PROMETHEUS

PRivacy preserving pOst-quantuM systEms from advanced crypTograpHic mEchanisms Using latticeS

E-Voting Use Case

- PROMETHEUS Industrial Workshop -

Presenter: Aleix Amill
Scytl Election Technologies S.L.U.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 780701.

Table of contents

Table of contents

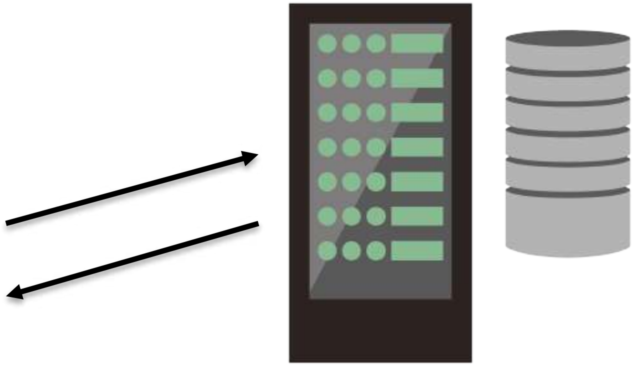
Online voting system

- I-voting advantages
  - Accurate and fast vote counts
  - Reduction of logistic cost of an election organization
  - Voters with disabilities can cast their votes independently
  - Abroad voting improvements

- Privacy is one of the main requirements
  - Encryption ensures votes confidentiality to the voters
  - Mixing (random permutation + re-encryption) ensures anonymity
  - Currently, the security of those processes relies on mathematical problems

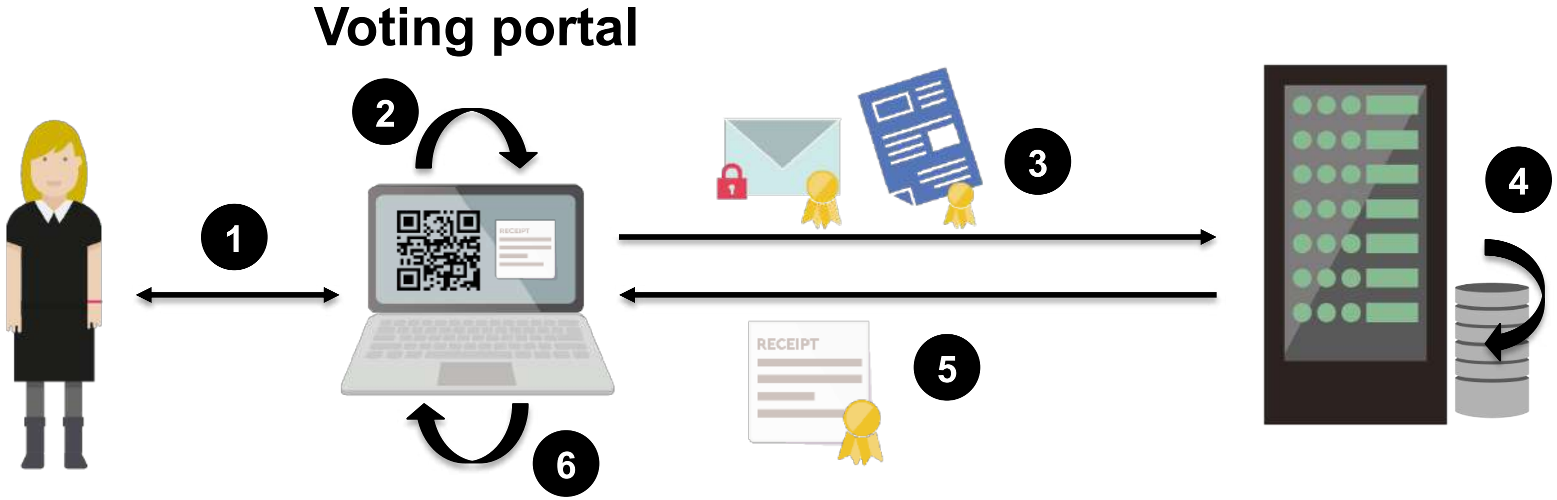
Configure election

Backoffice portal



1. Create and configure election
2. Create and assign voters
3. Generate key pair
4. Publish and Close election
5. Monitor election
6. Tally election (sub-processes)

Cast a vote



1 Select voting options

2 Prepare a ballot and a QR code for the cast-as-intended verification

3 Send vote

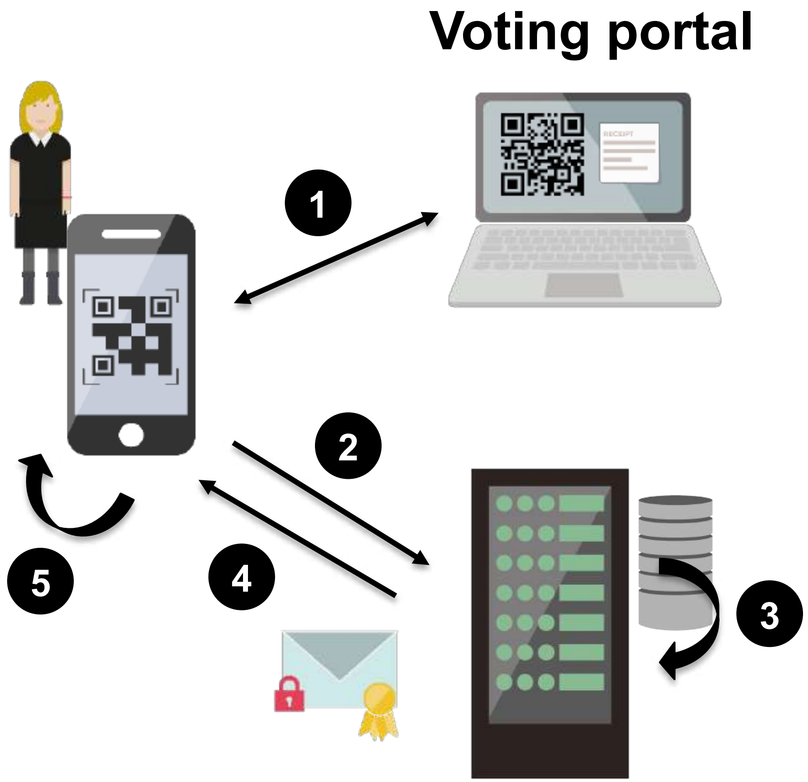
4 Validate vote, store it and generate receipt

