base64, PEM, JOSE, other...

BPMN Example



Technological Constraints

- Assess current boundaries of the system
  - Increased **performance**, **memory**, and **storage overhead**
  - Limited devices and slow networks
- Possible **protocol adjustments**:
  - streaming public keys and signatures into memory
  - key encapsulation instead of digital signatures (credit cards)
  - objects allocations on embedded devices (stack → heap)

Implementation

- Start at the beginning of the data lifecycle → step-by-step
- **Extensions, adjustments, adaptations** of crypto libraries
- Expect future changes - standardization is not over!
  - → Crypto agility
- Rest of this presentation

Where to Begin?

PQ Algorithms, Cryptographic Libraries, Encodings,
Hybrid modes

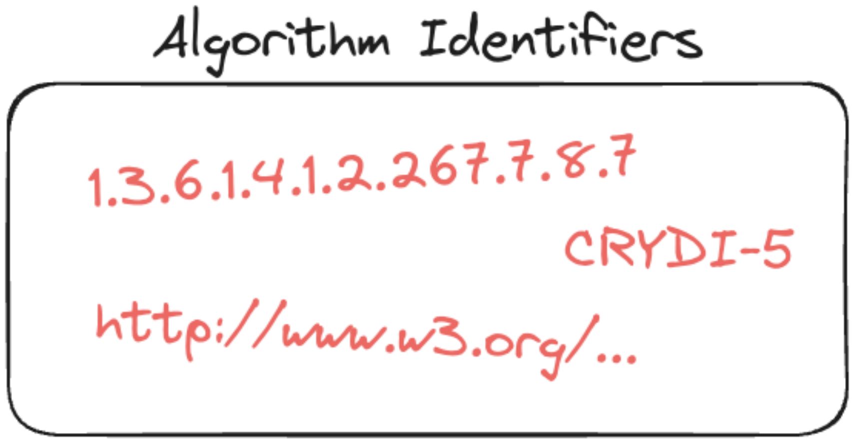
Post-Quantum Algorithms

- NIST standardization process (2016-now)
  - **Key Encapsulation Mechanisms:**
    1. ~~Kyber~~ → **ML-KEM** (FIPS 203)
    2. + round 4 (soon)
  - **Digital signatures:**
    1. ~~Dilithium~~ → **ML-DSA** (FIPS 204)
    2. ~~Sphincs+~~ → **SLH-DSA** (FIPS 205)
    3. Falcon → ~~FN-DSA~~ (TBD Q3 2024)
    4. + "on-ramp" round 1 (not before 2027)
- other evaluation efforts (BSI, ENISA, ...) → possibly more algorithms

Cryptographic Libraries

- PQClean (C)
  - *Cleaned* aggregation of NIST-submitted algorithms (latest + last round)
  - 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)
- custom wrappers of libOQS

