Post-quantum Cryptography
In Digital Identity

- NIST PQC standards
- PQC W3C-VC
- PQC EUDIW
- PQC *AdES
- PQC ZKP

https://forkbomb.solutions/pqc

Andrea D’Intino - 09.05.25

FORKBOMB
:(){ :|:& };:

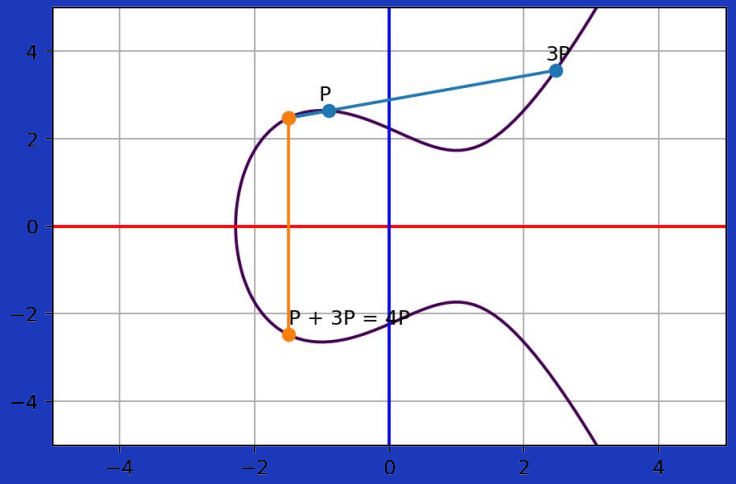
Intro to PQC

NIST standards

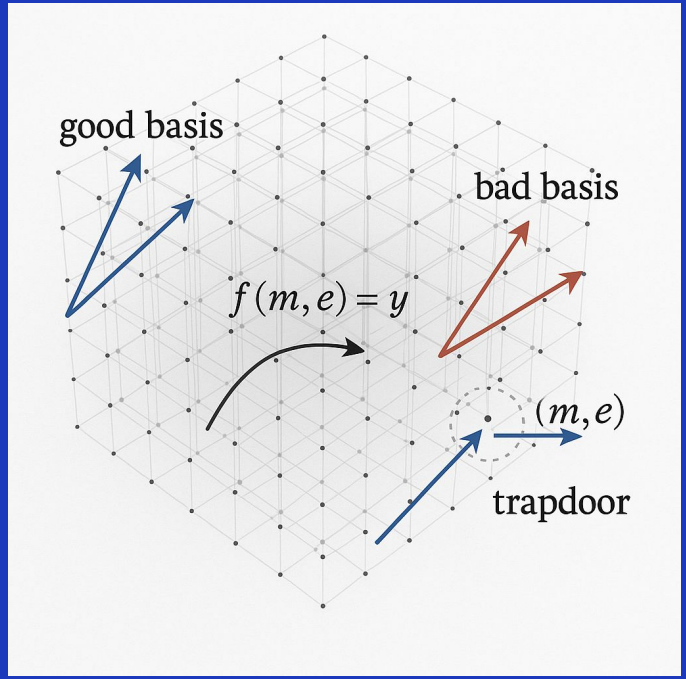
What is PQC? Why do we need it?

- Post-quantum cryptography:
  - IS cryptography running on today's computers and resist Shor's algorithm (quantum computers will break RSA and ECC)
  - IS NOT computation running on quantum hw (e.g. QKD)
- Why? Quantum computers will break rsa/ecdsa/eddsa soon, **hackers are "harvesting" NOW** to decrypt later.

Math: ECC VS Lattice PQC



Courtesy: the Internet



Courtesy: my friend ChatGPT

NIST standards (signatures)

NIST PQC competition 2016-2022, results standardized in August 2024:

- ✅ Kyber (KEM) → ML-KEM (FIPS-203)
- ✅ Dilithium (sig, size: ca 3.2KB) → ML-DSA (FIPS-204)
- ✅ Sphincs+ (sig, size 41KB!) → SLH-DSA (FIPS-205)
- 🚧 Falcon (sig, size 666B/1280B) → FN-DSA/FIPS 206 (WIP)
- 🚧 HQC (KEM, 11.03.25), fallback for ML-KEM (WIP)