5 Send receipt

6 Validate receipt signature. Show receipt and QR code

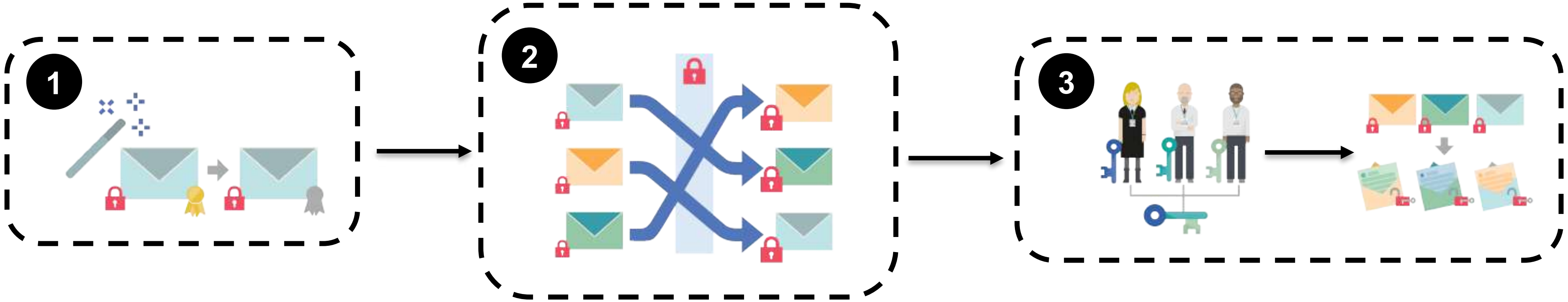
Verify a vote

Cast-as-Intended verification



1. Read QR code using a second device
2. Ask for the encrypted vote to the voting server
3. Retrieve the encrypted vote corresponding to the voter
4. Send the encrypted vote
5. Decrypt the vote using the information stored in the QR. Show the voting options to the voter

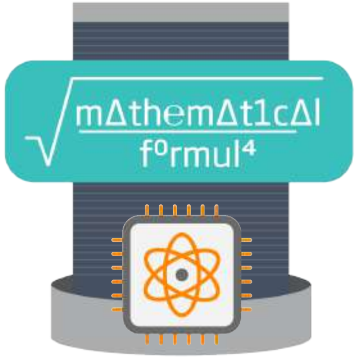
Tally process



1. Cleanse ballot box: remove all voter-related information, keep only ciphertexts.
2. Mix ballot box and generate proof of a correct shuffle.
3. Decrypt votes and generate proof of correct decryption.

Why PQ for online voting systems?

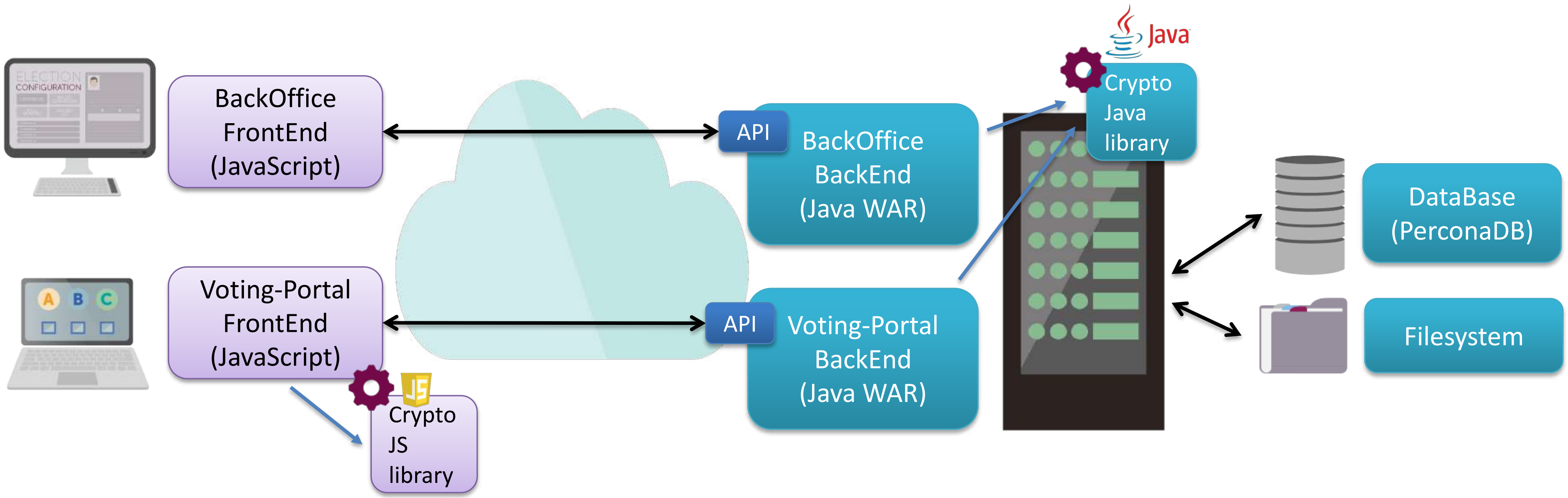
- Factorization and discrete logarithm will be easily solved by quantum computers
  - Encryption and mixing are not secure in the long term
  - Thus, the current state-of-the-art e-voting systems do not guarantee long-term privacy