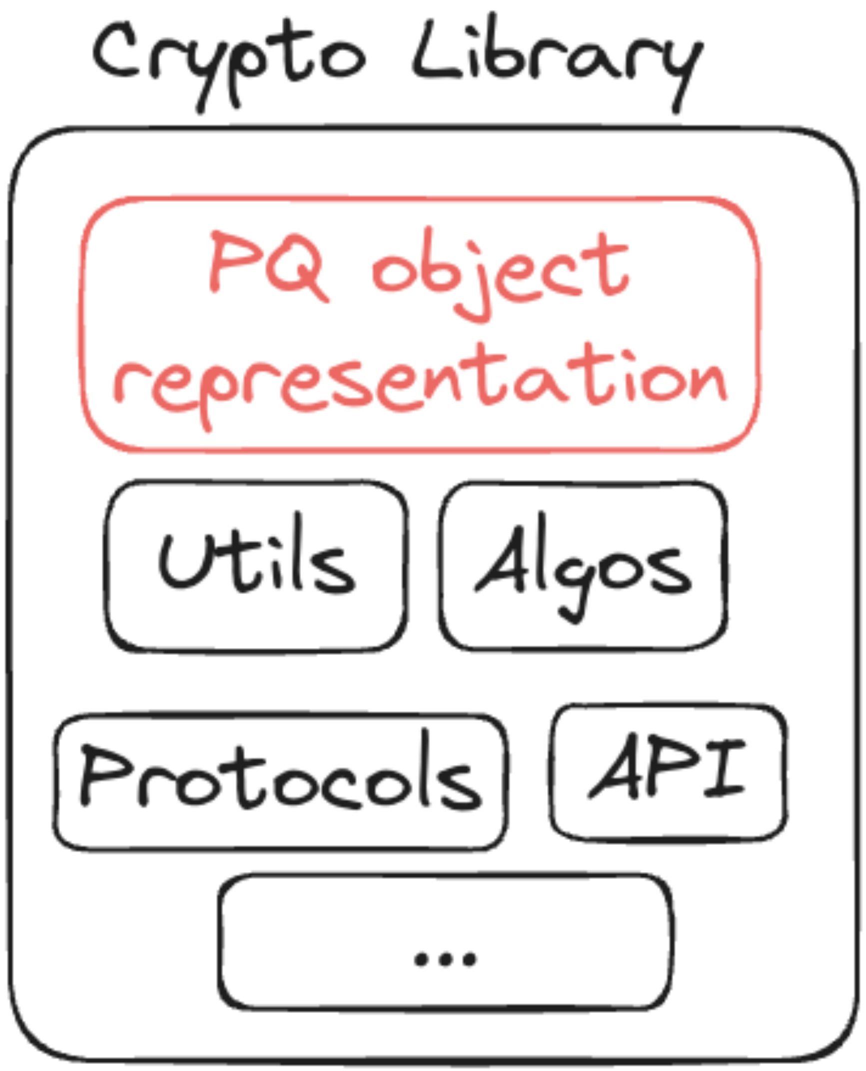
Algorithm Identifiers



- **ASN.1 Object Identifiers**
  - OQS, IETF Hackathon OID lists
- **JSON Web Algorithms**
  - RFC, but only for KEMs
- **XML Signature Syntax Algorithms**
  - ...
- **Other identifiers**

Object Encoding

- **Raw bytes**
  - originally NIST submission rule, now in libOQS
  - output from one function = input for second function
- **PQ ASN.1 structures**
  - IETF Hackathon - PQC certificates
    - tries to solve compatibility issues and unify structures
  - BouncyCastle maps to classes



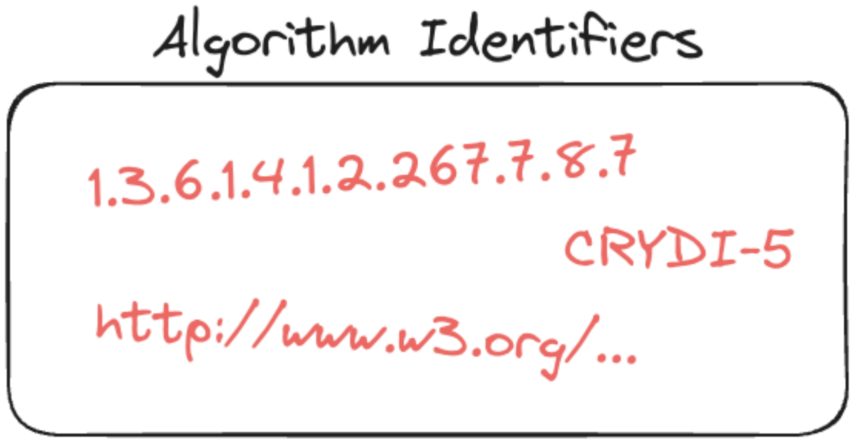
Hybrid mode (PQ + classic crypto)

