PQC - Secure Distributed WSCA for EUDI wallets

Aivo Kalu, Petr Muzikant

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.

CHESS
Cyber-security Excellence Hub in Estonia and South Moravia

Co-funded by the European Union

Agenda

- Short introduction to digital identity, eIDAS2 and wallets
- Private key protection problem and achieving user's sole control
- Threshold cryptography with classical algorithms
- Threshold cryptography with PQC-secure ML-DSA
- Overview of EUDIW wallet architecture
- Certification of wallets, WSCAs and WSCDs
- Summary and takeaways

...based on a published paper of

Trilithium: Efficient and Universally Composable Distributed ML-DSA Signing

Antonín Dufka[1][0009−0003−5058−2571], Semjon Kravtšenko[1][0009−0007−8204−9519], Peeter Laud[1][0000−0002−9030−8142], and Nikita Snetkov[1,2][0000−0002−1414−2080]

[1] Cybernetica AS, Mäealuse 2/1, 12618 Tallinn, Estonia
{antonin.dufka, semjon.kravtsenko, peeter.laud, nikita.snetkov}@cyber.ee
[2] Tallinn University of Technology, Akadeemia tee 15a, 12618 Tallinn

Abstract. In this paper, we present Trilithium: a protocol for distributed key generation and signing compliant with FIPS 204 (ML-DSA). Our

https://eprint.iacr.org/2025/675.pdf

Very short company intro

- Privately owned R&D company focusing on information security, digital identity, e-government solutions
- Spin-off from Estonian Academy of Sciences, since 1997
- Main offices in Estonia (Tallinn and Tartu) and small presence in Czechia!
- 200+ employees, 18 nationalities
- 500+ patents and publications
- We are architects behind most e-Estonia government solutions

Digital identity intro

Digital identity and eIDAS1 (1)

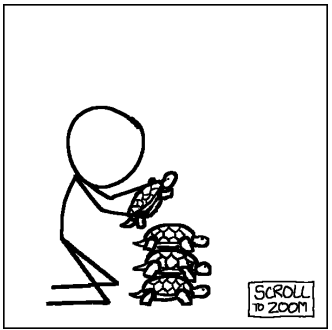
- Back in the old days, government issued digital identity has been:
  - Population registry and globally assigned identifier
  - Smart-cards with key pairs and X.509 certificates
  - Authentication (TLS-CCA), Digital Signatures (XAdES), Encryption/Decryption

Digital identity, eIDAS2 and wallets (2)

- eIDAS2 updated this approach for EU member states on May 20, 2024:
  - Population registries (authoritative data sources), biometric identity proofing and VC issuers
  - Wallets with key pairs and verifiable credentials (mdocs, VCs, QEAAs)
  - Authentication (PID VC presentation), Attribute presentations (e.g. age-over-18), Digital Signatures (PAdES), ...

Trend towards automated private key management

- 47-day SSL/TLS certificates with CertBot
- Batch VCs and updating VCs, based on OAuth2 tokens
- Attestations and proof-of-possession
- SPIFFE and SPIRE
- (Private Access Tokens, The Privacy Pass HTTP Authentication Scheme (RFC9577))



https://xkcd.com/1416

Underlying private keys and secrets protection problem

- So, you have a PKI/SSI system with millions of key pairs.
- Where do users keep their private keys?

Underlying private keys and secrets protection problem

- So, you have a PKI/SSI system with millions of key pairs.
- Where do users keep their private keys?
- Industry best practice so far:
  - Traditional approach – JavaCard, PKCS#11, USB/NFC

Underlying private keys and secrets protection problem

- So, you have a PKI/SSI system with millions of key pairs.
- Where do users keep their private keys?
- Industry best practice so far:
  - Traditional approach – JavaCard, PKCS#11, USB/NFC
  - Mobile platform services – Android StrongBox, IOS KeyChain and Secure Element

