STARKs in an Eggshell

October 2019

Daniel Benarroch

Anaïs Querol

qedit

institute IMdea software

"la Caixa" Foundation

ZKProof.org

© QEDIT 2019

Today

Introduction to proofs

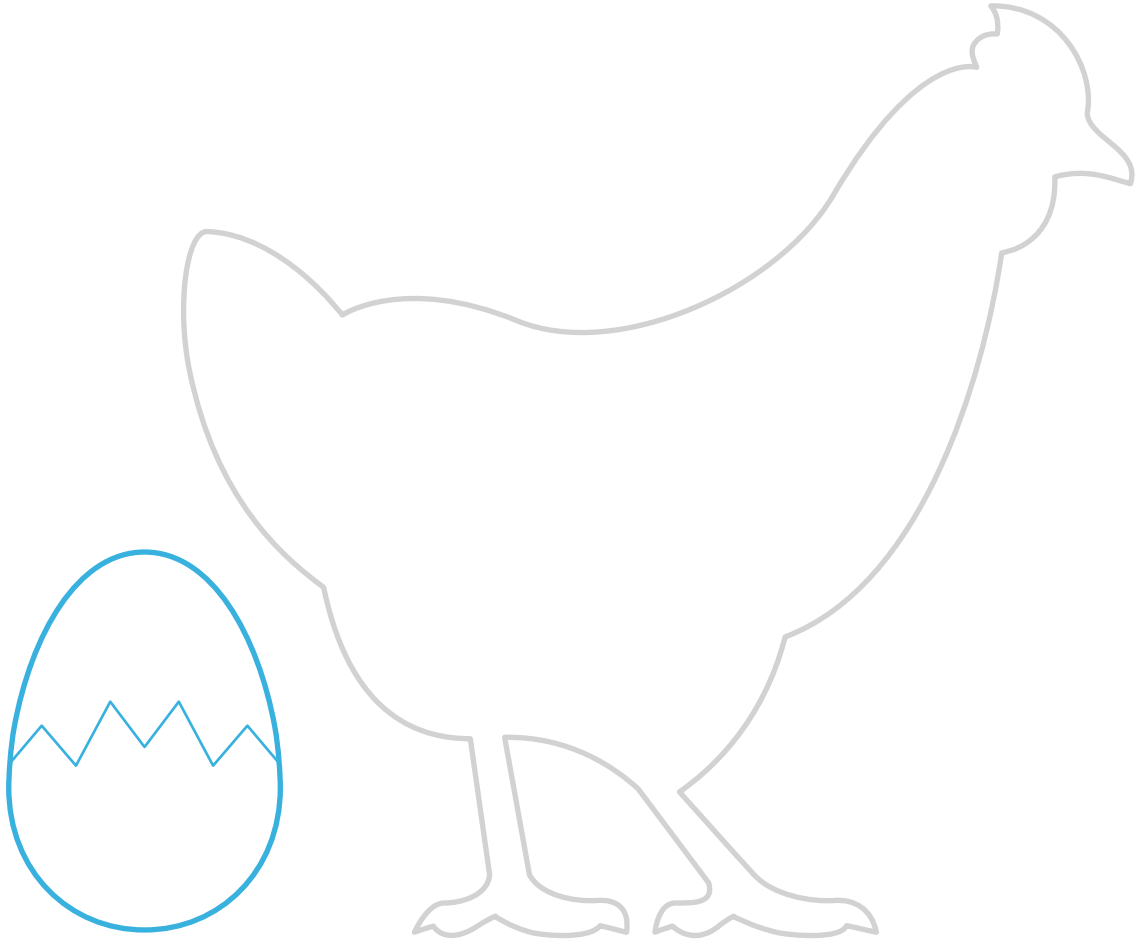
Survey on STARKs

Available libraries

Experiments

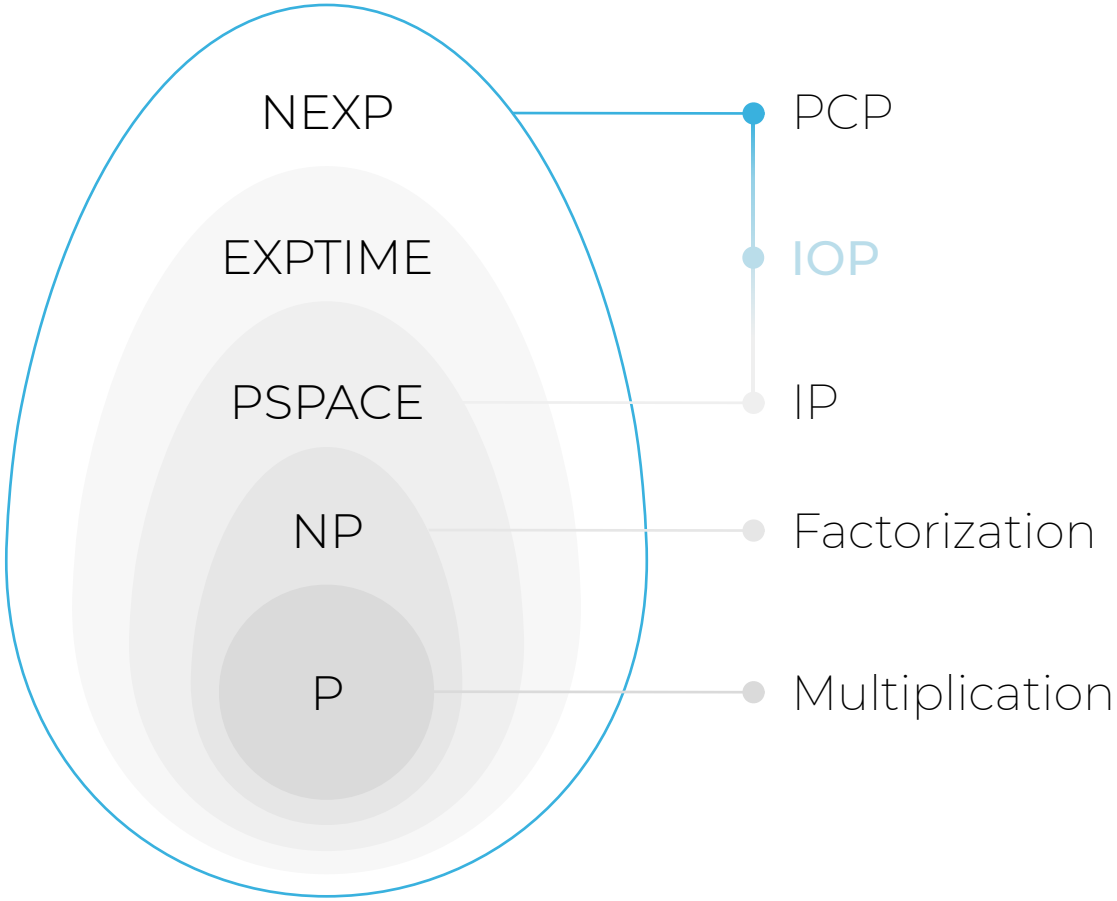
© QEDIT 2019

Philosophical "which came first" doses



© QEDIT 2019

Philosophical "which came first" doses



© QEDIT 2019

Cryptographic Beasts

—

S uccinct

N on-interactive

A

R guments of

K nowledge

© QEDIT 2019

Cryptographic Beasts and how to tame them

S calable

T ransparent

A

R guments of

K nowledge

Ben-Sasson, Bentov, Horesh, Riabzev https://eprint.iacr.org/2018/046.pdf

© QEDIT 2019

Proofs

$C^{\{T\}}(x, w) = y$

that's the truth

cannot do it

© QEDIT 2019

Interactive Proofs

$C^{\{T\}}(x, w) = y$

that's the truth

challenge$_i$

answer$_i$

let's check

⋮

© QEDIT 2019

Interactive

Oracle Proofs

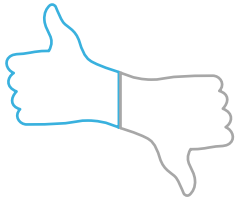
$$C^{\{T\}}(x, w) \stackrel{?}{=} y$$

rounds

challenge$_i$ ←

PCP$_i$ →

⋮



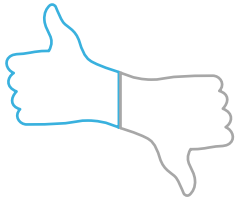
© QEDIT 2019

Non – Interactive

Oracle Proofs

$$C^{\{T\}}(x, w) \stackrel{?}{=} y$$

proof

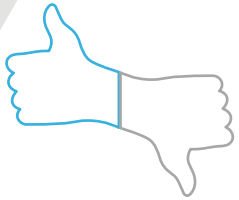


© QEDIT 2019

Non – Interactive Random Oracle Proofs

$$C^{\{T\}}(x, w) \stackrel{?}{=} y$$

proof



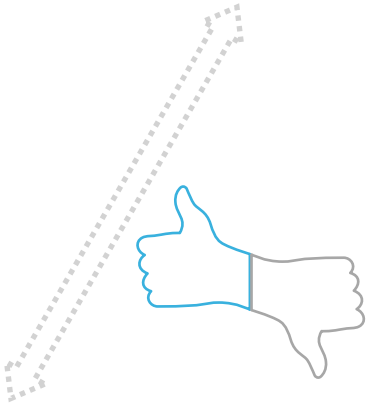
© QEDIT 2019

Scalable Transparent IOP of Knowledge

$$C^{\{T\}}(x, w) \stackrel{?}{=} y$$

challenge$_i$

PCP$_i$



© QEDIT 2019

Scalable Transparent IOP of Knowledge

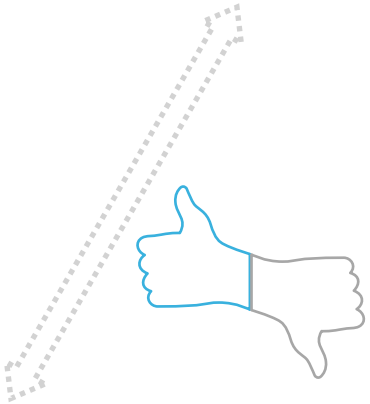
quasilinear

$$C^{\{T\}}(\mathbf{x}, w) \stackrel{?}{=} y$$

polylog

challenge$_i$

PCP$_i$



© QEDIT 2019

Scalable Transparent ARgument Knowledge

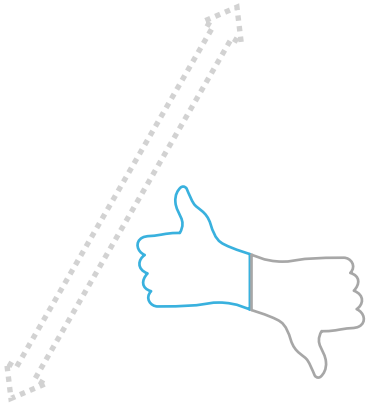
quasilinear

$$C^{\{T\}}(\mathbf{x}, w) \stackrel{?}{=} y$$

polylog

challenge$_i$

PCP$_i$



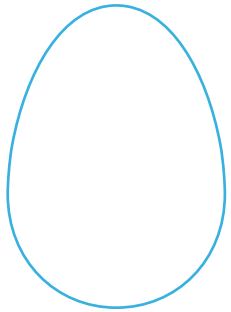
How to create a STARK?

© QEDIT 2019

Let’s stark by frying an egg!

© QEDIT 2019

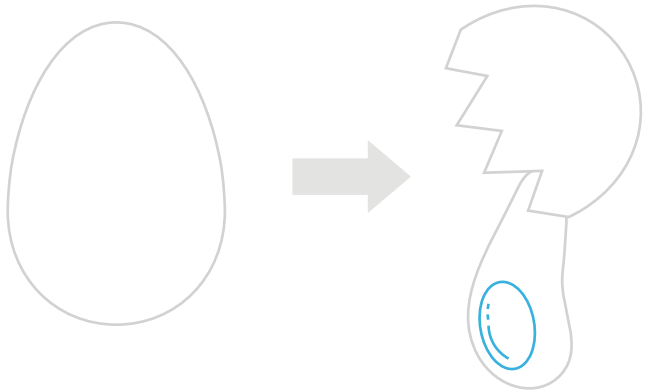
Let’s stark by frying an egg!