Post-Quantum e-voting systems guarantee the long term privacy of the voters

Table of contents

E-voting system components



PQ primitives

Cryptographic lattice-based primitives:

- RLWE encryption scheme
- FALCON Signature scheme
- Decryption proof
- Shuffle proof

- Commitment scheme
- ZKPs for proving polynomial relation between committed messages
- Amortized proof of knowledge for secret small elements
  - Preimages perfect Proofs
  - Preimages imperfect Proofs

System libraries

Application layer	• APIs an interfaces for the user interaction
Voting library	• High level services to setup the election, vote, mix and count.
Crypto library	• Cryptographic post-quantum operations required by the services of the above layer
Math library	• Mathematical operations required by the cryptographic layer

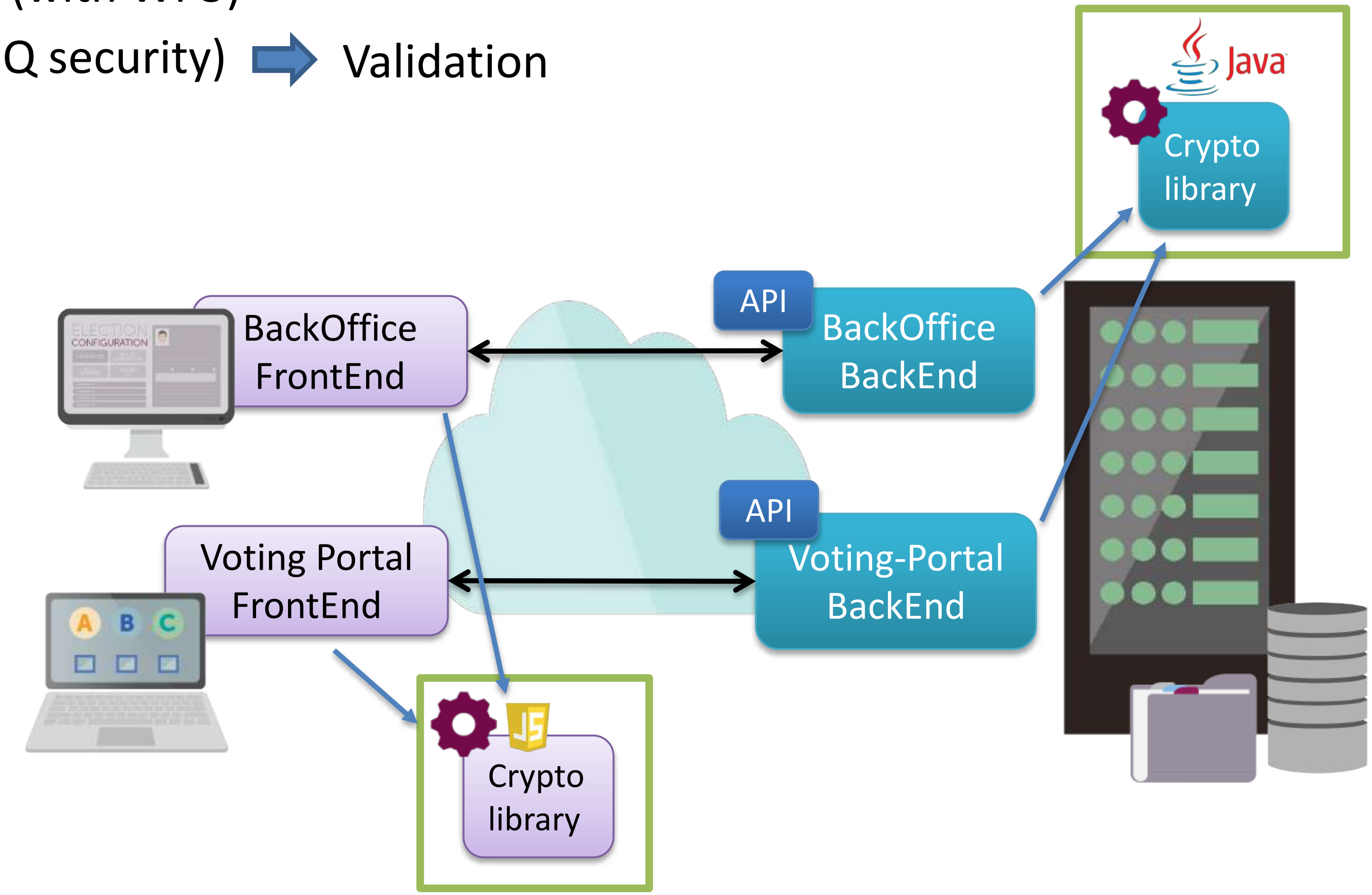
Table of contents

Implementation story

1. US + Demonstrator implementation without crypto (not secure)
2. Cryptographic libraries implementation (with WP5)
3. Demonstrator crypto integration (add PQ security) ⇨ Validation