- **Post-quantum cryptography:**
  - ensures the longevity of data protection
- **Classical cryptography:**
  - protects against emerging threats on unexplored PQC
- **Most common modes: concatenation or sequential**
  - both can have their issues → nothing concrete yet
  - RFC Draft for hybrid **KEM in TLS1.3** uses **concatenation**
  - **Cloudflare and Google Chrome** follow RFC draft using **concatenation** (X25519 + Kyber-768)

Engineering Obstacles

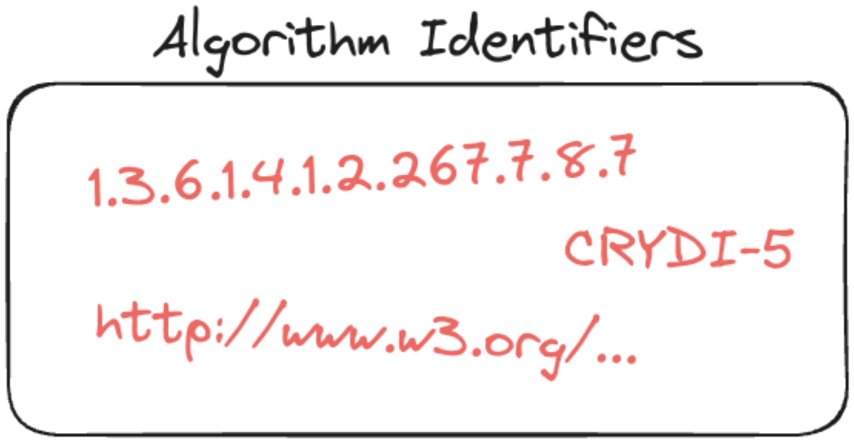
PQC Implementation is far from straight-forward

Algorithm Identifiers



- **ASN.1 Object Identifiers**
  - OQS, IETF Hackathon OID lists
- **JSON Web Algorithms**
  - RFC, but only for KEMs
- **XML Signature Syntax Algorithms**
  - ...
- **Other identifiers**