EGG

© QEDIT 2019

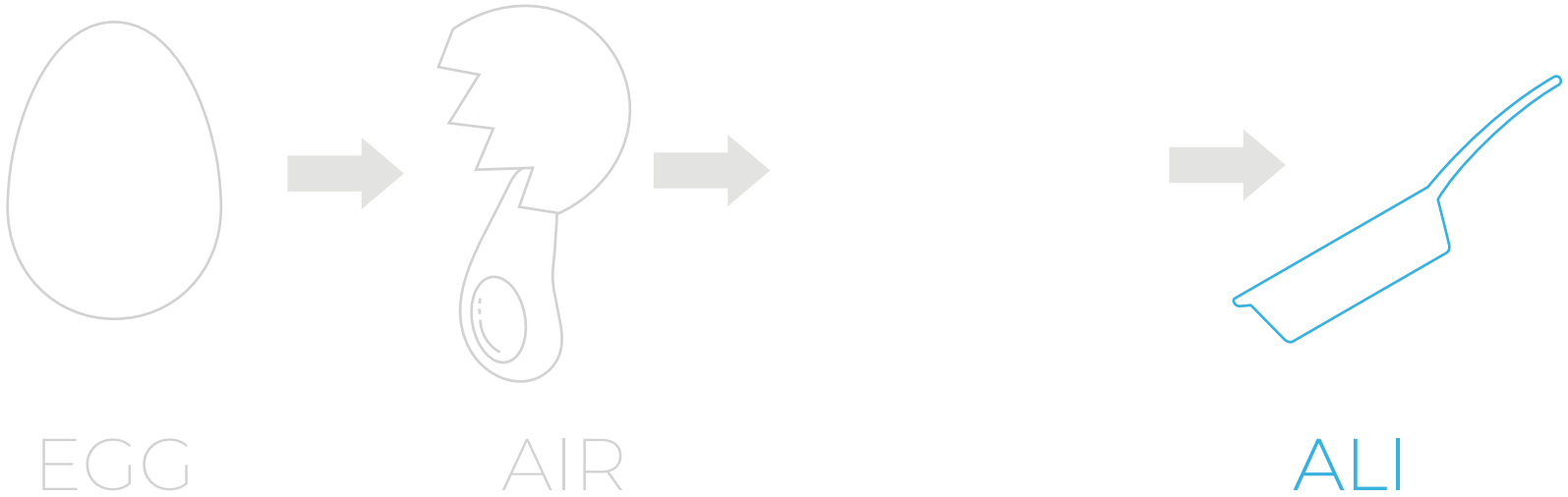
Let’s stark by frying an egg!



EGG AIR

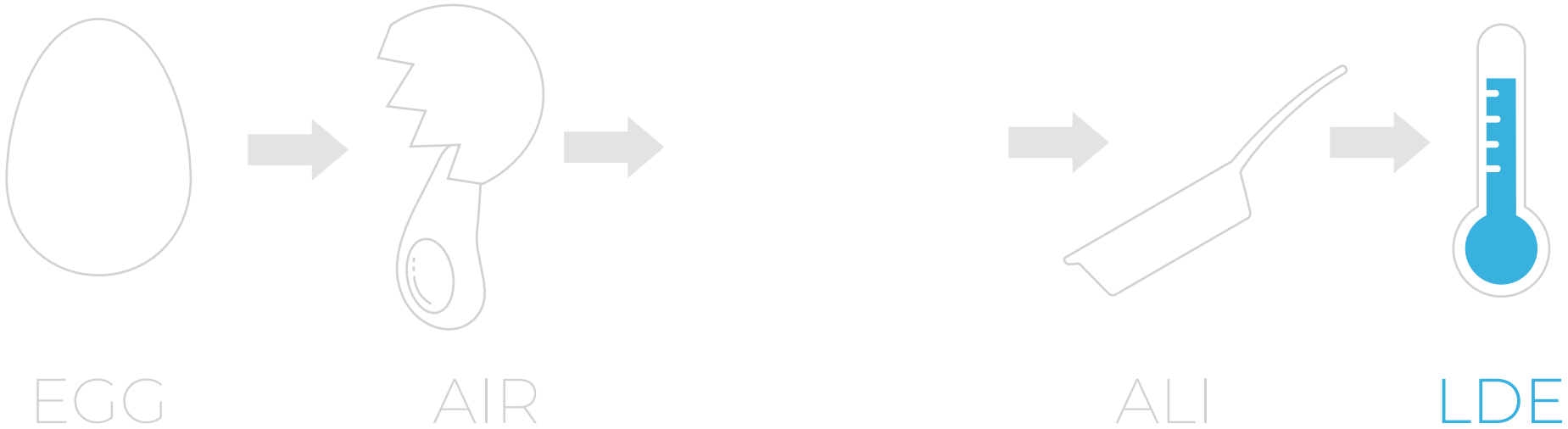
© QEDIT 2019

Let’s stark by frying an egg!



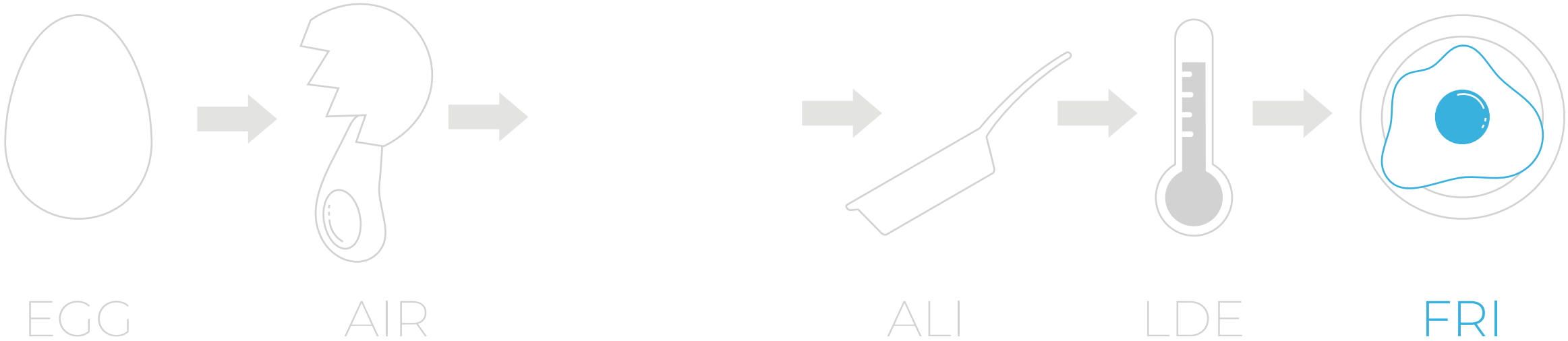
© QEDIT 2019

Let’s stark by frying an egg!



© QEDIT 2019

Let’s stark by frying an egg!



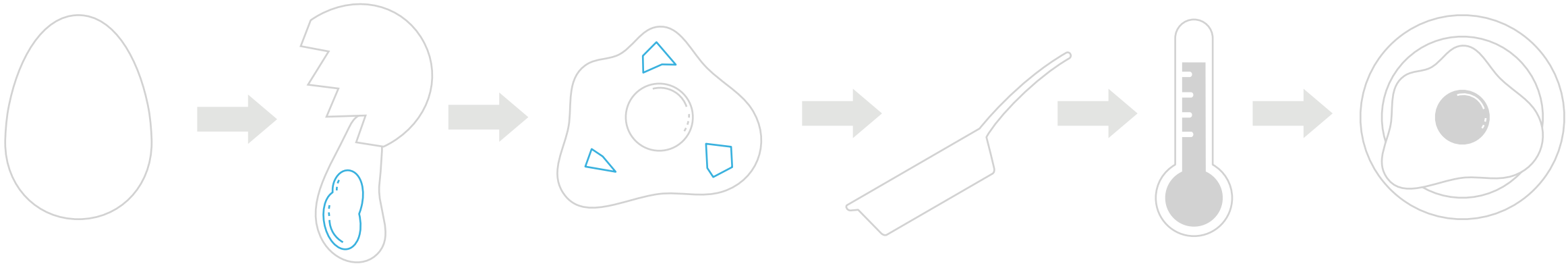
© QEDIT 2019

Let’s stark by frying an egg!

problem constraints compiler interpolate proof

© QEDIT 2019

Let's stark by frying an egg!



EGG	PAIR	APR	ALI	LDE	FRI
problem	constraints	optimize (not in genSTARK)	compiler	interpolate	proof

© QEDIT 2019

Let’s stark by frying an egg!