Underlying private keys and secrets protection problem

- So, you have a PKI/SSI system with millions of key pairs.
- Where do users keep their private keys?
- Industry best practice so far:
  - Traditional approach – JavaCard, PKCS#11, USB/NFC
  - Mobile platform services – Android StrongBox, IOS KeyChain and Secure Element
  - GP/GSMA solution – SIM-card applet, eSIM applet

Underlying private keys and secrets protection problem

- So, you have a PKI/SSI system with millions of key pairs.
- Where do users keep their private keys?
- Industry best practice so far:
  - Traditional approach – JavaCard, PKCS#11, USB/NFC
  - Mobile platform services – Android StrongBox, IOS KeyChain and Secure Element
  - GP/GSMA solution – SIM-card applet, eSIM applet
  - remote HSM in service provider's environment

Underlying private keys and secrets protection problem

- So, you have a PKI/SSI system with millions of key pairs.
- Where do users keep their private keys?
- Industry best practice so far:
  - Traditional approach – JavaCard, PKCS#11, USB/NFC
  - Mobile platform services – Android StrongBox, IOS KeyChain and Secure Element
  - GP/GSMA solution – SIM-card applet, eSIM applet
  - remote HSM in service provider's environment
  - or ...

Threshold cryptosystem

- Splitting private key material into shares and assigning shares to different people
- Everybody must come together to perform key operations
- Nobody can act on behalf of the whole group

Threshold cryptosystems

- Shamir's Secret Sharing with symmetric secrets since 1979
- public-key cryptography research since 1994
- NIST held first standardisation workshop on 2019
- FROST Protocol for Two-Round Schnorr Signatures (RFC9591, 2024)

Digital identity system with threshold cryptography

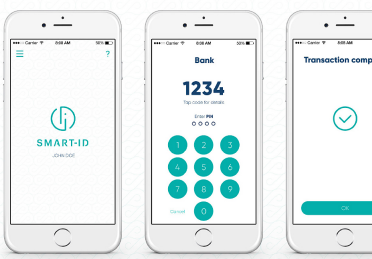
- User has a knowledge factor or biometric factor
- User's mobile device has a share of user's private key
- Service provider has a share of user's private key
- In order to create a signature:
  1. User has to know the knowledge factor
  2. User has to decrypt local share of user's private key with key derived from PIN
  3. User has to access server-side share of user's private key

Emerging security properties

- Off-line bruteforce – attacker, who has user's share of private key, cannot use it without a PIN
- On-line bruteforce – attacker, who tries to guess PIN-code has limited tries
- Clone detection – attacker's copy user's share of private key can be detected
- User's sole control – user always has to participate in the signature creation
- Limited power of service provider - they cannot create fraudulent signatures on user's behalf

Smart-ID – based on threshold crypto

- Commercial eIDAS1 eID scheme provided in EE, LT, LV, BE, (IS)
- Run by SK ID Solutions AS, since 2017
- 3.7 M users, 2-3 M transactions/day
- eIDAS QSCD (CC EAL4+ certified)
- eIDAS "high"-level scheme (EE, LV, LT, BE evaluated)
- Smart-ID system uses Cybernetica's SplitKey technology



SplitKey - RSA key generation

- User generates RSA private key (d_1, n_1) and splits d_1

$$d_1 = d'_1 + d''_1 \ mod \ \varphi(n_1)$$

- User encrypts d'_1 with encryption key derived from PIN and stores locally
- User transfers d''_1 to server
- Server generates RSA private key (d_2, n_2) and computes composite public key $n = n_1 \cdot n_2$
- There are three shares of private key material:

$$d'_1, d''_1, d_2$$

SplitKey - RSA signing

- User enters the PIN code and decrypts local share d_1''
- User computes $\sigma_0 = m^{d_1'} \ mod \ n_1$ and sends to server
- Server computes $\sigma_1 = \sigma_0 \cdot m^{d_1''} \ mod \ n_1$
- Server verifies that σ_1 is a correct signature
- Server computes $\sigma_2 = m^{d_1'} \ mod \ n_2$
- Server computes $s = CRT_{n_1,n_2}(\sigma_1, \sigma_2)$
- Signature s is verifiable by the composite public key $n = n_1 \cdot n_2$

Post-Quantum-Cryptography

Do you know that you are probably already using PQC algorithms every day?