NIST PQC standardized implementations

Currently available libraries

- [FIPS 204] ML-DSA (C, Golang, Rust, Java)
- [FIPS 205] SLH-DSA (C, Golang, Java)
- [FIPS 203] ML-KEM (C, Golang, Java)

Note: reference implementation in C, Java in BouncyCastle, Go in CloudFlare's Circl (not SLH-DSA), some implementations also in JS

NIST Considerations for achieving crypto-agility (05.03.2025)

W3C-VC

Quantum-safe W3C-VC standardization CCG

W3C-VC standardization effort

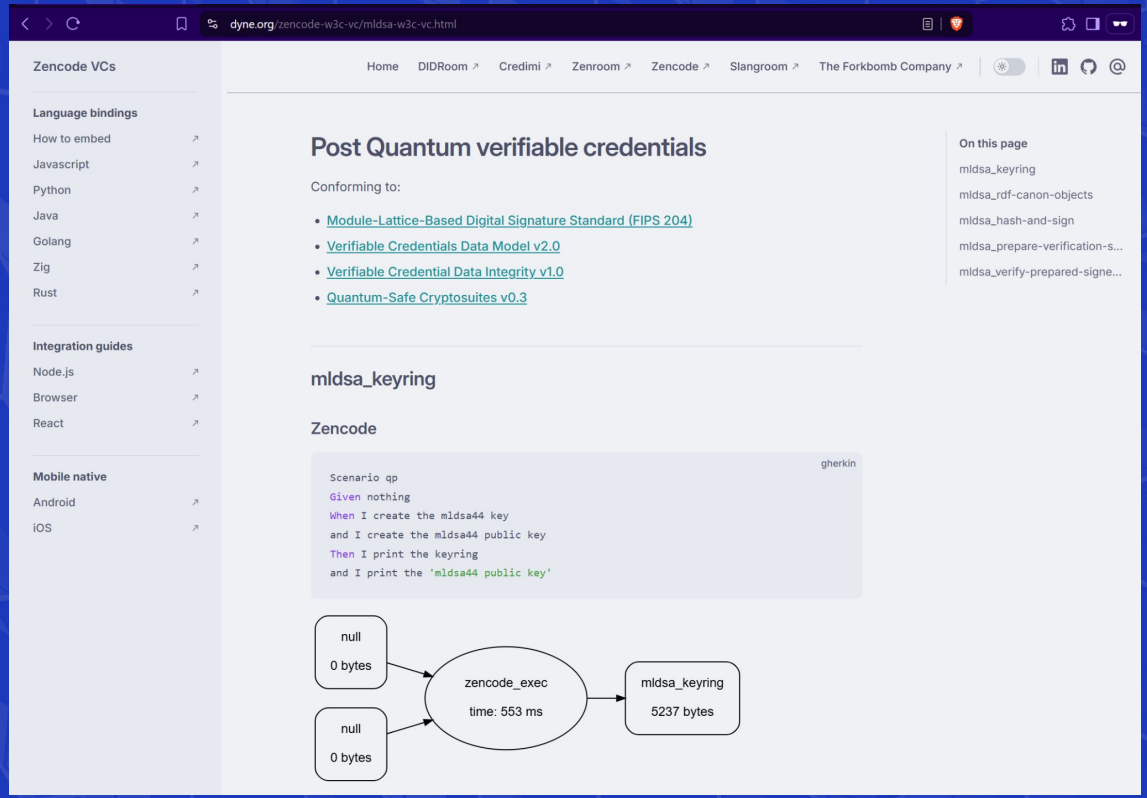
The W3C started a WG for the standardization of “quantum-safe W3C-VC”