claim on computation → claim on polynomials → claim on low-degreeness

© QEDIT 2019

AIR

the skilled step

© QEDIT 2019

Arithmetic Intermediate Representation

a_1 initial value

a_T final value

© QEDIT 2019

Arithmetic Intermediate Representation

a_1 initial value

a_2 transition function $P(X)$

$P(i) = P(i-1) + \ldots = a_i$

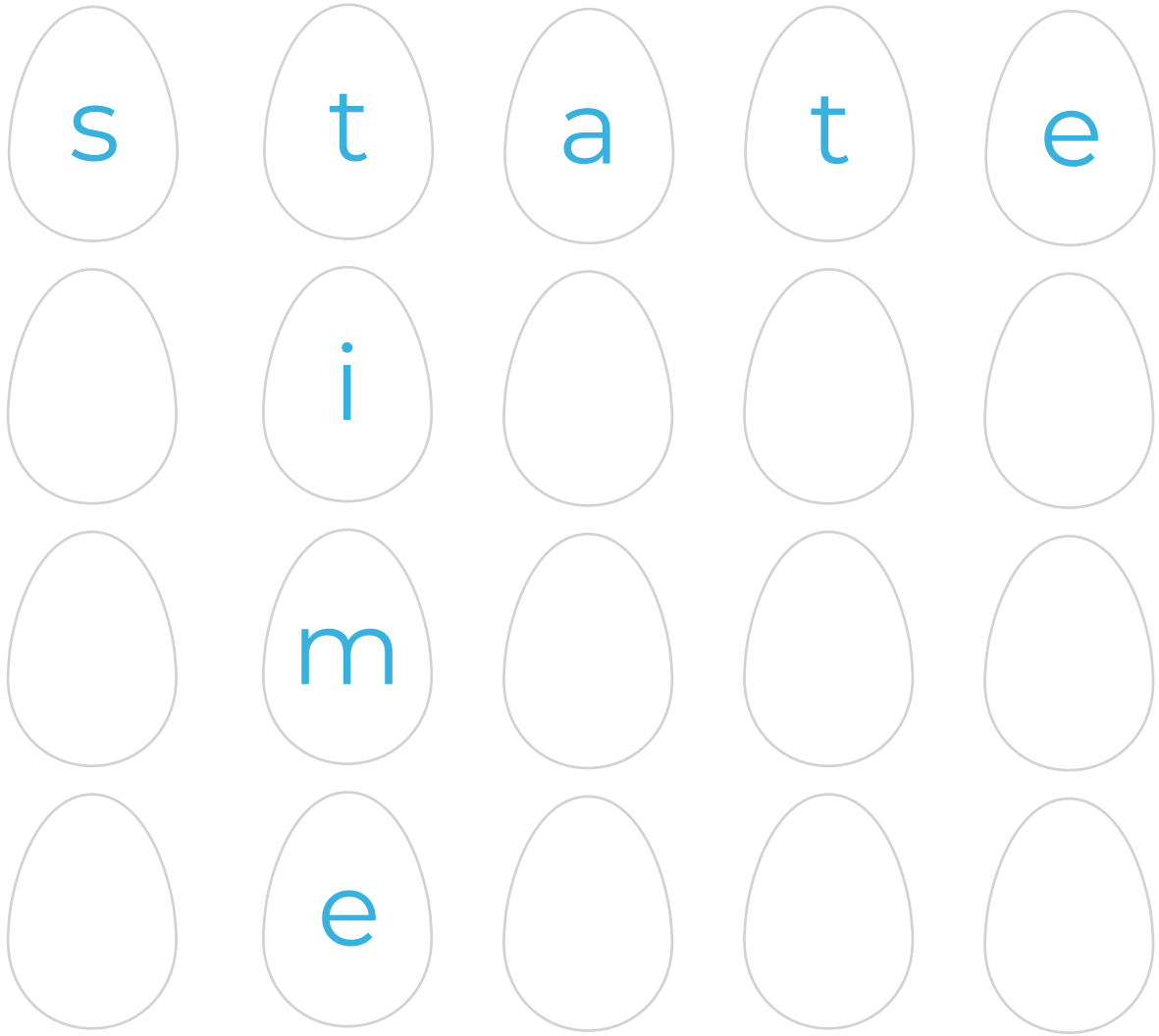
⋮ constraint polynomial $Q(X)$

$Q(i) = P(i) - a_i = 0$

a_T final value

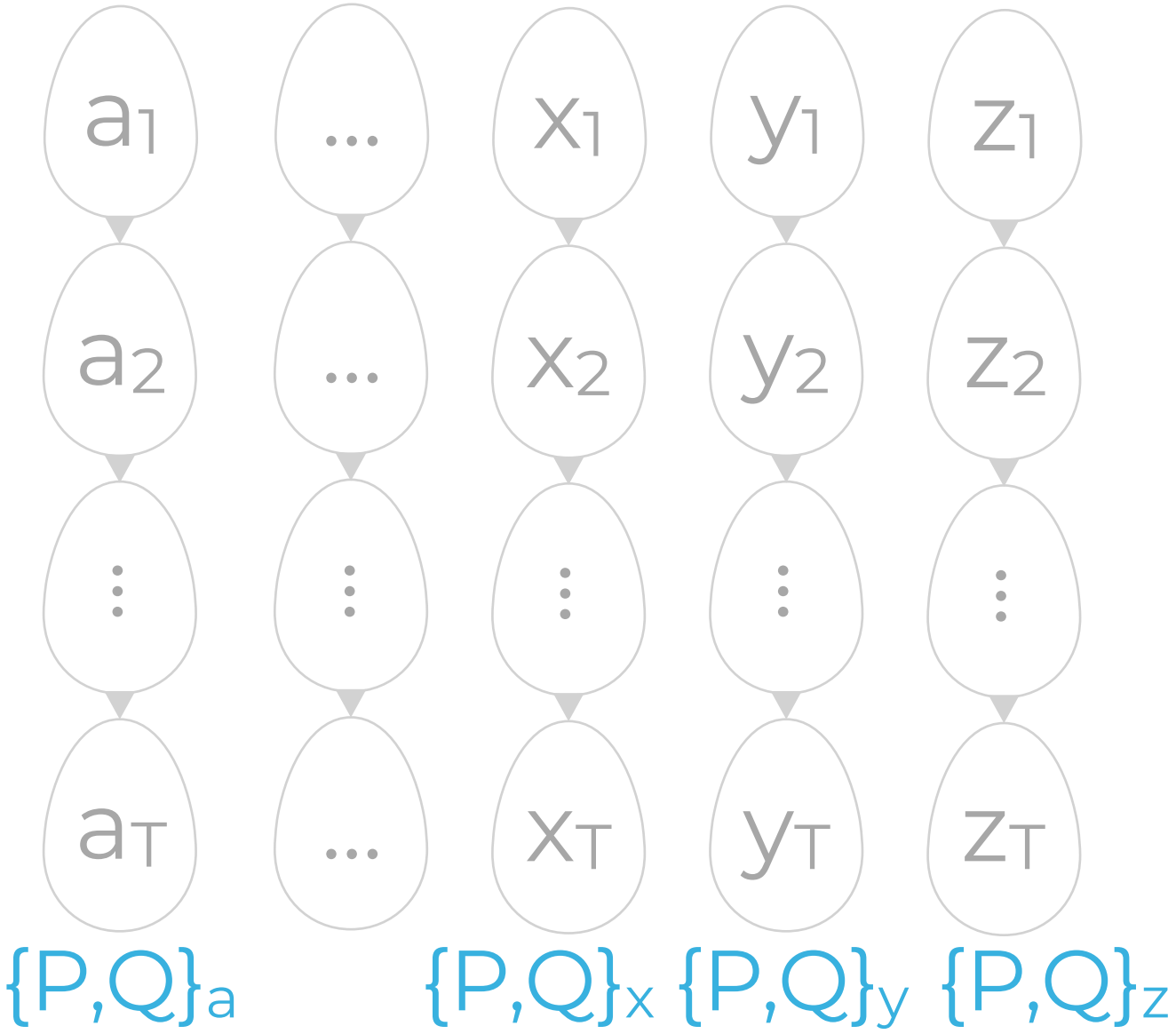
© QEDIT 2019

Arithmetic Intermediate Representation



© QEDIT 2019

Arithmetic Intermediate Representation



© QEDIT 2019

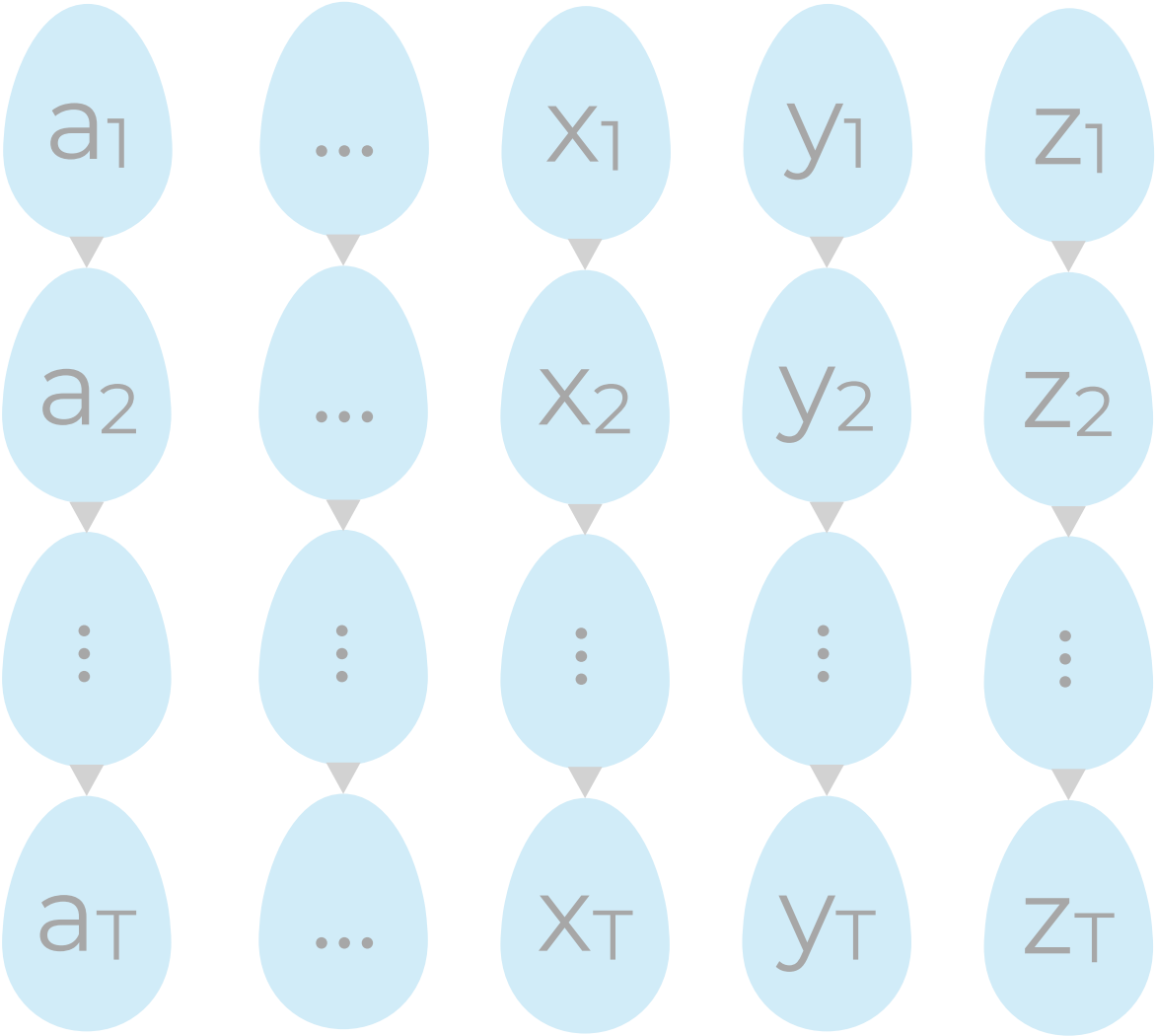
Arithmetic Intermediate Representation

Finite field

$\mathbb{F} =$

© QEDIT 2019

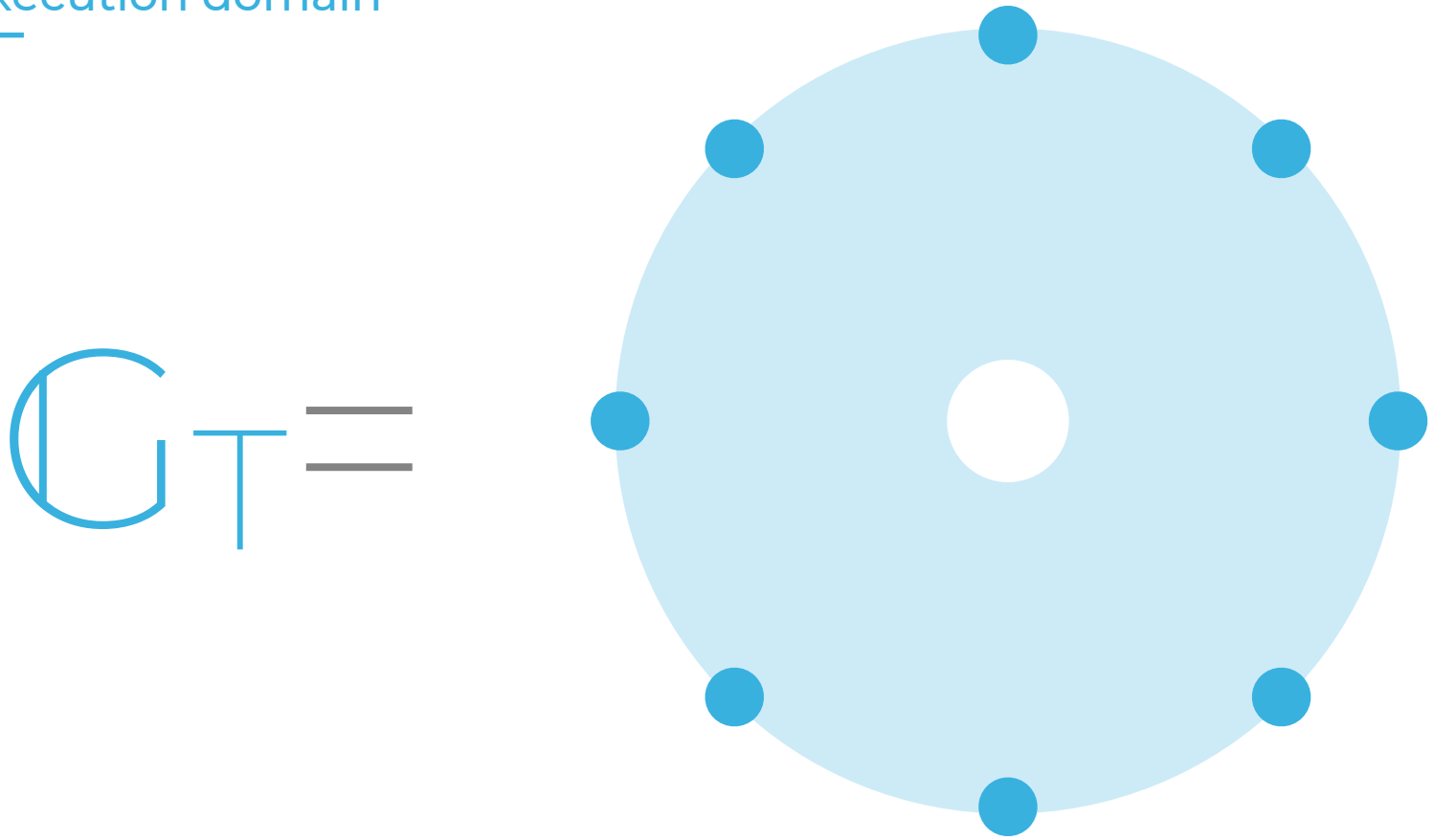
Arithmetic Intermediate Representation



© QEDIT 2019

Arithmetic Intermediate Representation

Execution domain



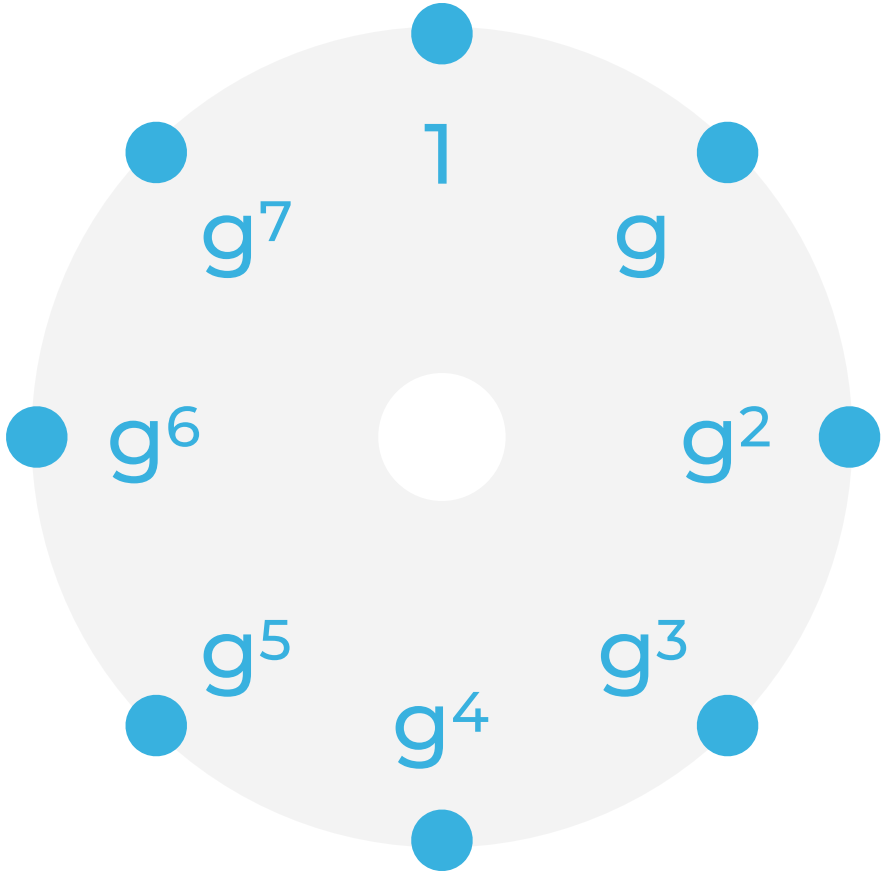
© QEDIT 2019

Arithmetic Intermediate Representation

Execution domain