Do you know that you are probably already using PQC algorithms every day?

Chrome + Cloudflare, Signal, Apple Messages, ...

Post Quantum Cryptography migration

- NIST has published standards, such as FIPS 204 - Module-Lattice-Based Digital Signature Standard, August 2024
- NSA has mandated to prefer PQC algorithms for signing by 2025 and exclusive use by 2030.
- NIST has published IR 8547 - "Transition to Post-Quantum Cryptography Standards" on November 2024.
- EU "Agreed Cryptographic Mechanisms, v2" from April 2025 include PQC
- EU has a "Coordinated Implementation Roadmap" by 31.12.2026, "First Steps" should be implemented by member states

September 2025 Heatmap: Current State of PQC Standards and Adoption

Standard	Overall	Pure PQC encrypt	Hybrid PQ encrypt	Pure PQ sig	Hybrid PQ sig
SSH	3 to 8	3	8	3	3
TLS 1.3	3 to 9	7	9		3
X.509	4 to 7	7	4	7	4
S/MIME	3 to 7	5	3	7	3
IKE/IPSec	2 to 8	8	8	3	2
MLS	2 to 4	4	4	4	2
DNSSec	1 to 1	-	-	1	1

https://pqcc.org/learn-more/state-of-the-migration/

PQC - secure ML - DSA (1, intro)

- It's a Schnorr - like signature system:
  - private key x and public key $y = g^x$
  - Signing:
    1. choose random k and compute commitment $r = g^k$
    2. compute challenge $e = H(r||M)$
    3. compute response $s = k + x \cdot e$ and output signature (s, e)

PQC - secure ML - DSA (1, intro)

- It's a Schnorr - like signature system:
  - private key x and public key $y = g^x$
  - Signing:
    1. choose random k and compute commitment $r = g^k$
    2. compute challenge $e = H(r||M)$
    3. compute response $s = k + x \cdot e$ and output signature (s, e)
- but it's not working on natural numbers like 1 or 2 or 78432784239784932

PQC - secure ML - DSA (1, intro)

- It's a Schnorr - like signature system:
  - private key x and public key $y = g^x$
  - Signing:
    1. choose random k and compute commitment $r = g^k$
    2. compute challenge $e = H(r||M)$
    3. compute response $s = k + x \cdot e$ and output signature (s, e)
- but it's not working on natural numbers like 1 or 2 or 78432784239784932
- computations are made on 256 - degree polynomials, like $p(x) = 123456 + 8374120x + 17x^2 + 998877x^3 + \ldots + 54321x^{255}$
- which are organised into vectors or $k \times l$ matrixes (like 4×4 or 6×5 or 8×7)

PQC-secure ML-DSA (2, key generation)

- public matrix $\mathbf{A} \leftarrow \left[\mathbb{Z}_q[x]/(x^{256}+1)\right]^{k \times l}$
- short secret vector $(\mathbf{s_1}, \mathbf{s_2}) \leftarrow S_\eta^l \times S_\eta^k$
- public key vector $\mathbf{t} = \mathbf{A} \cdot \mathbf{s_1} + \mathbf{s_2}$
- ML-DSA public key $(\mathbf{A}, \mathbf{t})$

PQC - secure ML - DSA (3, signing)

- generate ephemeral secret vector $\mathbf{y} \leftarrow S^{l}_{\gamma_1 - 1}$
- compute commitment vector $\omega = \mathbf{A} \cdot \mathbf{y}$
- compute challenge $c = H(M || \omega^{H})$
- compute response with secret key vectors $\mathbf{z} = \mathbf{y} + c \cdot \mathbf{s}_1$
-
- (some details omitted ...)
-
- test if response leaks too much of secret key vectors and restart if needed
- output signature $\sigma = (\mathbf{z}, c)$

Threhold crypto with ML-DSA

Multi-Party Computations (MPC) on additively shared values

- Additive sharing – we take a value v and share it between parties:

$$[\![v]\!]_C, [\![v]\!]_S$$

$$v = [\![v]\!]_C + [\![v]\!]_S \ mod \ q$$

- It turns out that we can make computations with them, without knowing value of individual shares
- Constant introduction, addition of shares, multiplication of shares, ...
- and we can implement more complex algorithms and protocols ...
- and we can implement cryptographic key generation and signing algorithms

Addition. How?