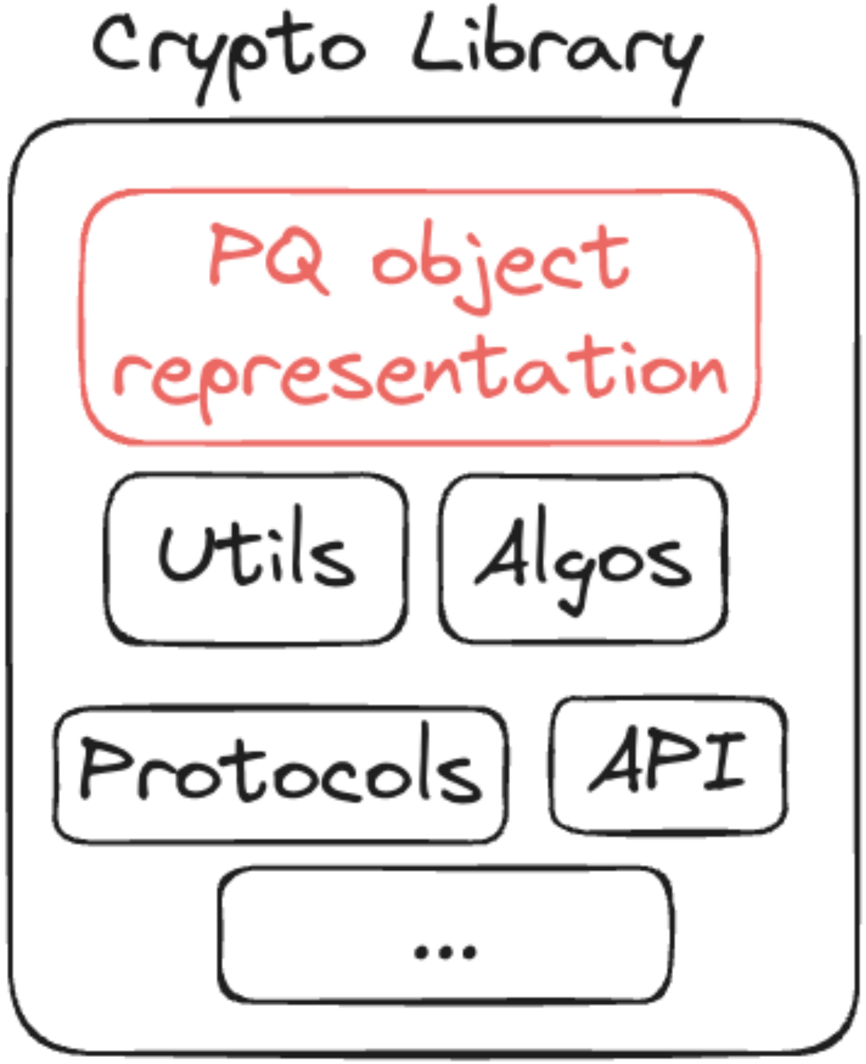
Algorithm Identifiers



- **ASN.1 Object Identifiers**
  - Wild West
  - OQS → BouncyCastle → OQS → IETF Hackathon → ???
  - ML-KEM vs CRYSTALS-Kyber?
- **JSON Web Algorithms**
  - PQ alternative to *ES256*?
  - Recent RFC, but only for KEMs
- **XML Signature Syntax Algorithms**
  - PQ alternative to http://www.w3.org/2001/04/xmldsig-more#rsa-sha256?

Object Encoding

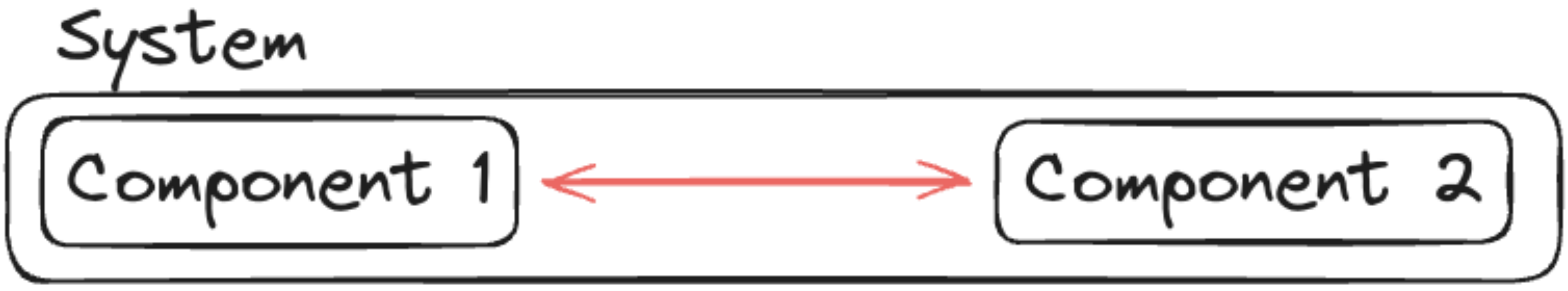
- Most crypto libraries have classical algorithms **hard-coded**
  - PHP extension for OpenSSL, PHPSecLib
  - cryptography, asn1crypto (Python)
  - crypto (Go)
- **Two options**:
  - **Extend** vs **circumvent**