$$\mathbb{G}_T =$$

$T = 8$



© QEDIT 2019

Arithmetic Intermediate Representation

a_1 initial value

a_2 transition function $P(X)$

$P(g^i) = P(g^{i-1}) + \ldots = a_i$

⋮ constraint polynomial $Q(X)$

$Q(g^i) = P(g^i) - a_i = 0$

a_T final value

© QEDIT 2019

ALI

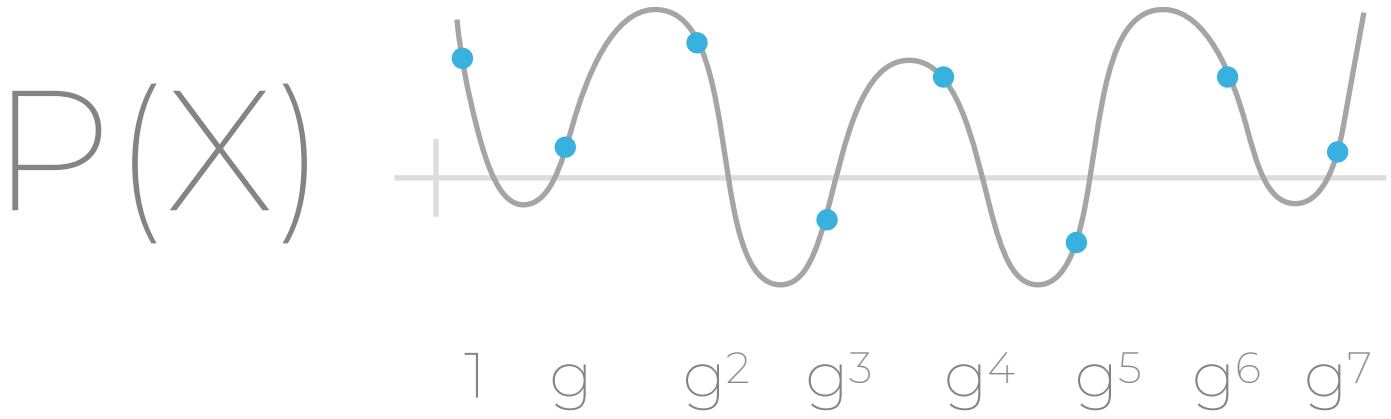
more polynomials

© QEDIT 2019

Algebraic Linking IOP

Boundary constraints

$$B(X) = \frac{P(X) - I(X)}{Z'(X)}$$



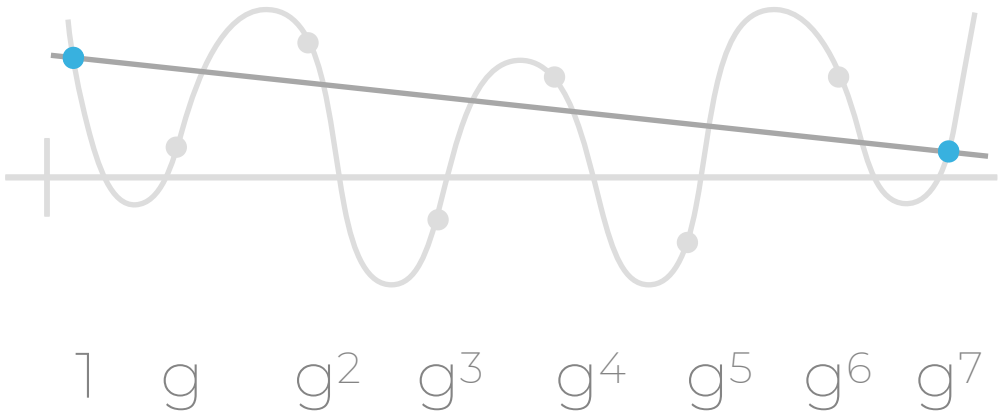
© QEDIT 2019

Algebraic Linking IOP

Boundary constraints

$$B(X) = \frac{P(X) - I(X)}{Z'(X)}$$

$I(X)$



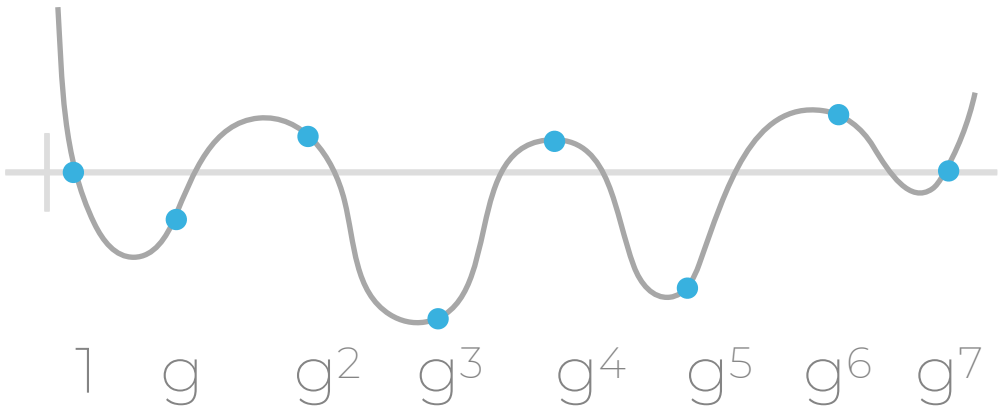
© QEDIT 2019

Algebraic Linking IOP

Boundary constraints

$$B(X) = \frac{P(X)-I(X)}{Z'(X)}$$

P - I



© QEDIT 2019

Algebraic Linking IOP

Boundary constraints

$$B(X) = \frac{P(X) - I(X)}{Z'(X)}$$

$$Z'(X) = (X - 1) \cdot (X - g^7)$$

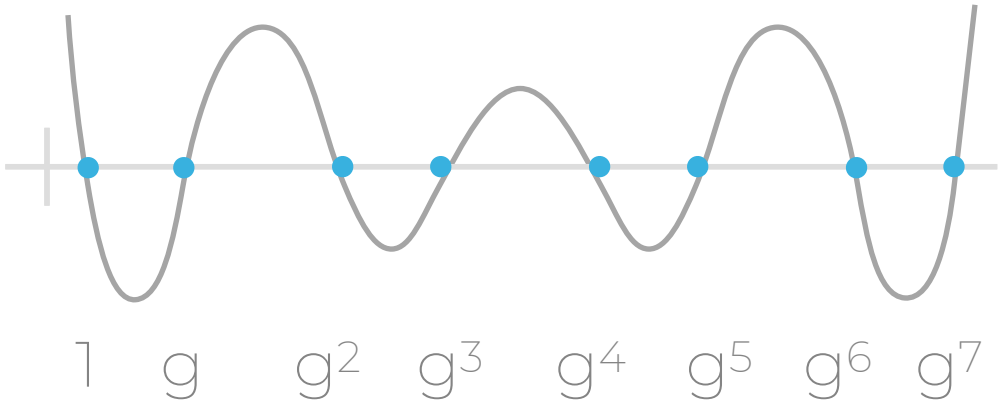
© QEDIT 2019

Algebraic Linking IOP

Execution trace

$$D(X) = \frac{Q(X)}{Z(X)}$$

$Q(X)$