- $[\![a]\!] + [\![b]\!]$?

Addition. How?

- $[\![a]\!] + [\![b]\!]$?
- Client has $[\![a]\!]_C$ and $[\![b]\!]_C$
- Server has $[\![a]\!]_S$ and $[\![b]\!]_S$

Addition. How?

- $[\![a]\!] + [\![b]\!]$?
- Client has $[\![a]\!]_C$ and $[\![b]\!]_C$
- Server has $[\![a]\!]_S$ and $[\![b]\!]_S$
- Client and Server can compute:

$$[\![a+b]\!] = [\![a+b]\!]_C + [\![a+b]\!]_S =$$

$$= [\![a]\!]_C + [\![b]\!]_C + [\![a]\!]_S + [\![b]\!]_S = [\![a]\!] + [\![b]\!]$$

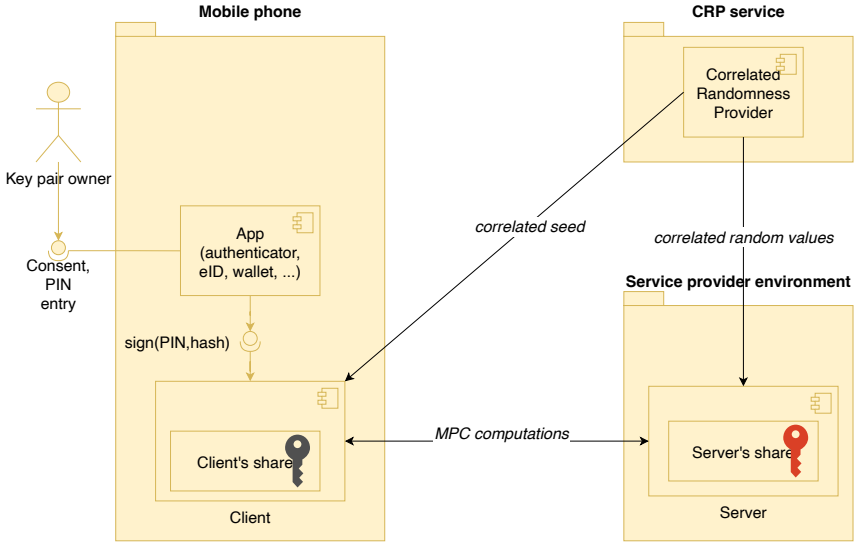
Protecting against malicious parties

- But we need additional security for such non-linear MPC algorithms/protocols, which involve round-trips
- In case one party uses maliciously modified values, they might learn from the result, what the other party's confidential value was
- To counter, we use BeDOZa-style[1] message-authentication-codes
- So, whenever client or server need to send it's share of $[\![v]\!]$ to each other, they must accompany it with a control value (MAC) $M_S = [\![v]\!]_C \cdot \Delta_S$,
- Parties can verify in a declassify protocol that both have acted honestly

[1] Bendlin, Damgård, Orlandi, Zakarias: Semi-homomorphic encryption and multiparty computation. EUROCRYPT 2011

Correlated Randomness Provider

- For generating such Δ values for client and server we use a (somewhat) trusted third party – CRP component
- CRP component doesn't need to store state and basically, just generates correlated random values
- client (the phone) receives a seed from CRP and generate random values form the seed
- server receives individual MAC values from CRP


Mobile phone
CRP service
Correlated Randomness Provider
Key pair owner
App (authenticator, eID, wallet, ...)
Consent, PIN entry
correlated seed
correlated random values
Service provider environment
sign(PIN,hash)
MPC computations
Client's share
Server's share
Client
Server