https://w3c-ccg.github.io/di-quantum-safe/

Note: will be updated in June 2025

Our work with Quantum Safe W3C-VC



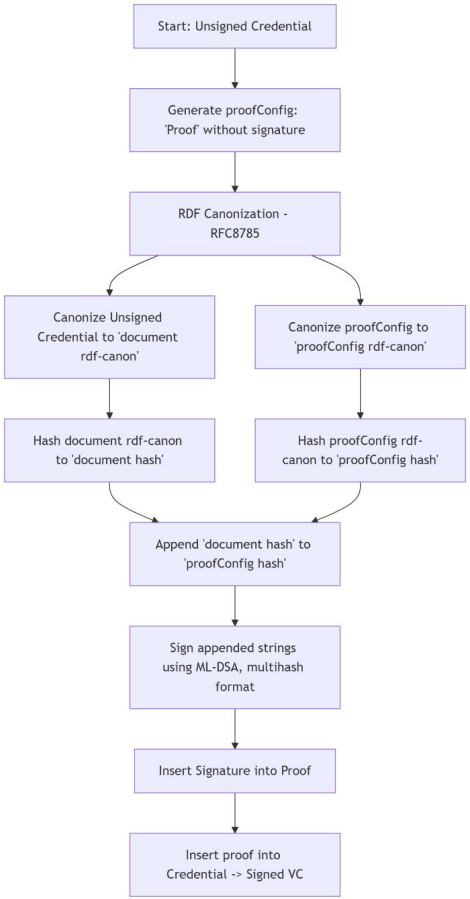
https://dyne.org/zencode-w3c-vc

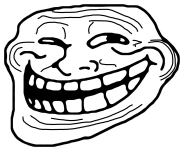
Lifecycle of Quantum Safe W3C-VC

1. Issuer produces (or receives) an unsigned Credential (JSON-LD), containing (pre-authenticated)
2. Produces “proofConfig” (JSON-LD), a.k.a. a “proof” without signature
3. Performs RDF Canonization (RFC8785) from VC Data Integrity:
   a. The unsigned Credential is canonized → *“document rdf-canon”*
   b. The proofConfig is canonized → *“proofConfig rdf-canon”*
4. Hash of canonized objects:
   a. *“document rdf-canon” → “document hash”*
   b. *proofConfig rdf-canon → “proofConfig hash”*
5. Appends “document hash” to “proofConfig hash”
6. Signs the appended strings (output of 5), as multihash
7. Issuer inserts signature in the “proof”, and “proof” in the unsigned Credential