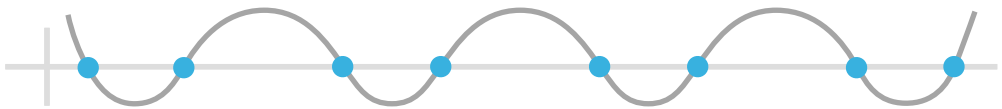
© QEDIT 2019

Algebraic Linking IOP

Execution trace

$$D(X) = \frac{Q(X)}{Z(X)}$$

$$Z(X) = \prod_{i=0}^{7} (X - g^i)$$



$1 \quad g \quad g^2 \quad g^3 \quad g^4 \quad g^5 \quad g^6 \quad g^7$

© QEDIT 2019

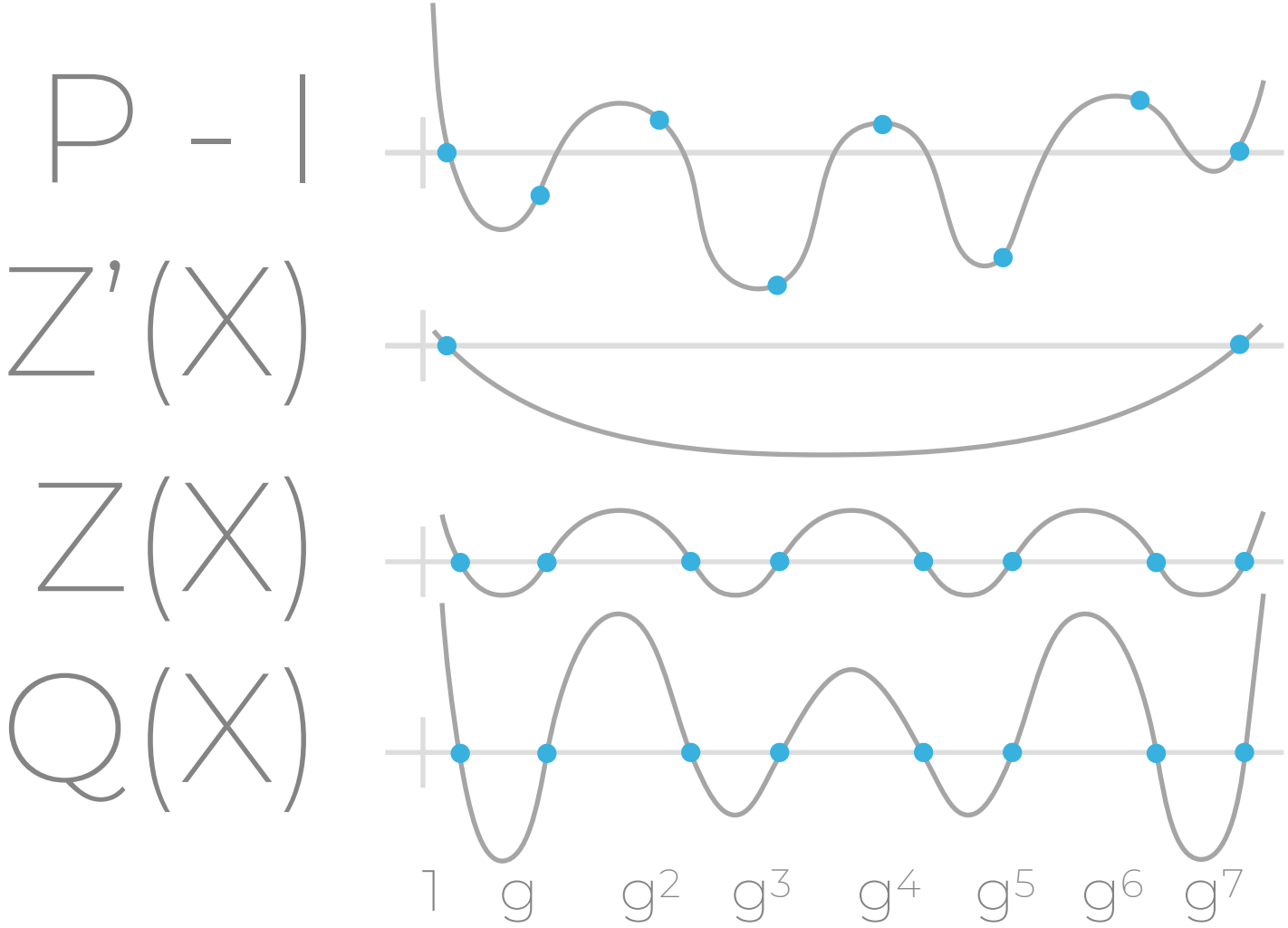
LDE

the bottleneck

© QEDIT 2019

Low – Degree Extension

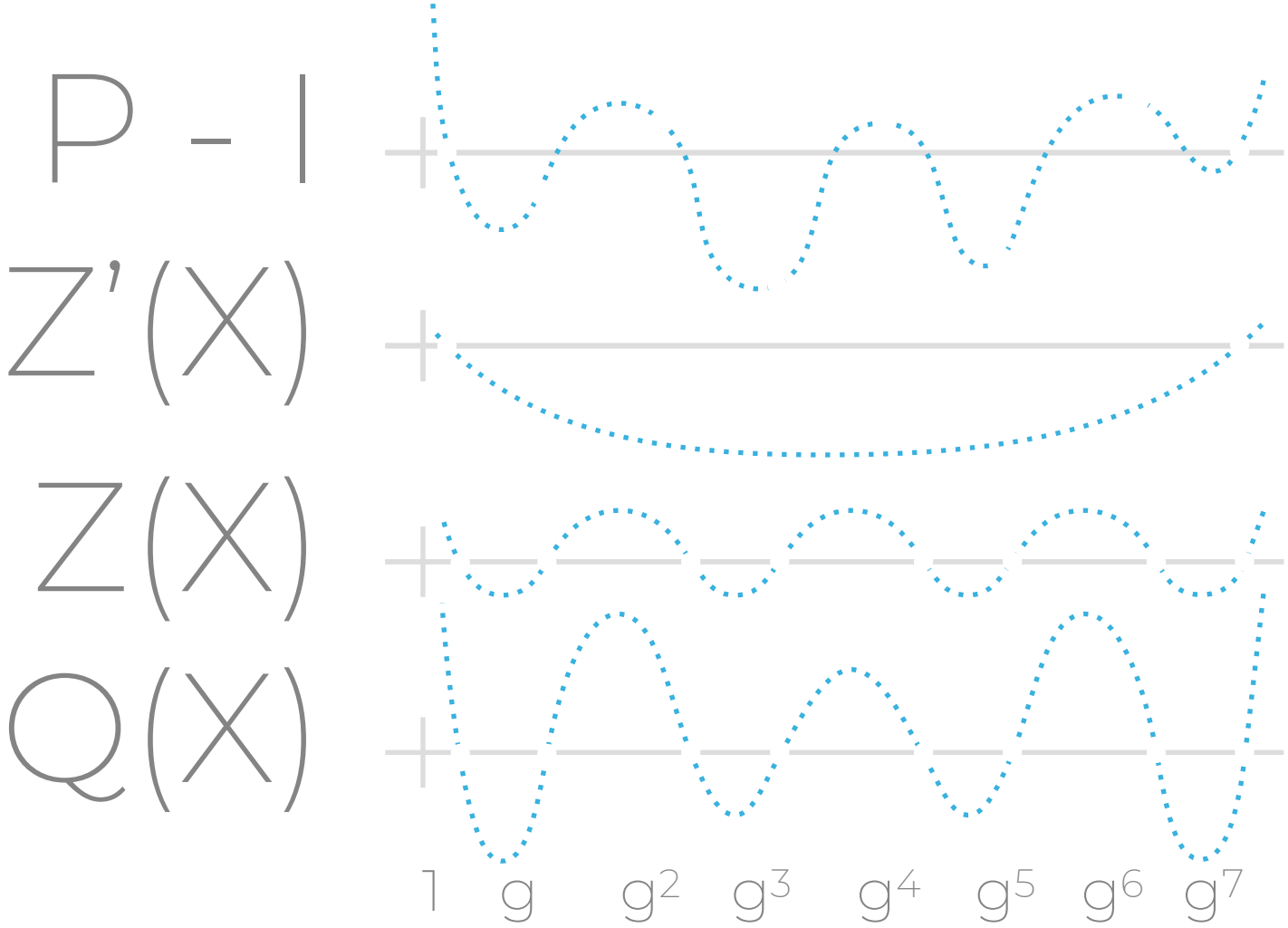
Execution domain



© QEDIT 2019

Low – Degree Extension

Evaluation domain

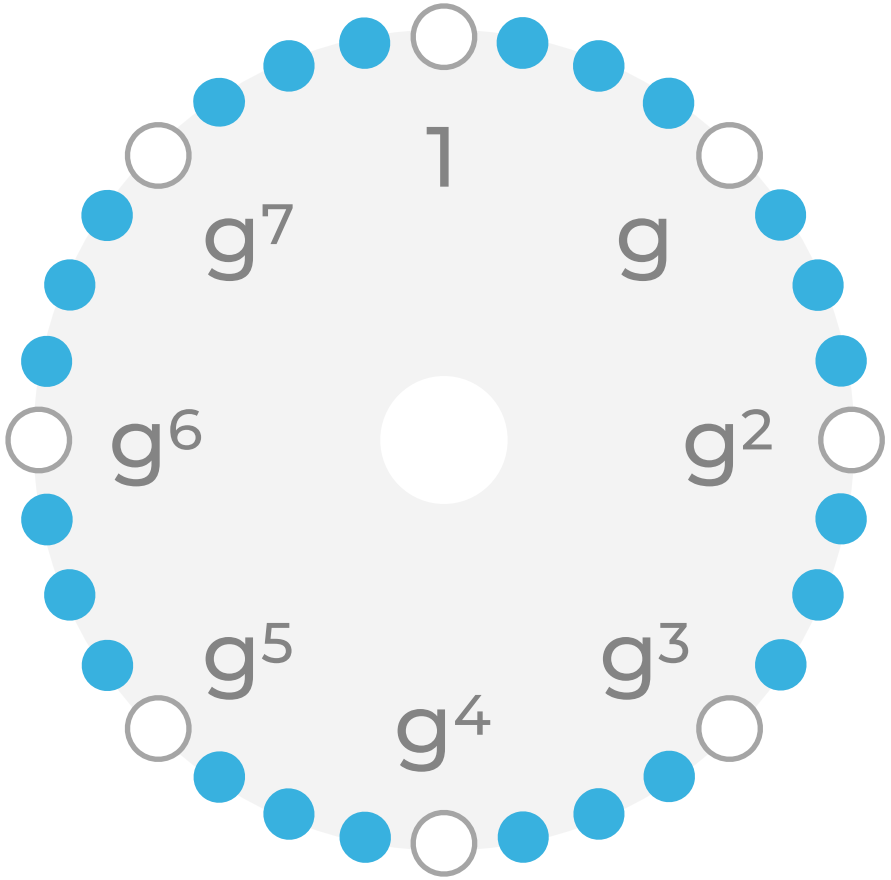


© QEDIT 2019

Low – Degree Extension

Extended group

$\mathbb{G}_E =$

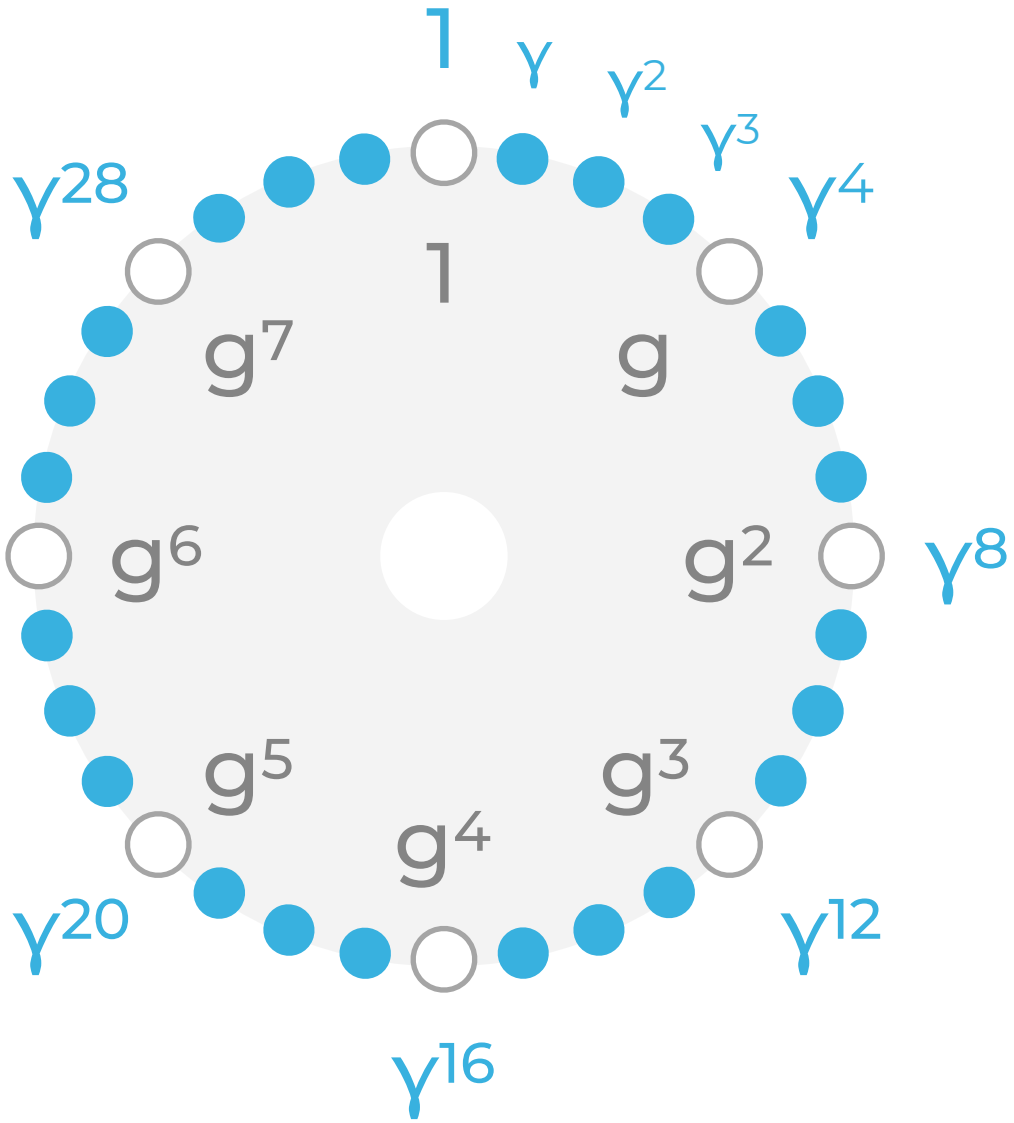


© QEDIT 2019

Low – Degree Extension

Extended group

$$\mathbb{G}_E =$$



$$E = T \cdot e$$

$$e = 4$$

© QEDIT 2019

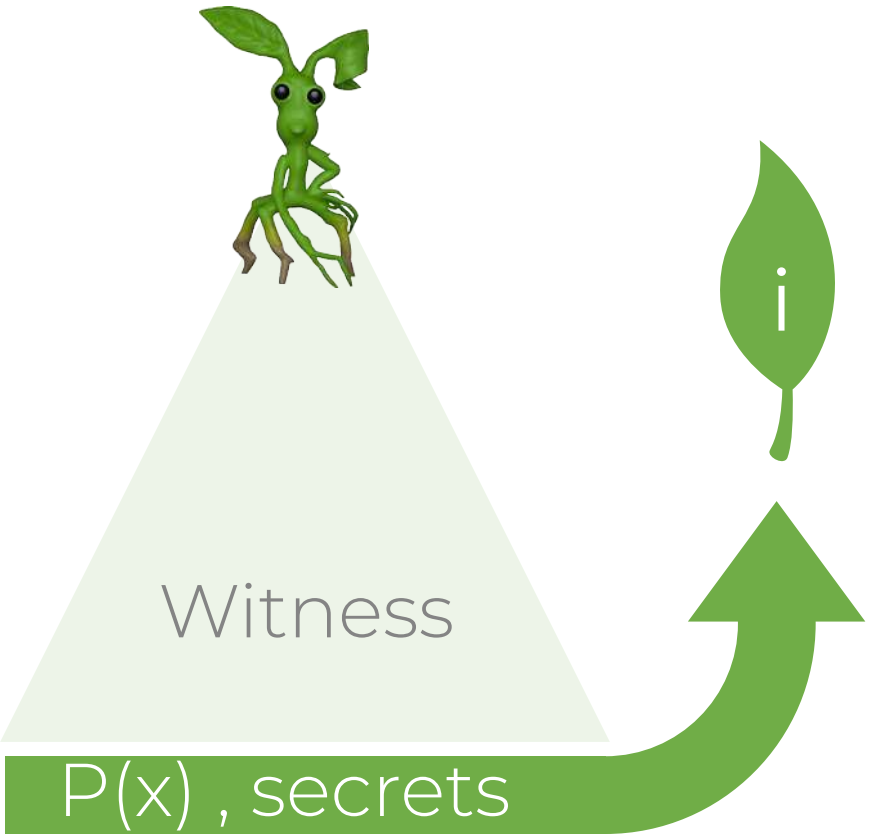
FRI

their new technique

© QEDIT 2019

Fast Reed Solomon IOP Proximity

Merkle trees



$P_a(\gamma^i), \ldots, P_z(\gamma^i)$

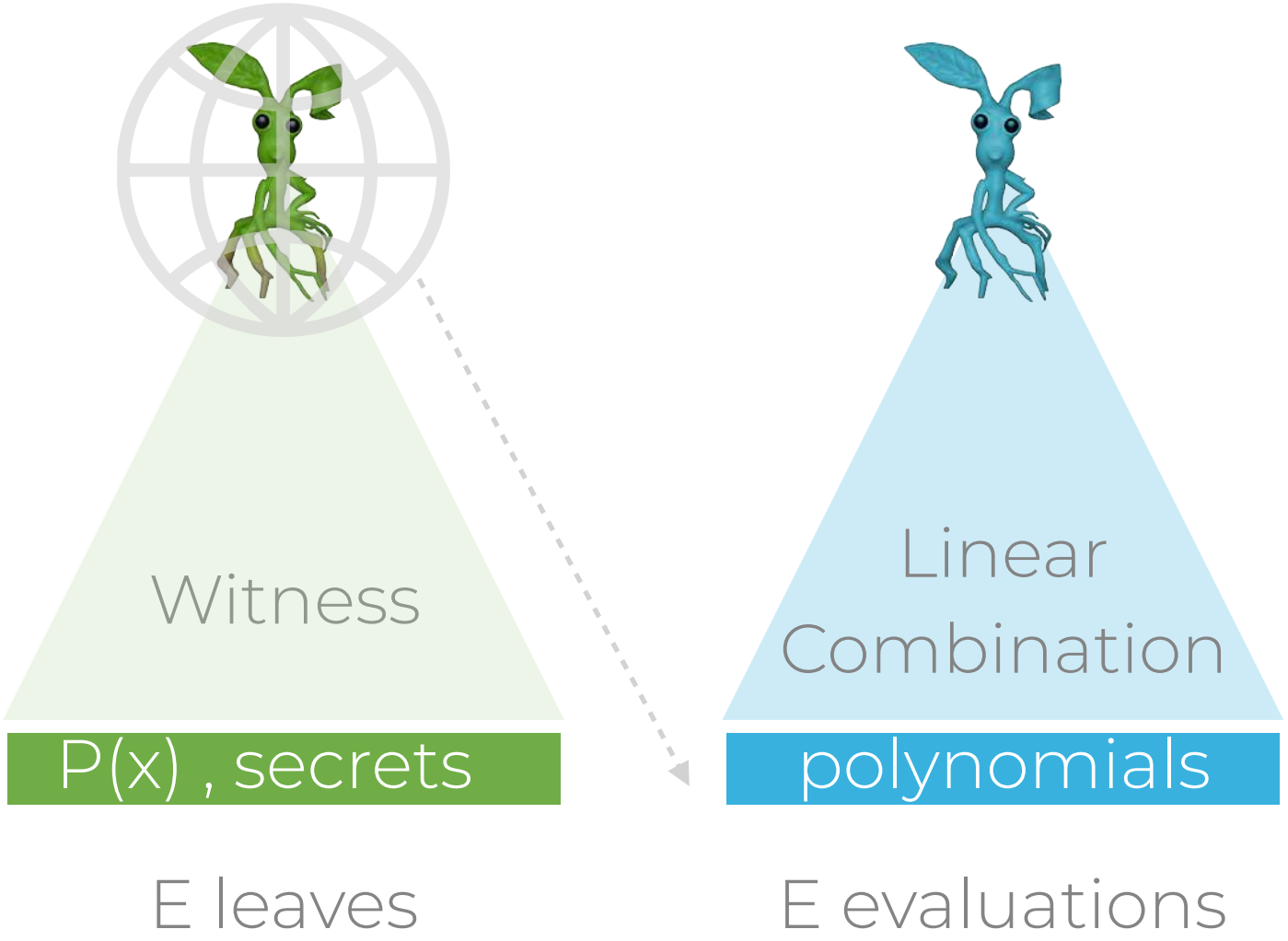
$S_\alpha(\gamma^i), \ldots, S_\omega(\gamma^i)$