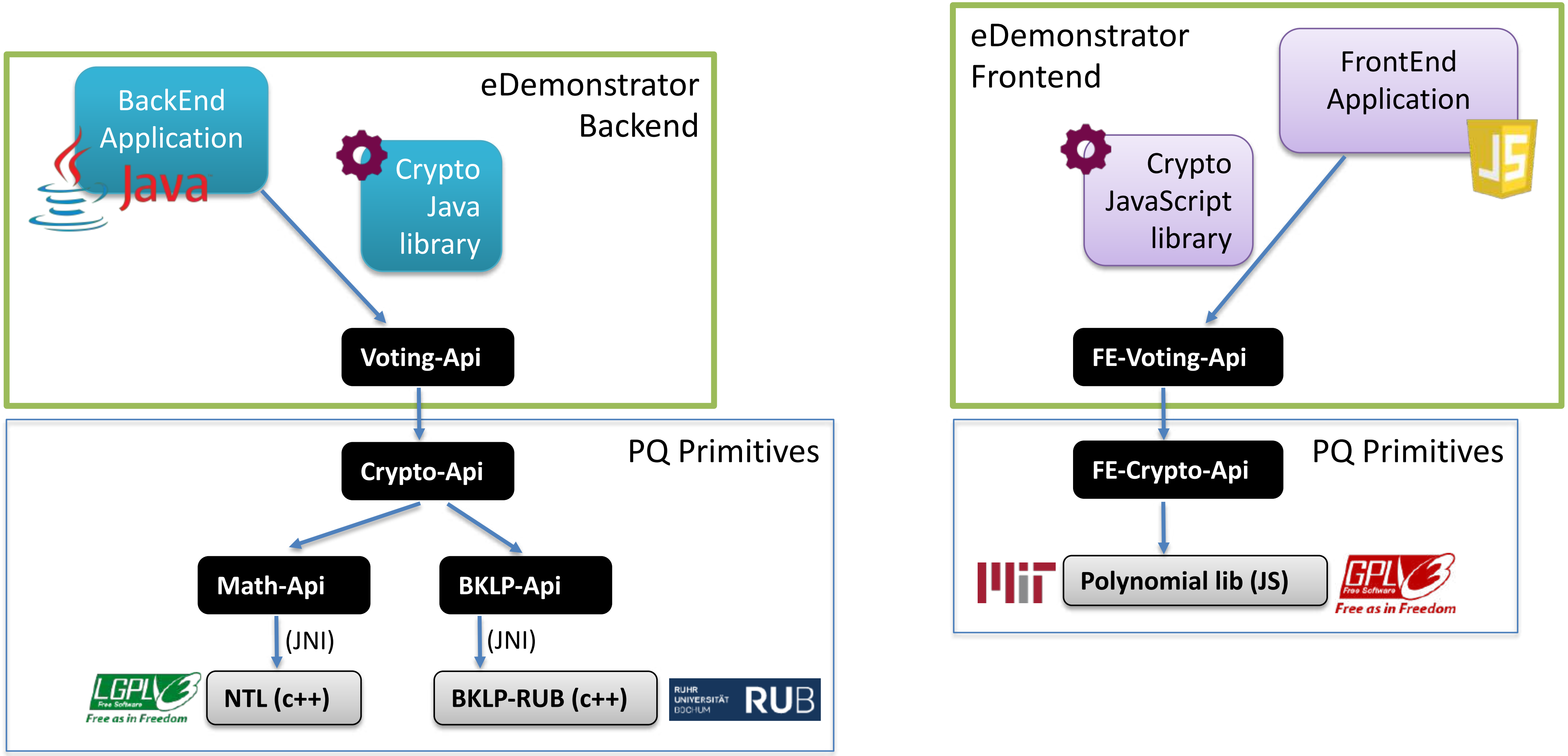
Prioritization in the implementation:

- Vote encryption
- Mixnet and shuffle proofs requirements:
  - commitments, zkps, preimages
- no signatures
- no decryption proofs



PROMETHEUS

E-voting system libraries



Shuffle proof primitive implementation

Shuffle proof sub-processes:

- Commit to **encryptions of zeros**
- Compute Encryptions of zeros **Preimage Proof**
- Commit to **permutation**
- Use Fiat-Shamir to get the first **challenge**
- Commit to the first challenge raised to permutation
- Use Fiat-Shamir to get the second and the third challenges
- Compute **Product Proof**
- Compute **Multi-exponentiation Proofs (x2)**

Limitations in the implementation

- Lack of lattice-based libraries implemented on client-side browser-oriented language (JavaScript)
- Challenging task to estimate secure parameters
  - all primitives schemas parameters must comply with some relations constraints
- Parameters size constraints in third-party libraries
  - Polynomial JS library 53-bits modulus constraint
- High processes memory resources requirements
  - heap and stack memory minimum size
- Coexisting native code executions in the JVM with parallelization
  - segmentation faults errors