Key generation performance of SplitKey-PQC (initial results)

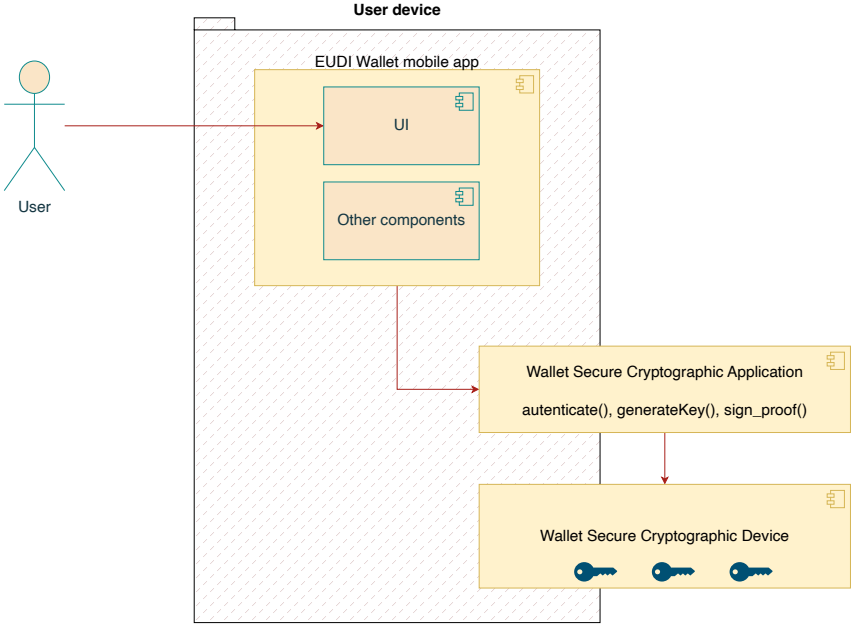
ML-DSA category	key generation	amount of transmitted data		
		C → S	S → C	CRP → S
ML-DSA-44	448 ms	460 KB	460 KB	1.42 MB
ML-DSA-65	716 ms	880 KB	880 KB	2.6 MB
ML-DSA-87	598 ms	1850 KB	1850 KB	4.4 MB

Signing performance of SplitKey-PQC (initial results)

ML-DSA category	expected signing time	expected amount of transmitted data		
		C → S	S → C	CRP → S
ML-DSA-44	2516 ms	395 KB	395 KB	233 MB
ML-DSA-65	3886 ms	622 KB	622 KB	362 MB
ML-DSA-87	4975 ms	631 KB	631 KB	365 MB

- We are running 3 parallel signing iterations, with simulated 30 ms latency between the client and server