E leaves

© QEDIT 2019

Fast Reed Solomon IOP Proximity

Merkle trees



© QEDIT 2019

Fast Reed Solomon IOP Proximity

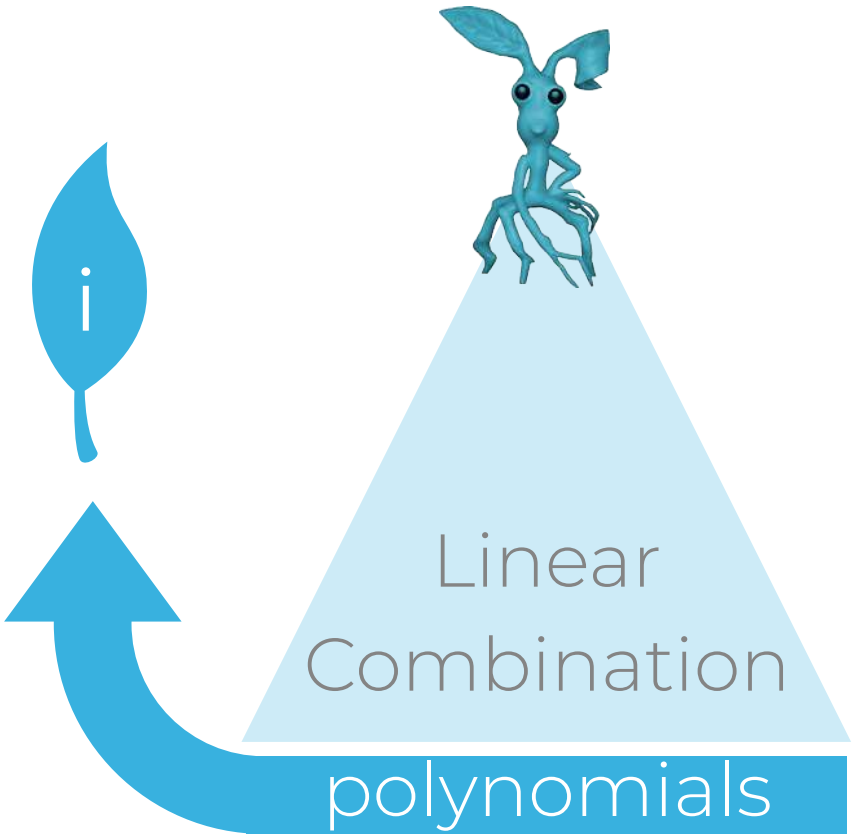
Merkle trees

$P_a(\gamma^i), \dots, P_z(\gamma^i)$

$S_\alpha(\gamma^i), \dots, S_\omega(\gamma^i)$

$B_a(\gamma^i), \dots, B_z(\gamma^i)$

$D_a(\gamma^i), \dots, D_z(\gamma^i)$



E evaluations

© QEDIT 2019

Fast Reed Solomon IOP Proximity

Motivation

need $T(\mu-1)+1$ points $> T$

$E = T \cdot e$ degree $T(\mu-1)$ P, S, B, D

© QEDIT 2019

Fast Reed Solomon IOP Proximity

Motivation

linear

FFT-ish

log

$$F(X) = F_{even}(X^2) + X \cdot F_{odd}(X^2)$$