CYBERNETICA

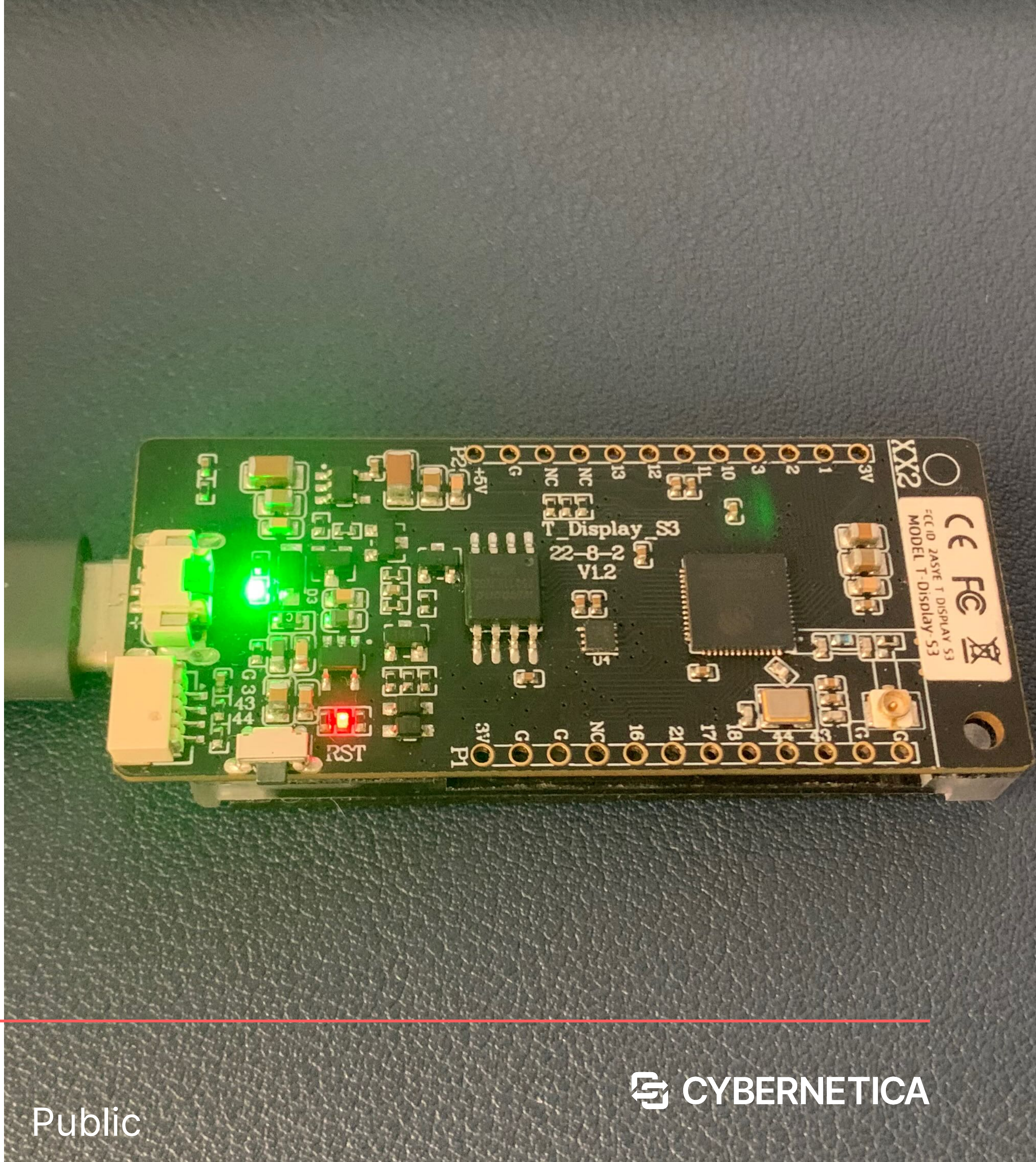
Interoperability Awareness

- Growing with system complexity
- **Active thinking** about all components
  - Identifiers, encoding, MTU, processing