Table of contents

PQ E-voting system DEMO

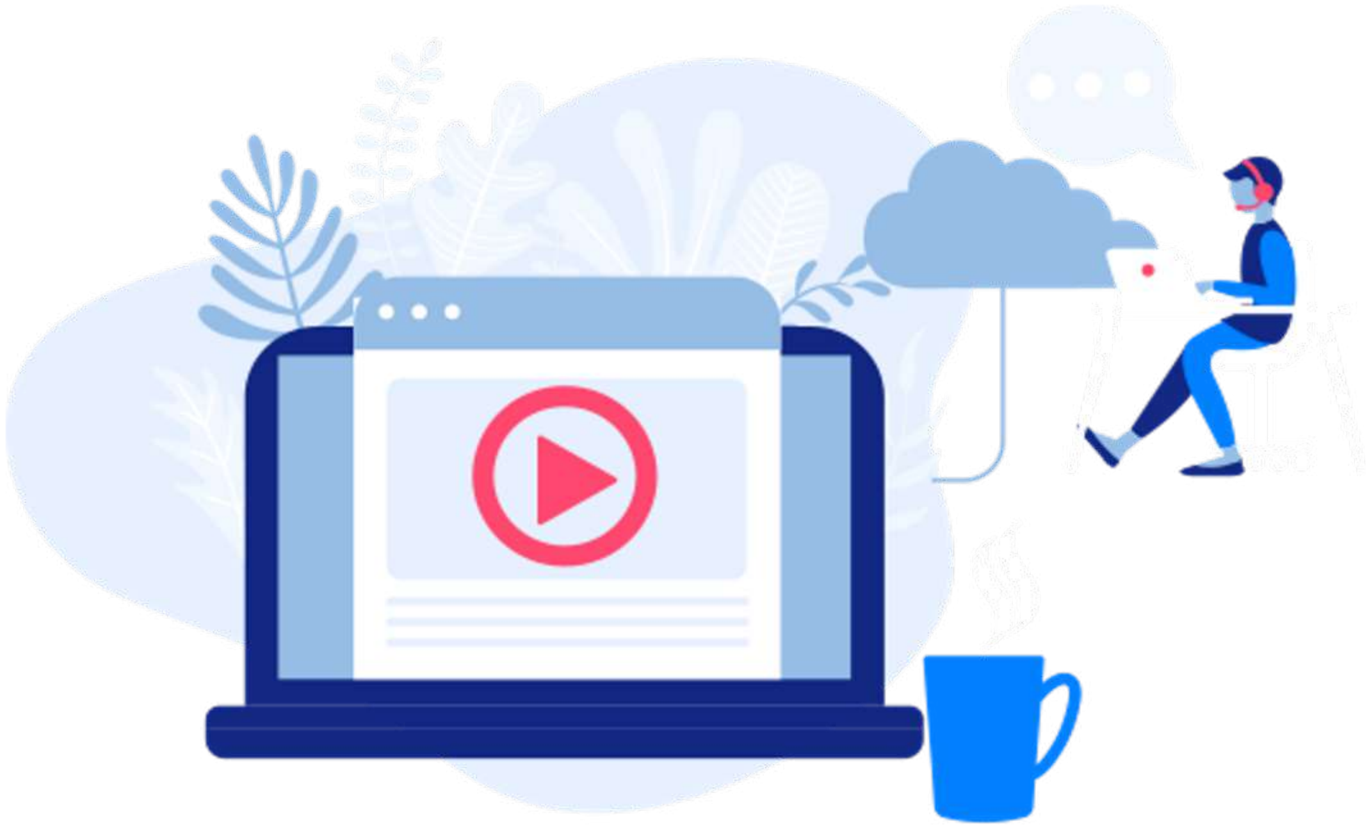


Table of contents

- Introduction
  - Online voting system
  - Why PQ for online voting system?
- E-voting system demonstrator design
  - PQ modules
  - System components & libraries
- E-voting system demonstrator implementation
  - Implementation story
  - Limitations in the implementation
- Demo
- System validation & benchmarking
- Conclusions

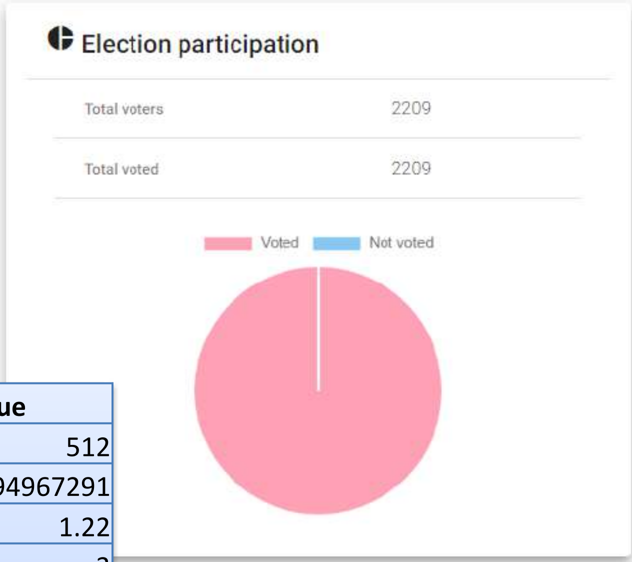
PQ System performance validation

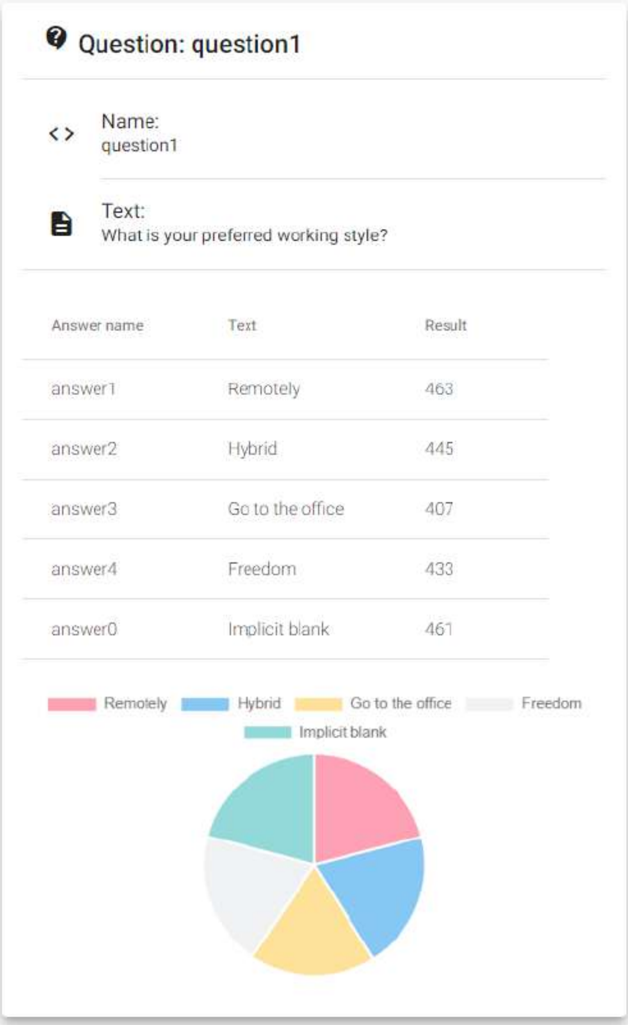
- Environment:
  - x86-64 server
  - 16 vCPUs
  - 16gb RAM

- Parameters:

Context	Parameter	Description	Value
Polynomial ring params	n	degree	512
	q	modulus (=3 mod 8, >=n^gamma)	4294967291
LPR params	sigma	gaussian standard deviation	1.22
	eta	binomial	3
BKLP params	kappa	bounds the knowledge error	21
	gamma	controls size of the modulus	3
	k	multiplicative overhead	9
	cbd eta err	cbd eta of error	1
	cbd eta rand	cbd eta of randomness	1
dPL params	r	integer r	128
	alpha	integer alpha	64
CDXY params	gss-k	good set system k param	23
	gss-m	good set system M' size (number of baskets)	2209
	beta	upper bound on the witnesses	8
Shuffle params	N'	shuffle batch size	2209

Election results





System benchmarking

- Compare PQ e-voting demonstrator performance vs. current Scytl's e-voting system product: invote
- Use similar environments and amount of inputs

Preference

Start date: 2022-05-26 16:00 (CEST)
End date: 2022-06-26 16:00 (CEST)

Counting date: 2022-05-30 11:29 (CEST)

1. What is your preferred working style? — **2208 Votes**

Options	Votes
Remotely	543
Hybrid	558
Go to the office	550
Freedom	557
Implicit blank	0

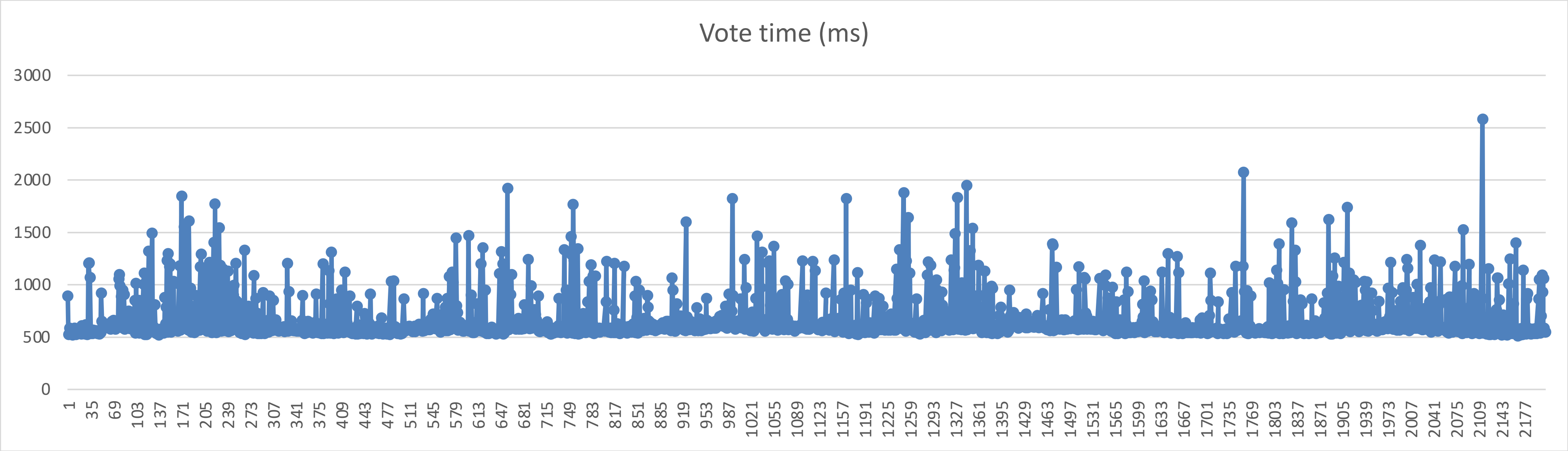
PQ system Client-side performance

PQ Client-side performance times

- Average voting time: 662 milliseconds
- Minimum voting time: 510 milliseconds
- Maximum voting time: 2581 milliseconds

invote performance

- Average voting time: 2559 milliseconds
- Minimum voting time: 1914 milliseconds
- Maximum voting time: 4328 milliseconds



PQ system Server-side performance

PQ Server-side performance times

- Ballot box cleansing: 8,3 seconds
- Ballot box verifiable mixing: **10938** seconds
- Mixing proofs validation: **10618** seconds
- Ballot box decryption: 3,1 seconds
- Ballot box tally: 1,5 seconds

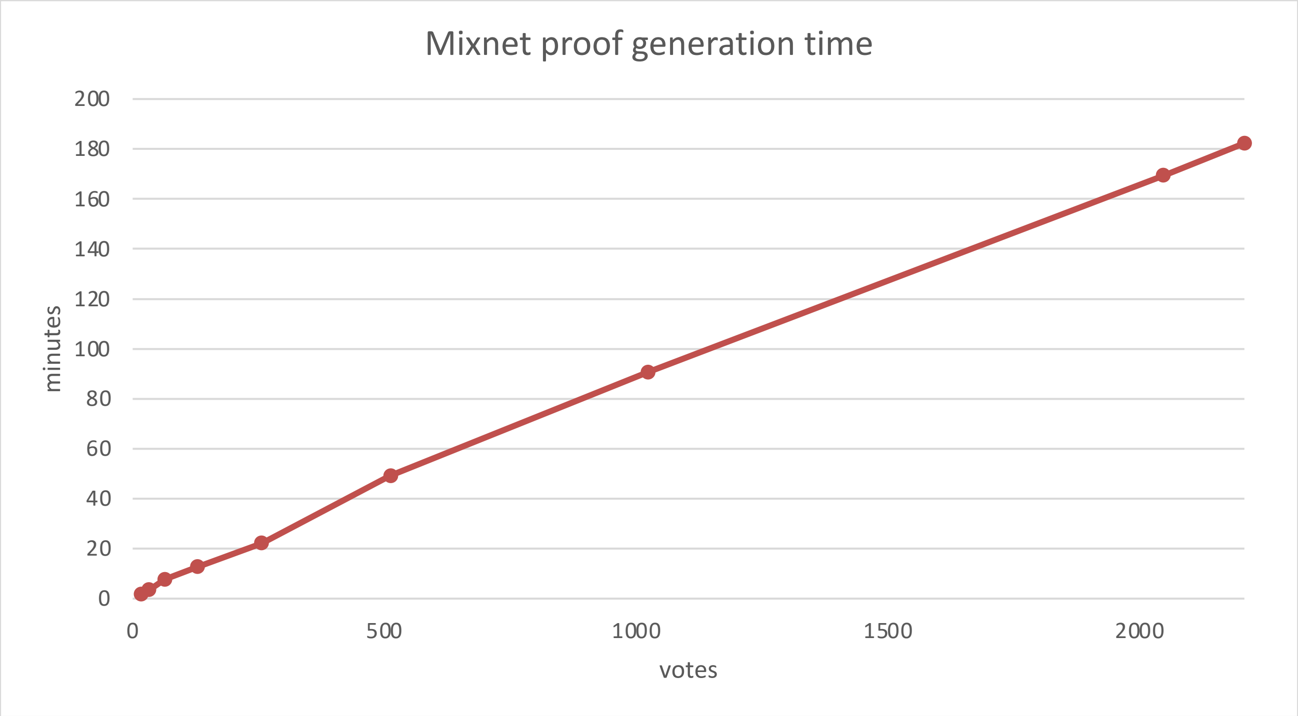
~3 hours (~2k votes)