Lifecycle of Quantum Safe W3C-VC







Anatomy of a Quantum Safe W3C-VC


```
{
  "document": {
    "@context": [
      "https://www.w3.org/ns/credentials/v2",
      "https://www.w3.org/ns/credentials/examples/v2"
    ],
    "credentialSubject": {
      "alumniOf": "The School of Examples",
      "id": "did:example:abcdefgh"
    },
    "description": "A minimum viable example of an Alumni Credential.",
    "id": "urn:uuid:58172aac-d8ba-11ed-83dd-0b3aef56cc33",
    "issuer": "https://vc.example/issuers/5678",
    "name": "Alumni Credential",
    "proof": {
      "created": "2025-04-25T09:07:09Z",
      "cryptosuite": "mldsa-rdfc-2025",
      "proofPurpose": "assertionMethod",
      "proofValue":
        "zb97gSRqiC1pSimfapSh82NbgbwBZn985Uq49USZgutFRUNERF4dijsvneLBbRVyrkeBmWcGtyspJhwPqv6HEPaThzGpiztResDMYgKy6P3UYbGn5mjiwW62Bf9TsK1AV3sNhpBwebodV5GJowtrMtwab6yHBjqgCkvvCU5MpUa6wWMxRmTd3tej
        NhMqwdte3uZNPrjVvKiryJe22dJx2PWV8zgC2hWFbJWZLgeTC4jQqZiR75pxm7m4YpejyciGaNraKxHpmGZk4rLWiFsJLcRrBxASZTLCHBztdgW91GyGz9FkMY4TiJyZnR6XY5KTR2V5MdwQ3wYAzhJ8t4w4iHZWt1HZEsZGFQTTCxzCRQDyh1QTf
        GC86FfXmF53GK4xGhUNCjXc6sxrqJkn9KY1rr36cp7Hr8LdZdJfiAX65rD4JCCgfN2XdA2PMRSDr2txtQHb1yEAZRWfAVZViTc2thsx9Z5MEPcbUFq9gPTFFGw9nCaHYa672oTQV63jM6xVvQDa9DUtpm8jdUxpWT8DMoyY8RKLTw4aD928N8u9nL
        kBCSaxA7zrMbBGofxSA9b4BS6JL9nZrRmcMdEQuQrFHEpu2CLERNrbWyXvDwaZzMmB6UmTZZppy9nbj8YFDbg7j93fp36F4mStnE9t7zohdCd4wpked8Awxxkiunuys E15mch1EenswqKKihQfK1mNHeanHo8iMnnehMUtRMbztANpL7Gj18iRS2r
        BkmmMw4KZaZHAgw3oVHvryPwdSpZXNxPzyM5i7LsAfa5ED3oQvifBEDPtVxEdP8YSTTPf491juQfXh9UZGCEsR4rRZVx5vcVoKbCtDv7wQzmzY9CHS9msS7gki9ZfvPvcT56dkpTVPTuK8zyXRXuWTFK3buVg6mBaihtmvTRwfu3Nbnt7P4n9Es8W
        wL6yRYFdYv7GH9PX7PSHfdRyWAUx37hTdweumCgasyQupkpNPfCPqZfnzcaJp7AXDFJA3j4PJ9hiB35KmMLBjY8r91mTXDM9CTYJetvyBEnXC2yRMLpkQvizJa7geEnh8jDC69DHVNQ5Gn4VCZpbGLK7eWMxXxmQAnAbvmmqUXJCy4xry4u6Sngvu
        MrVcBaegNAz4mSjpDj2uY8wQ7KWsAFoT9T6RFviwgKV9QVZmktWh6mDyJ5gJusbaPoG26iBc4nFxgYzBaBDK9g5rx9U4QqeXE5ffpkGMYmFKj8shVLeaFYSE4tvJxQkFfc58K7evEAxQwECFi6Ho7oBUJwWPij4PBBxVZUzbMtWxZPq11PGGZ2Qu4
        2VGrzvRvxHc5DSyxfhsdwbfnT8RLgxUj9znxzKvmYWmMntdoh7GBXTMMtuueK5fcC3AKFjAunAw9gkZZCTwFQiHpbpkvcqaq8uovwX5JDgWciDUAPDbbcza4VQH6H5QG6485XxwonM715Hwru3UVH7G8D5nGrEwYPHs61C8Ai3YReZc4kBNtx9Mir
        F82adyNZqns8YXxac7KU4eWqsTKQFtgc2cBJ661eir3uAA4htpUpwBFMBPniV2RmNtf7AsRVK45UCbVHBYzYdv9Hyzbr4bYYe7Ez7BSikfNmXVf4wXs1M2ccztP9eDLjD7rmVcbxQPz9sEK7Tah2qmzdUEh69mfRWuk8a2MeH78jrU93Tez8oMEew
        ng9SYYJUhdp51bx7fcYrAZrHFwnW3oHPyMTaCaLVzFA4J1kE5JqfKGPYc7vvTYuS3V4bYdFtNXWzhLUUD6swPp2ZoQk1ThtP31DuFqA7jJVoYD2mfZUGsUW3Wi8LwB6P9UgMNyJq7Ej6WWF6CHHEBWvsqUhvejPs7J5hsmD45H1fdNGmzo9uVyx4M
        n6i78FbRr1A8ATojeHEE5hKWRjhbhjpRCnkrcwr7NXvtJacnchTWKLjN4VKWv3JxHQDXpeA7V1X4Yz81LmxY33XQRXi9DoMnWTsVgpMZoLfjZLnfu4EyhtK4bs4bCShXtbFnKV4ZniQxyYCuu7Yhbjuwgz4nqp7ZZaxwrPu8QfSyM4gDNdrmU4LRY
        gktghQdrW4gaxRNL38SFaPDjWj1fK1ZYUpX66QGkyg4taTnNPgTTmmwB6tBBdxaCs9zj6CqmMjhnPVkZmGd5eEg41FJSCXdZ9cdJkVLJqgUwwEFH49tCSuPZTgn25HpfpH7499nzxMNWVpuoVpJBS1jhLZviuXzoWVfawyg2ySar9rRrDKkKai4Wr
        QJyChxuH1gDnGep4gSKZS75vAZPLcmG9BckgmFNZ6pAdvZ9wPeEibfDjSJd3iKD6MiHiiR1AKiLSMHH5W7rEng8kyZ1TL92w5dZ6vUtTK6rB7cZzhhfpBbra6iPvshr2nY3tz5fmbxTTFKWaFERHAdPczRhQonRmEXJAwoXMm2LqHrQuSCbfUAT4D
        6td7nJHMTh8pQ3sh2XVj7GtYjNNiADGx5bzwnCMFs6vP8ZFe4pabtuN82PQ9VAgBrLxjnZ6BaR9PYxoUZ7Co37F4fGBztn6L7hvuZSaXEYW7bchg1ztJfq9Dyr4RWvQW8jYoq5kxt1GU3W4sThwEQ9xkynz9uhaC5eaWHCTasfMvAei1CSpSffgnd
        iwg5fmgjftb1UavW8wjeduHoAeReANP4cmq8LxHpG4DRH8MHZF6FBrhVfC63xw1m6thqiVA6SmuAzYzwXuQUHfDv2BUKojovFnzNsPXBJfsv7rUTPt4UKocMvqqKYhxZLXtVAkdRzduBJr52i7RhgpQfU5MKrsHwEkg9663sJhgkjRo79MsQPisnx
        PmRLwgadYV86rFqCZDu32mZrMTHrmjeMAAQe6u4pjXUApQzYsZxMtBSNXv8ho9zMQ9YKkPkN8drhtUyNnJyuuBqhrDYuijvWPFj2yUy2eRyctVu5x4Y1EwgY4xGT2fFdGdkdXtivGCqAnsCNXUjv7RKjS8pFCiGVYCk1PxthAC2M6GgNc2xeYF338
        eiuyUmkZWXujg9wQcRMvXuTxwhcdvHsbBMvVbXw84wUQdBDW1nFyvrJyj6kP6vYh9gnPNFVfw66mUUbbeG3Zcs6pBuc5wtuD7ohwdmP4sUHzWQJg9tgYkEyykcCdDkhcw6uHDczjfyWXfmLhTKNu9nk7qhrTzFz2QU9RJvLm8QRa8mPb5mofGiJPz
        1ryweh6aV2XdorHzgz9ApSjPit6wABcCrE9hHmEcUTrRmUiyJiPPY38pqPrVEZcuhocR6VsQicYqZBrhiNAXDeFuTy38zfmLRy21wSCWJ5Cd87HtaKZDxrAe1Uw9TNRxQbjsnU5ucgZwbDthoQgYeKg66c6u8CjzYH",
      "type": "DataIntegrityProof",
      "verificationMethod": "did:dyne:b4dc0ff3"
    },
    "type": [
      "VerifiableCredential",
      "AlumniCredential"
    ],
    "validFrom": "2023-01-01T00:00:00Z"
  }
}
```