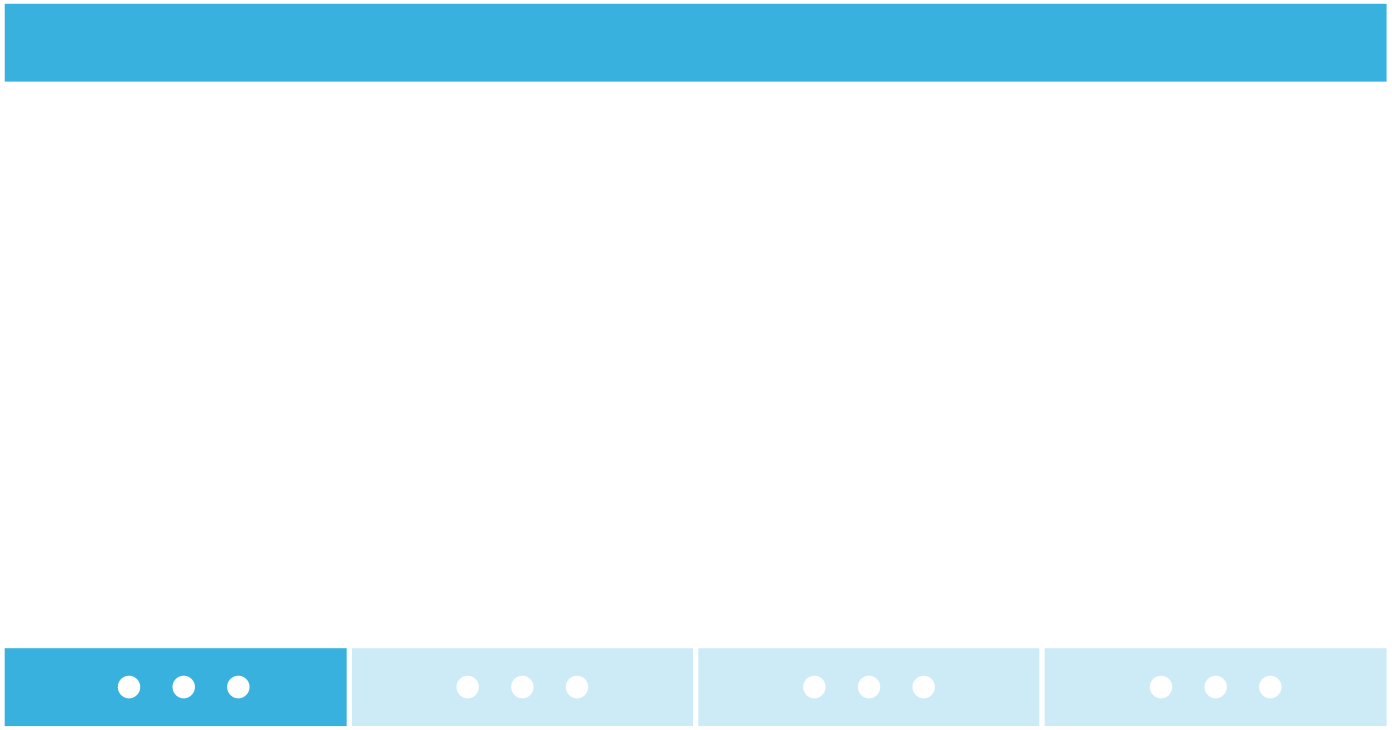
© QEDIT 2019

Fast Reed Solomon IOP Proximity

Commit phase

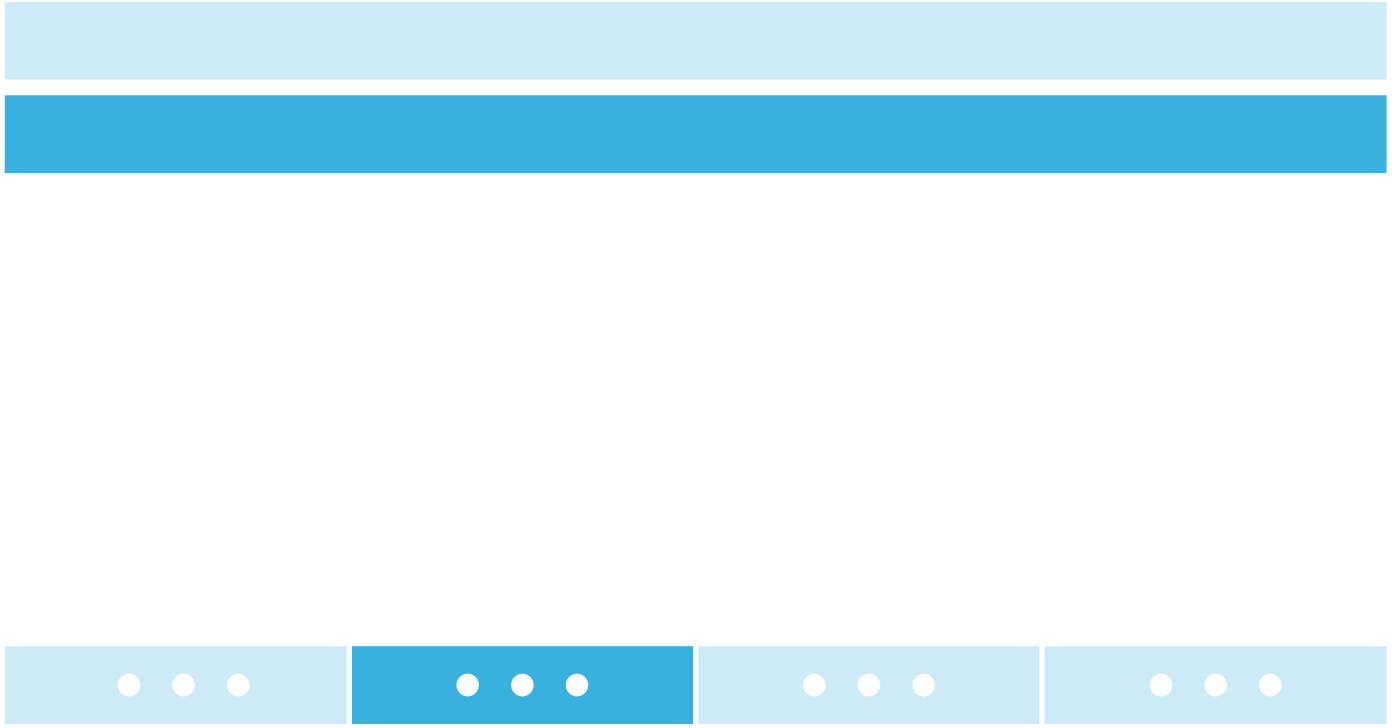
Fast Reed Solomon IOP Proximity

Commit phase



Fast Reed Solomon IOP Proximity

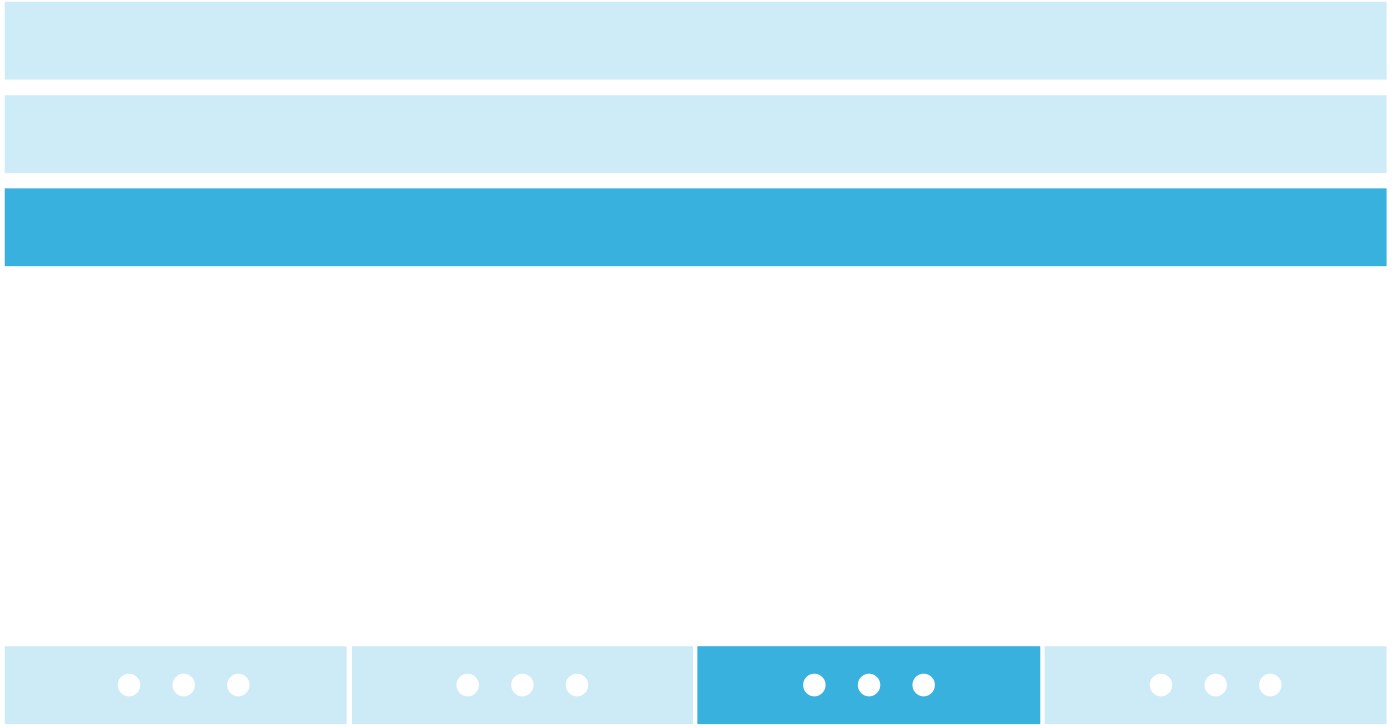
Commit phase



© QEDIT 2019

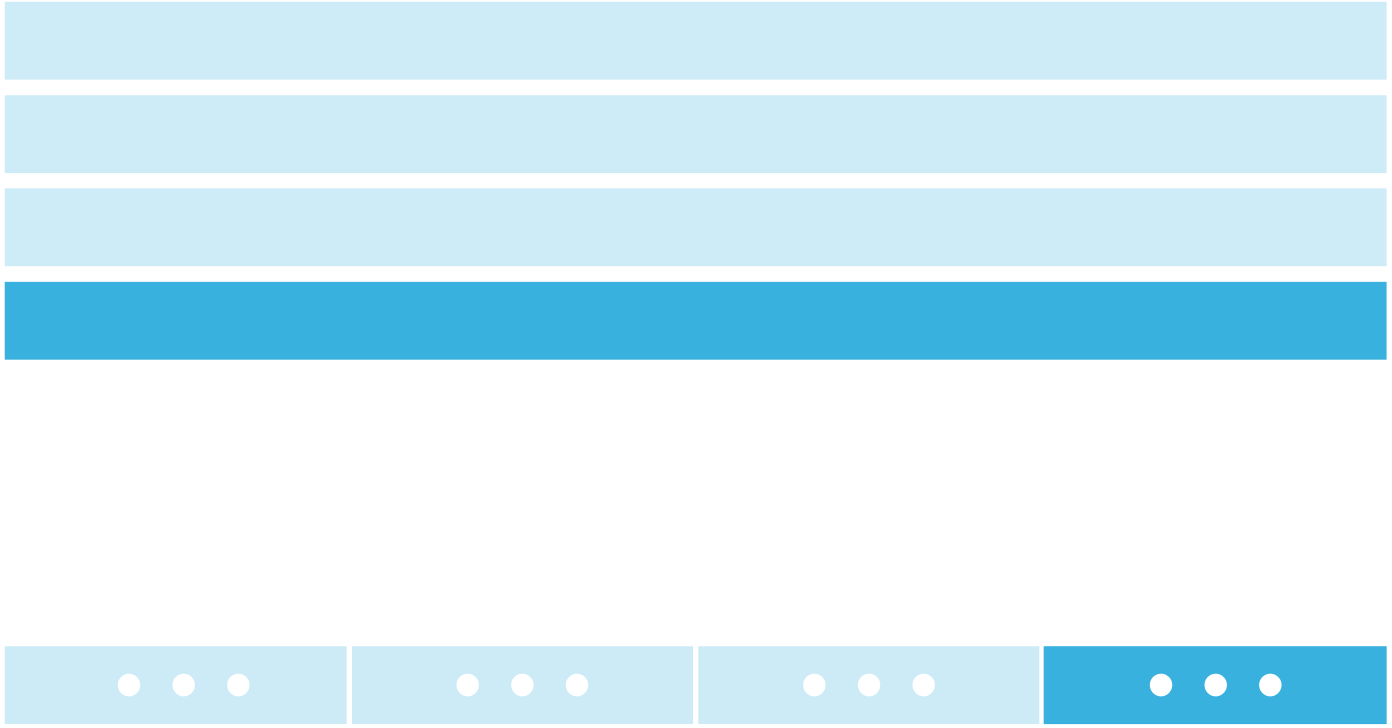
Fast Reed Solomon IOP Proximity

Commit phase



Fast Reed Solomon IOP Proximity

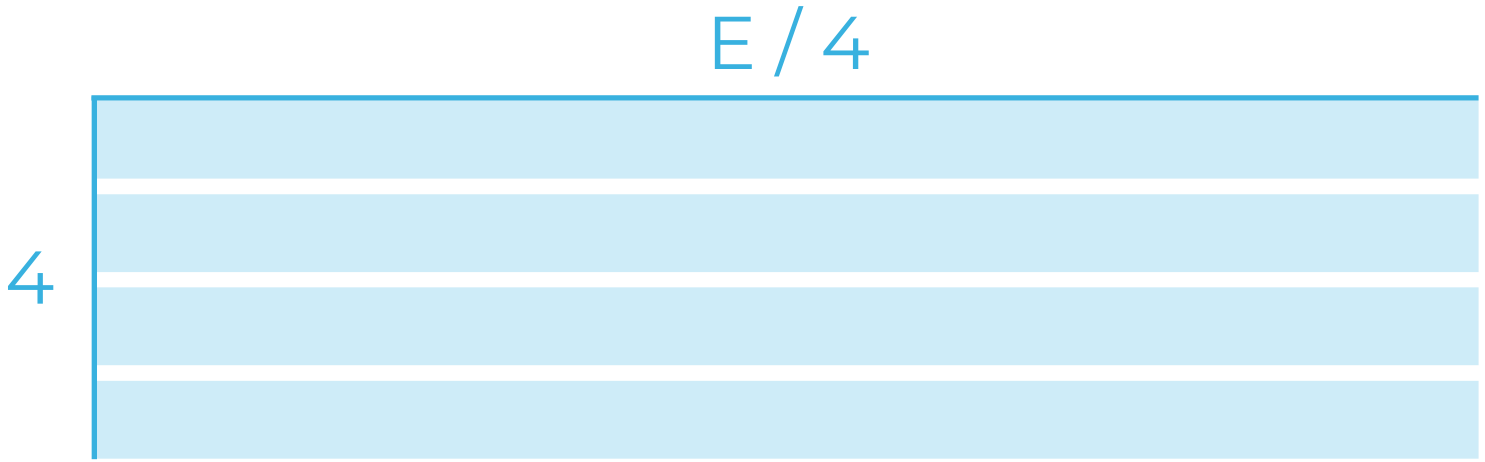
Commit phase



© QEDIT 2019

Fast Reed Solomon IOP Proximity

Commit phase



© QEDIT 2019

Fast Reed Solomon IOP Proximity