invote performance

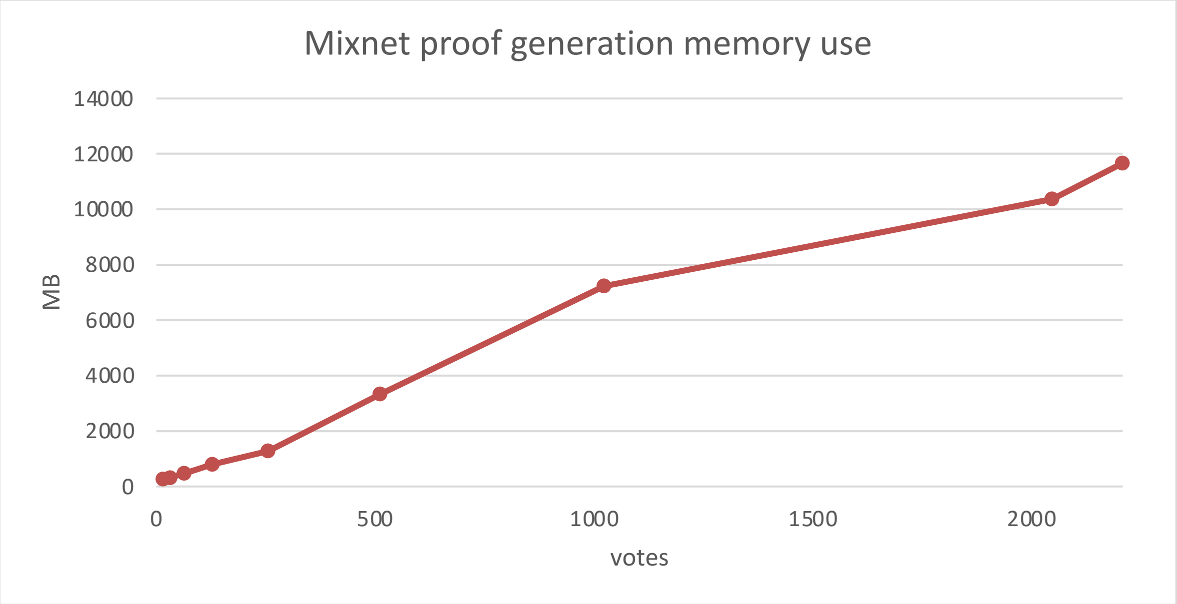
- Ballot box cleansing: 55 seconds
- Ballot box verifiable mixing: 43 seconds
- Mixing proofs validation: 271 seconds
- Ballot box decryption & tally: 20 seconds

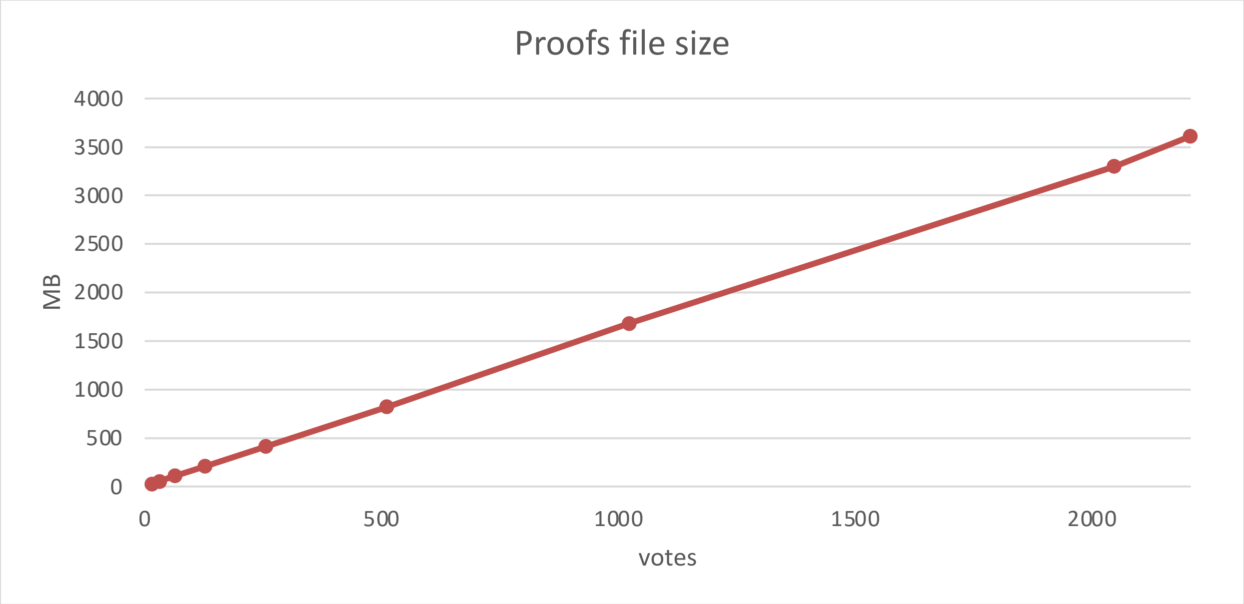
Mixnet in-deep performance analyses

- Ballot box sizes: 16, 32, 64, 128, 256, 512, 1024, 2209



Mixnet in-deep performance analyses





- Up to 12GB memory use for 2k votes
- Up to 3.6GB Proofs file size for 2k votes

Shuffle proof inner timing analysis

Sub-processes:

- Commit to encryptions of zeros
- **Compute Encryptions of Zeros Preimage proof**
- Commit to permutation
- Use Fiat-Shamir to get the first challenge
- Commit to the first challenge raised to permutation
- Use Fiat-Shamir to get the second and the third challenges
- **Compute Product proof**
- **Compute Multi-exponentiation proofs (x2)**

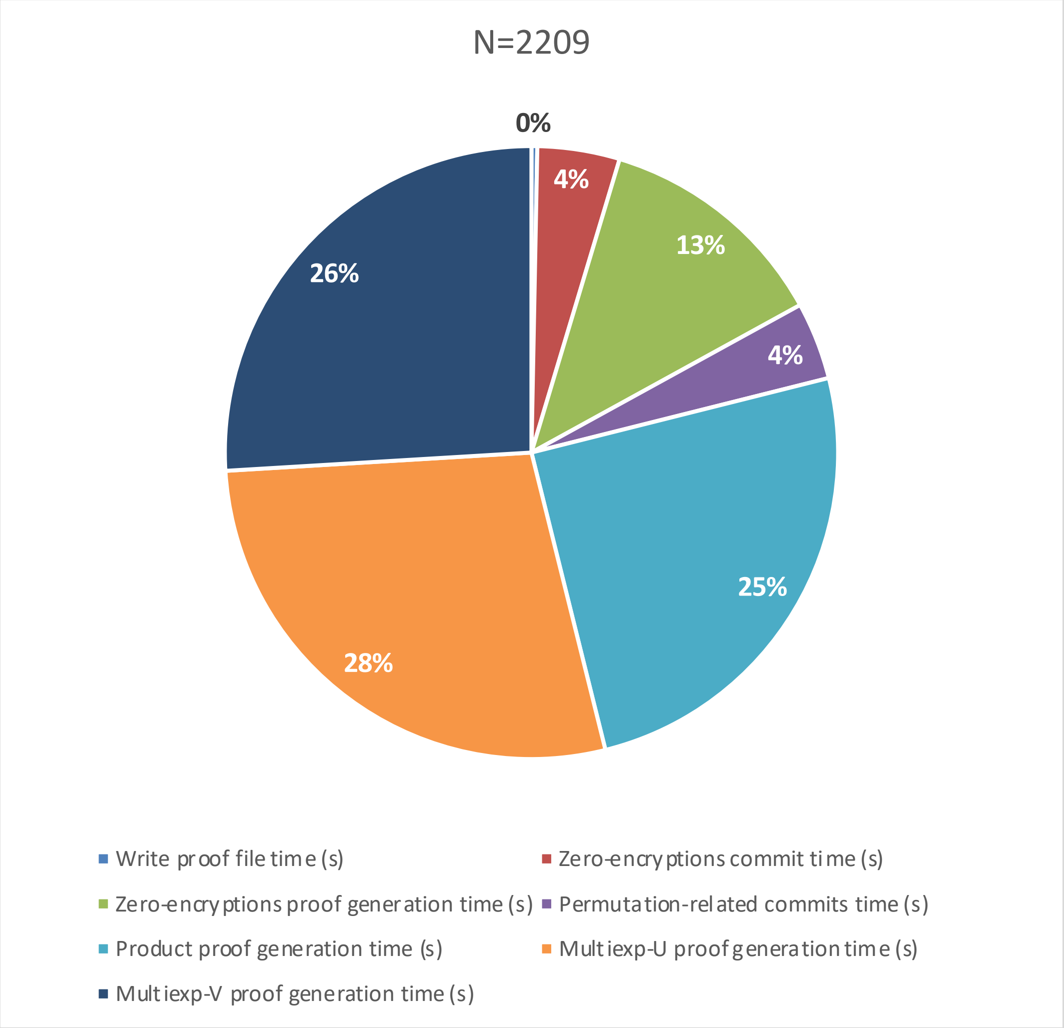
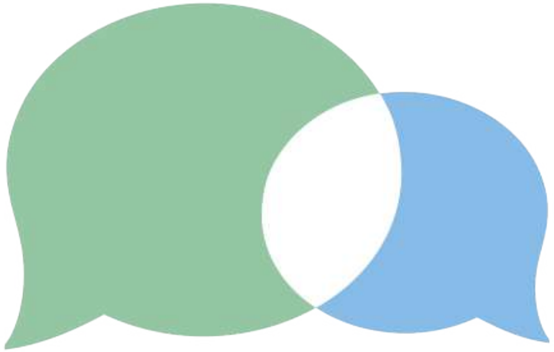


Table of contents

Conclusions



- Real post-quantum e-voting system demonstrator implemented
- But still not ready for production
  - Several implementation limitations exist (reduced security)
  - Mixnet too slow (hours to complete)

Voter perspective:

lattice-based cryptography does not significantly impact individual voter experience

Administration board perspective:

lattice-based cryptography significantly impact tally performance

- mixnet cannot yet reach the product performance requirements
- more work and optimizations are needed to achieve reasonable numbers
- refinement of mixnet protocol should take priority over parameters optimization

Thanks!