https://dyne.org/zencode-w3c-vc/

PQC in Zenroom

- Imported Dilithium2, Kyber512 (and NTRUP), ML-DSA, ML-KEM via PQClean in Zenroom libs (see: docs)
- Zencode (English-like DSL) interpreter in Lua
- End-to-end vector tests
- W3C-DID with Dilithium pubkey (https://did.dyne.org)
- W3C-VC 2.0 support (ready in Q2-25)
- Runs on Linux/Mac/Windows, Android/iOS and in Browser
- Extreme crypto agility: mix RSA/ECC + PQC in seconds 🔥

🔥 WANNA TRY IT AT HOME? 🔥



https://pqc.api.forkbomb.eu/docs

:&

PQC EUDIW

A.k.a.: PQC SD-JWT & mDL (via HSM)

Goal, requirements and Challenges

- Goal 1: produce and verify a QEAA and EAA, SD-JWT an mDL with a PQC signature
- Goal 2: produce a PID with a PQC signature SD-JWT an mDL
- Tech requirements:
  - Goal 1: Authorisation Servers/Issuers/Relying Party should support PQC and/or hybrid crypto → difficulty: Easy
  - Goal 2: Mobile Wallets should support PQC → difficulty: Medium or Nightmare

Warning, commercial ad: it’s already possible!





Goals, requirements and Challenges

- Challenges
  - What algorithms, do we trust NIST?