Applying threshold crypto to EUDI Wallet

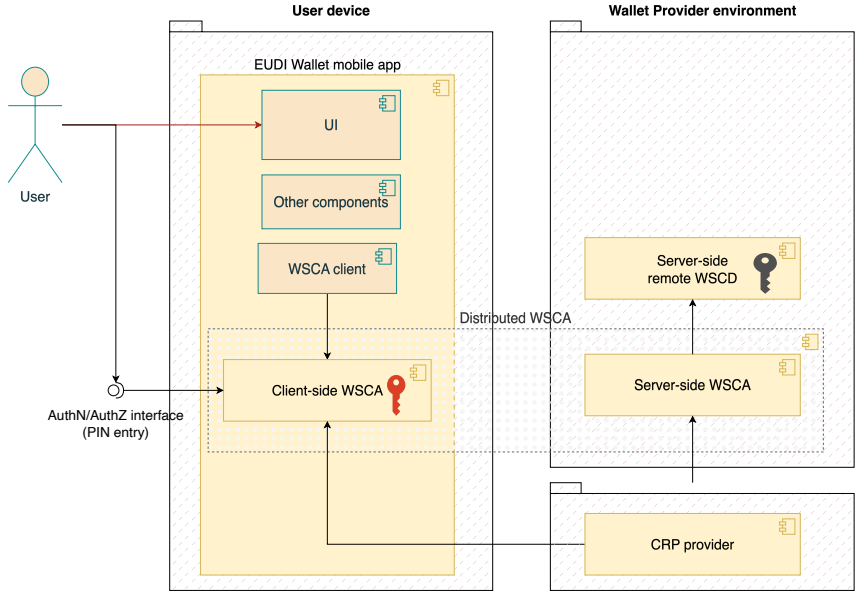

User device
EUDI Wallet mobile app
UI
Other components
User
Wallet Secure Cryptographic Application
autenticate(), generateKey(), sign_proof()
Wallet Secure Cryptographic Device

WSCA/WSCD architecture options for EUDIW

- eIDAS2, implementation acts, and ARF (section 4.5) have defined following options:
  - local native WSCD – component embedded in user's device and accessed over operating system API
  - local internal WSCD – component within user's device (SIM, e-SIM, embedded secure element)
  - local external WSCD – external device, such as smart-card
  - remote WSCD – remote device, such as HSM, accessed over network

SplitKey - PQC applied to WSCA/WSCD

- We need following functions
-
- `session=authenticate(knowledge_factor)`
- `handle=generateKey()`
- `proof=sign_proof(hash)`


User device
Wallet Provider environment
EUDI Wallet mobile app
UI
Other components
WSCA client
Client-side WSCA
User
AuthN/AuthZ interface
(PIN entry)
Distributed WSCA
Server-side
remote WSCD
Server-side WSCA
CRP provider

Wallet certification requirements

- eIDAS2 Article 5c and CIR 2024/2981 require that certification of wallets, in accordance of functional, cybersecurity and data protection requirements, should ensure high level of trust
- Wallet solutions should reach assurance level "high"
- Member states must come with a national wallet certification schemes
- ENISA AHWG is currently preparing a pan-EU wallet certification scheme

Certification pathways

- WSCA/WSCD should be certified according to CC/EUCC, at EAL4 assurance level with AVA_VAN.5 vulnerability analysis
- EUCC CIR 2024/482 Annex II refers to following existing standards:
  1. CEN EN 419241-2:2019 - "Protection Profile for QSCD for Server Signing"
  2. CEN EN 419221-5:2018 - "Protection Profiles for TSP Cryptographic Modules"

Certification pathways

- WSCA/WSCD should be certified according to CC/EUCC, at EAL4 assurance level with AVA_VAN.5 vulnerability analysis
- EUCC CIR 2024/482 Annex II refers to following existing standards:
  1. CEN EN 419241-2:2019 - "Protection Profile for QSCD for Server Signing"
  2. CEN EN 419221-5:2018 - "Protection Profiles for TSP Cryptographic Modules"
- SplitKey-RSA solution has been already evaluated according to 419241-2 (2018, 2024)

Certification pathways

- WSCA/WSCD should be certified according to CC/EUCC, at EAL4 assurance level with AVA_VAN.5 vulnerability analysis
- EUCC CIR 2024/482 Annex II refers to following existing standards:
  1. CEN EN 419241-2:2019 - "Protection Profile for QSCD for Server Signing"
  2. CEN EN 419221-5:2018 - "Protection Profiles for TSP Cryptographic Modules"
- SplitKey-RSA solution has been already evaluated according to 419241-2 (2018, 2024)
- CEN/CENELEC TC224 WG17 is currently developing a new PP for WSCAs

Summary – key takeways

- Protecting secrets inside user's applications is pretty difficult
- Threshold cryptography can be useful and there are examples in production (Smart-ID, SplitKey, ...)
- Threshold PQC research is active and ongoing
- We now have a first prototype for ML-DSA compatible distributed WSCA
- Open to collaboration within ETSI and CEN/CENELEC working groups!

Thank you!

Questions?

- Aivo Kalu, aivo.kalu@cyber.ee
- Petr Muzikant, petr.muzikant@cyber.ee

cybernetica
Cybernetica
cybernetica_ee
Cybernetica