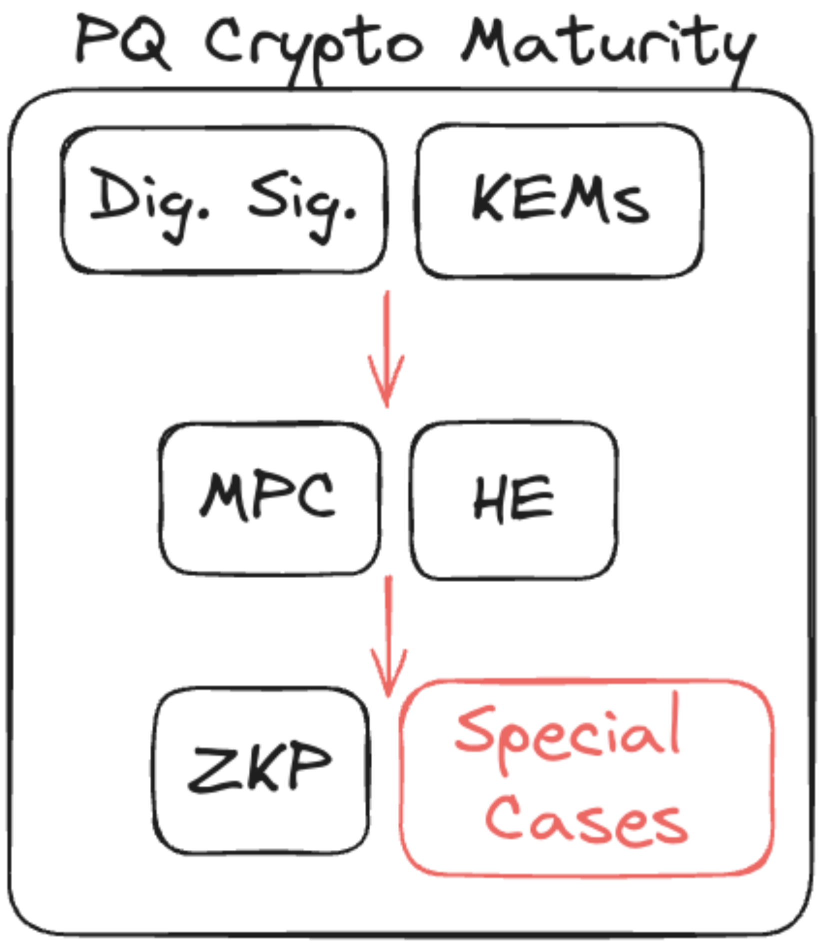
Cryptographic Tokens

- ~~Smart cards~~
- Chip manufacturers?
- Embedded devices for local testing purposes
  - Performance OK (ESP32-S3)
  - Memory OK, but complicated
  - Safety not OK (no HSM, TPM, not certifiable)
- Protocol adjustments might be required (~~stack~~ → heap, streaming)



Maturity of PQ Custom Crypto

- What if application requires:
  - **Multi Party Computation**?
  - **Homomorphic Encryption**?
  - **Special features**?
    - e.g. *ElGamal* in vote encryption - special decryption without private key
- Still lot of **R&D to be done**



Miscellaneous PQ Engineering Efforts

- **All the little things**
  - (OQS-)OpenSSL encodes private keys as:
    - *0×04 or 0×03 || length || private_key || public_key*
  - Custom wrappers → data type conversions
  - Adding single lines into dependencies' files to support PQ
  - Build issues, insufficient or confusing documentation

Conclusions

- **Implementing PQC today is...**
  - **...complicated**
    - not straight-forward
    - different libraries → different approaches and documentation level
    - computational constraints, adaptation and tweaking
  - **...doable**
  - **...worth it**
    - long-term data protection, experience, possibility to set good practices
  - **...helpful**
    - big space for open-source PQ contributions, reduce confusion, helps shaping the industry

Thank you for listening!

References:

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

Petr Muzikant, petr.muzikant@cyber.ee

https://cyber.ee/
info@cyber.ee
cybernetica
CyberneticaAS
cybernetica_ee
Cybernetica

CYBERNETICA