  - How do we manage HSM on server and mobile?
  - Standardization: PQC goes into SD-JWT & mDL

Experiment SD-JWT with ML-DSA-44

eyJhbGciOiAiRVMyNTYiLCAidHlwIjogInZjK3NkLWp3dCJ9.eyJfc2QiOiBbIkdyaW05dlFaUFBJaDlTejFyNnpvS1I4d3lNSzRWV3otcTE3b1
NUeTVrZ1kiLCAiWWMyQS1LUDhrRHN5bHAxeGlvUUQ3T0tYODUxMjk4Z0FPcXZ4NTNYYlpLVSIsICJtWGZOWjdXSEw0aUh2ZG1ham9WUjhyNUlVb
EVuUVhGUVgtOTN5UDZZUjlnIiwgIjZWOGIwZmhPTVFIa1N0U1R4R0t2eUpua0pMM1g2ZXVHeXpiWlBOR2hTNnciLCAiYThtNy1Sa3pBYndSdzk0
Q2w0TTF6MVhCM2Z4TUtRVTJaSE9sQXNubmxhayIsICJyQUgtZi1KbkxQdFdMdTNXVDZVaVVxTnNIdi03MVJIMTdHWTBqVkM2VmowIiwgIkdLUjU
ydllRUV9mLVVhLXBKZlR6ODl0dW1kb3B4Skd5UUE4VmhBTDgyUjAiXSwgIl9zZF9hbGciOiAic2hhLTI1NiIsICJpc3MiOiAiaHR0cDovL2V4YW
1wbGUub3JnIiwgInN1YiI6ICJ1c2VyIDQyIn0.WCvsZ2vQLderm9RiXI8HzjtLYSWd_NK-mjSoGvfaxLRQXlVfSQ8LLBJt3P1axASKvT8d8MCAT
Jbs0QiGUR1faw~WyI5OUV5OTlPTktyNkZkbkpUQXZLWEx3IiwgImdpdmVuX25hbWUiLCAiSm9obiJd~WyJzTC1pcjFOYU56enlUOFlWU3FZa1Rn
IiwgImZhbWlseV9uYW1lIiwgIkRvZSJd~WyIwcHhLS3NZSzdEVEJPWFNvTzVWZXFnIiwgImVtYWlsIiwgImpvaG5kb2VAZXhhbXBsZS5jb20iXQ
~WyJEbXM4M0RnUkZsb3Y2bUZVeFBsdi1BIiwgInBob25lX251bWJlciIsICIrMS0yMDItNTU1LTAxMDEiXQ~WyJGbWh4QjUxdnJuSFh2TG1naW9
sSThnIiwgInBob25lX251bWJlcl92ZXJpZmllZCIsIHRydWVd~WyJIYkw5Q1g3NmJ4RTdOQWhtY1Bibnd3IiwgImFkZHJlc3MiLCB7ImNvdW50c
nkiOiAiVVMiLCAibG9jYWxpdHkiOiAiQW55dG93biIsICJyZWdpb24iOiAiQW55c3RhdGUiLCAic3RyZWV0X2FkZHJlc3MiOiAiMTIzIE1haW4g
U3QifV0~WyIybXcyay1qV2Z5NmJSQ29nVXRYNVhRIiwgImJpcnRoZGF0ZSIsICIxOTQwLTAxLTAxIl0~

*AdES document signatures

(PAdES, CAdES, XAdES, JAdES)

In collaboration with



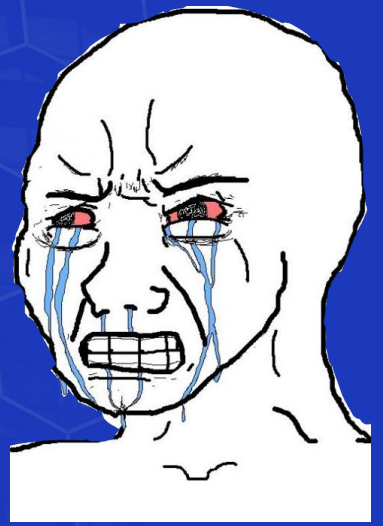
Goal, requirements and Challenges

- Goal: produce and verify a PDF (or other docs) signed with PQC
- Tech requirements:
  - CA (e.g. EJBCA) produces an X.509 with PQC sig and pubkey: Easy
  - Produce a PQC sig, matching the pubkey in the X.509: Easy
  - Embed the PQC in the PDF (e.g.: DSS): Easy
  - Clients (browsers, etc) verify PQC in X.509 and PDF: Medium

Goals, requirements and Challenges



- Challenges, wait for:
  - Standardized PQC/hybrid **X.509**
  - Standardized PQC/hybrid ***AdES**
  - Vendors to adopt it (Adobe, Microsoft, Google, Apple…)



:&

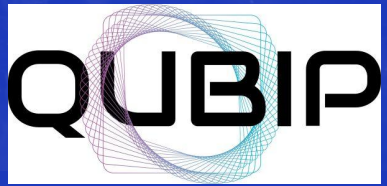
Standardization efforts

- *AdES (EU Standard): XAdES (ETSI 319 132-1), CAdES (ETSI 319 122-1), PAdES (ETSI 319 142-1) and JAdES (ETSI 119 182-1)
- PQC/hybrid X.509 (global!): see *truskovsky-lamps* and ETSI EN 319 412

PQC ZKP

Selective Disclosures + ZKP + PQC

In collaboration with



:&

QUBIP's scientific approach

- **Objective**: issuance and verification of verifiable credentials with selective disclosures and zero-knowledge proof
- After research, work based on "BLNS": "A Framework for Practical Anonymous Credentials from Lattices" (J. Bootle, V. Lyubashevsky, N. K. Nguyen, and A. Sorniotti) https://eprint.iacr.org/2023/560
  - First Analysis of "BLNS": https://zenodo.org/records/14848814

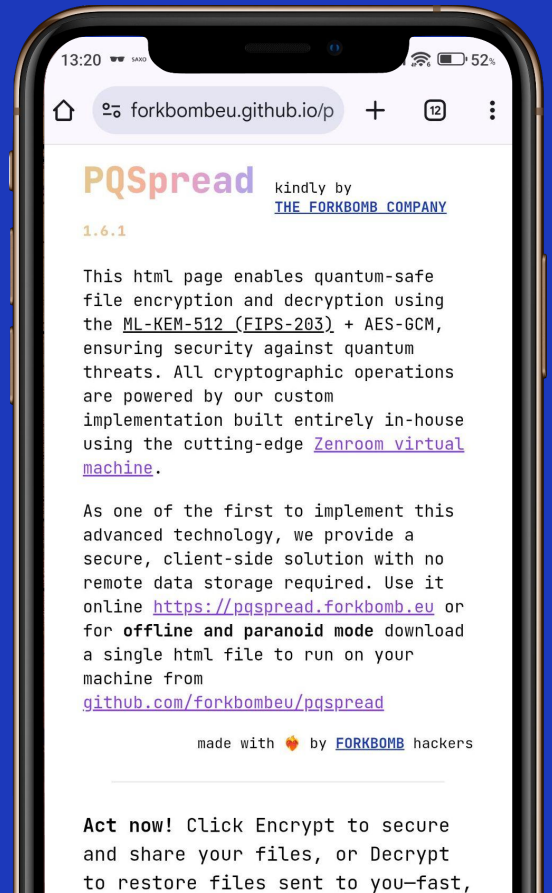
Descriptions, challenges, way forward

The math is still *very early days* → needs 5+ years of research



- BLNS offers unique three party model (issuer. holder, verifier)*
- The math is similar to Falcon (Lattice based)
- Math should be improved in multiple directions:
  - Efficiency: generation of secret keys (needs minutes!)
  - Security: while it's lattice based as Falcon uses floating point based math, is prone to time-based side channel attacks
- Road to standardization (10 years?): Qubip now uses NTL for math. Needs to re-implement all the math, in order to be standardized

Bonus: encrypt/decrypt docs with PQSpread



- ML-KEM + AES encryption
- In browser and offline
- Open Source
- Single HTML, serverless

https://forkbombeu.github.io/pqspread

Questions? andrea@forkbomb.eu



https://forkbomb.solutions/pqc

:&