Commit phase

y_0

y_{E-1}

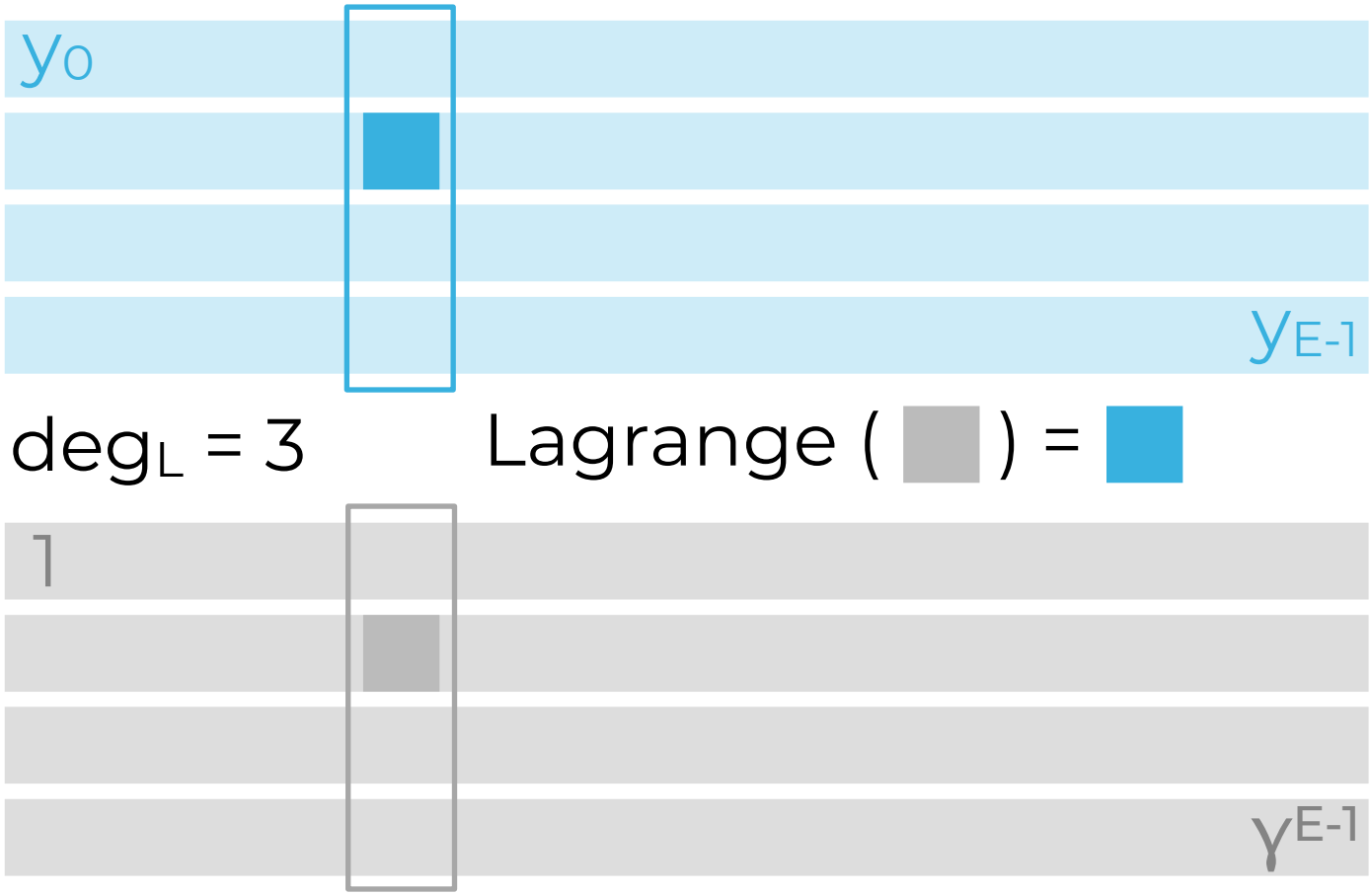
1

γ^{E-1}

© QEDIT 2019

Fast Reed Solomon IOP Proximity

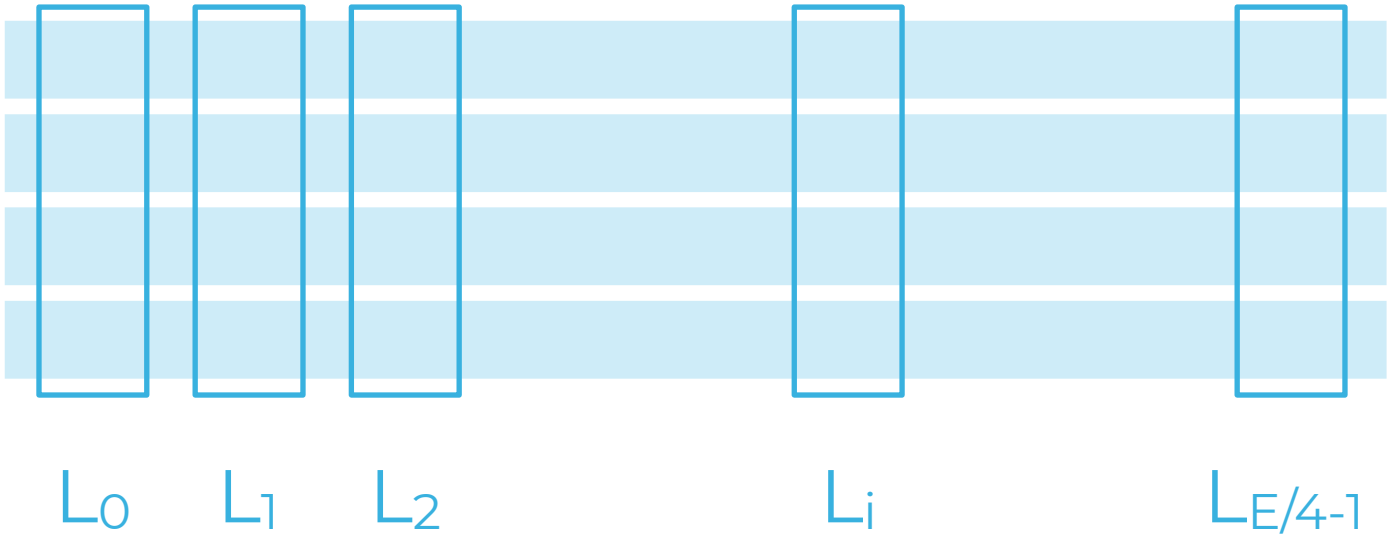
Commit phase



© QEDIT 2019

Fast Reed Solomon IOP Proximity

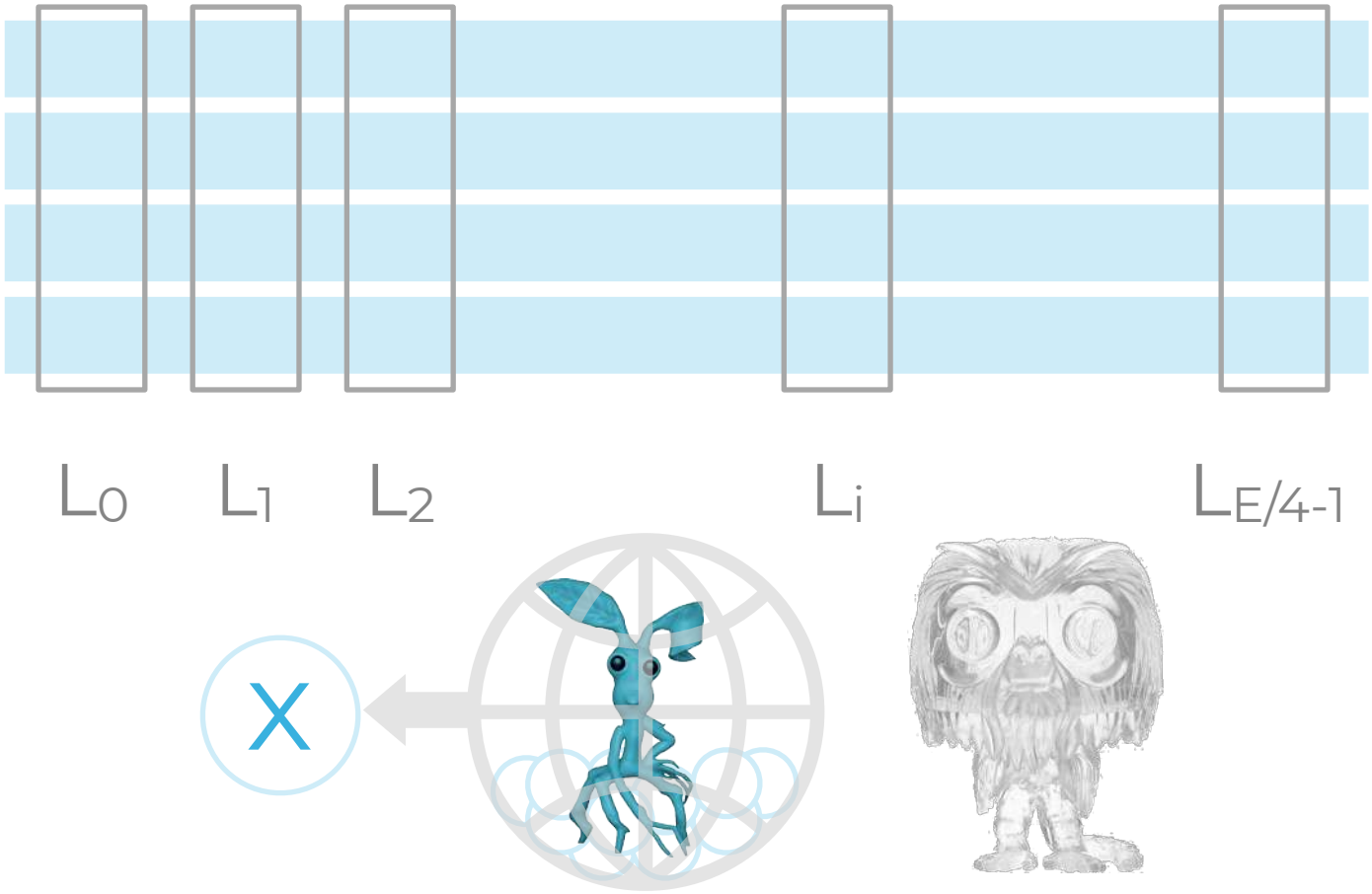
Commit phase



© QEDIT 2019

Fast Reed Solomon IOP Proximity

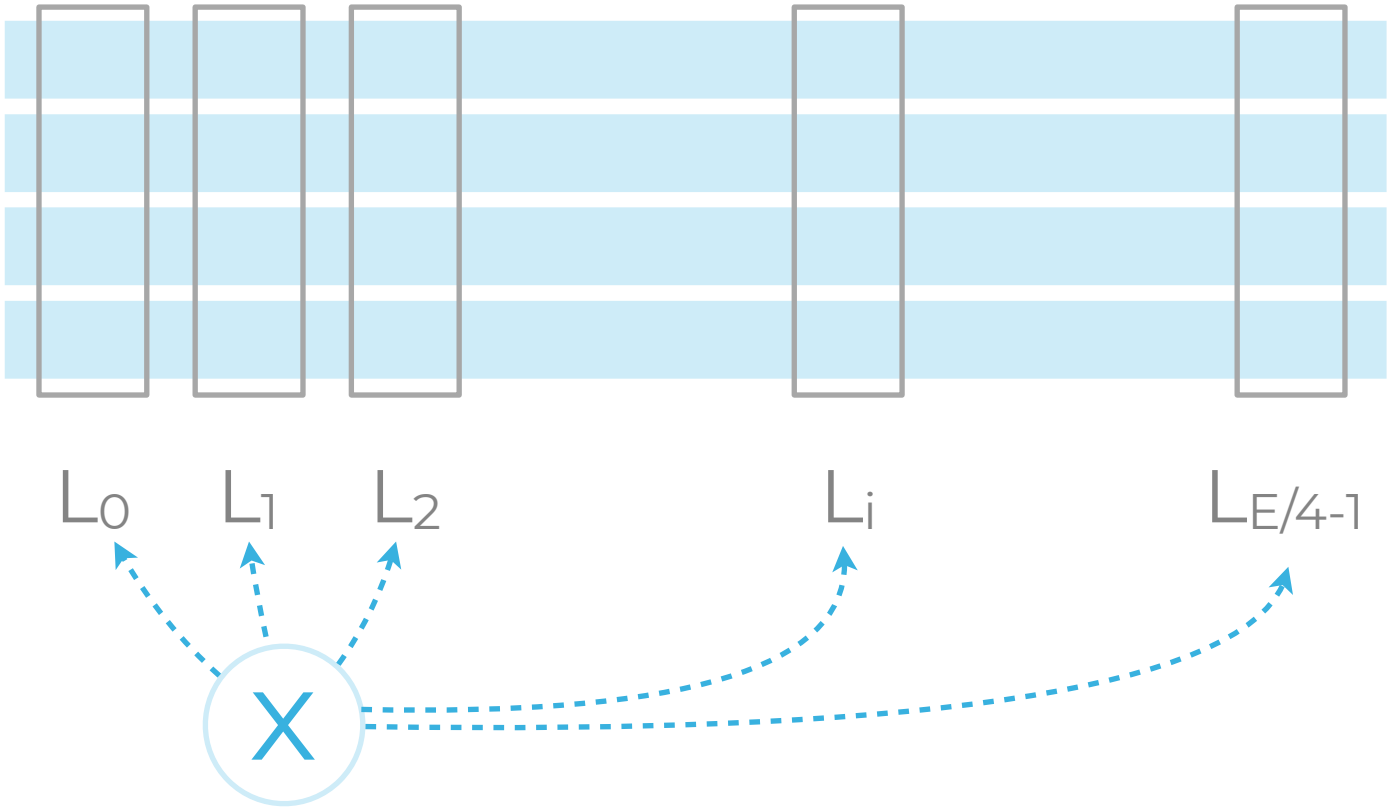
Commit phase



© QEDIT 2019

Fast Reed Solomon IOP Proximity

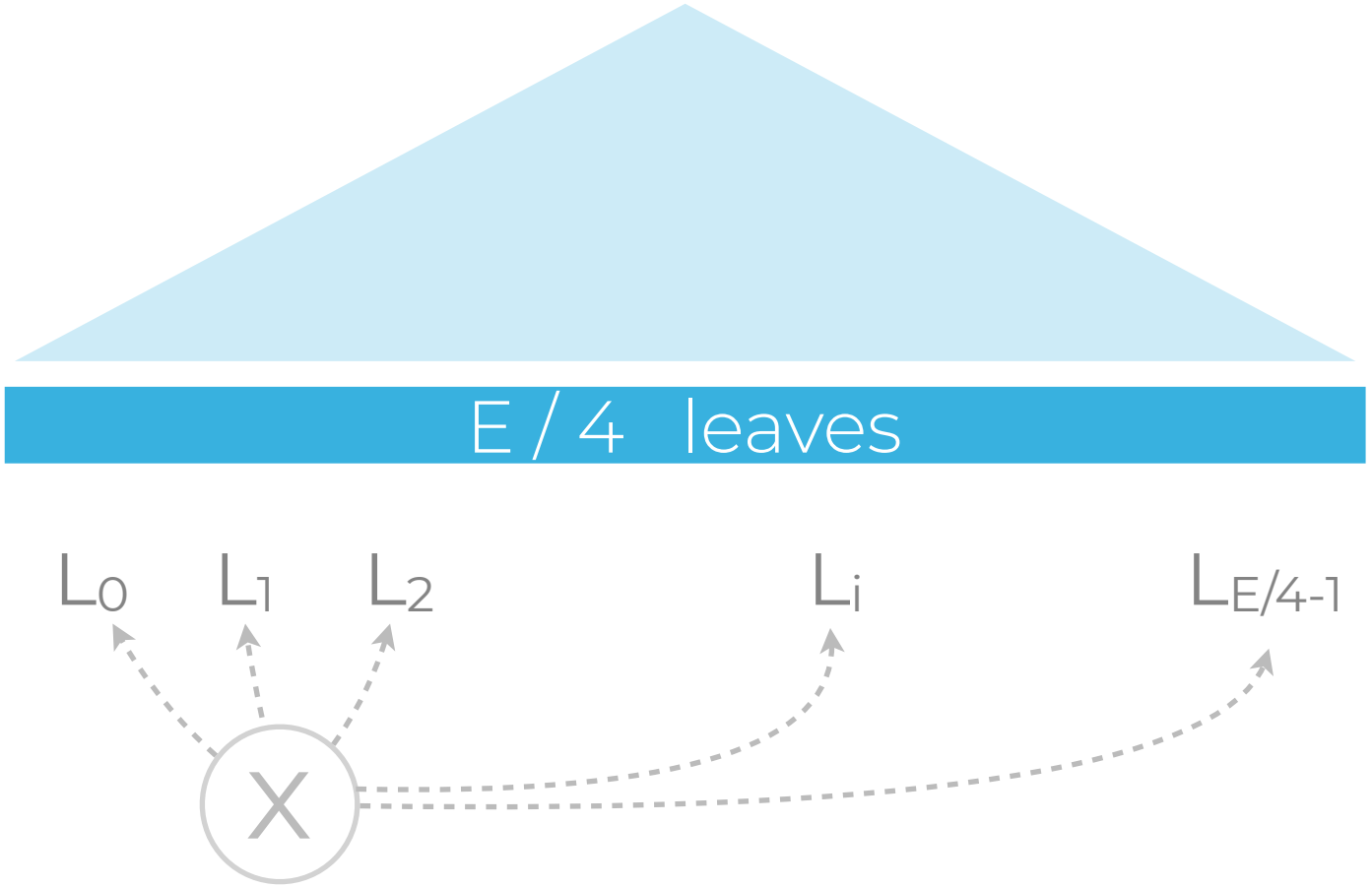
Commit phase



© QEDIT 2019

Fast Reed Solomon IOP Proximity

Commit phase



© QEDIT 2019

Fast Reed Solomon IOP Proximity

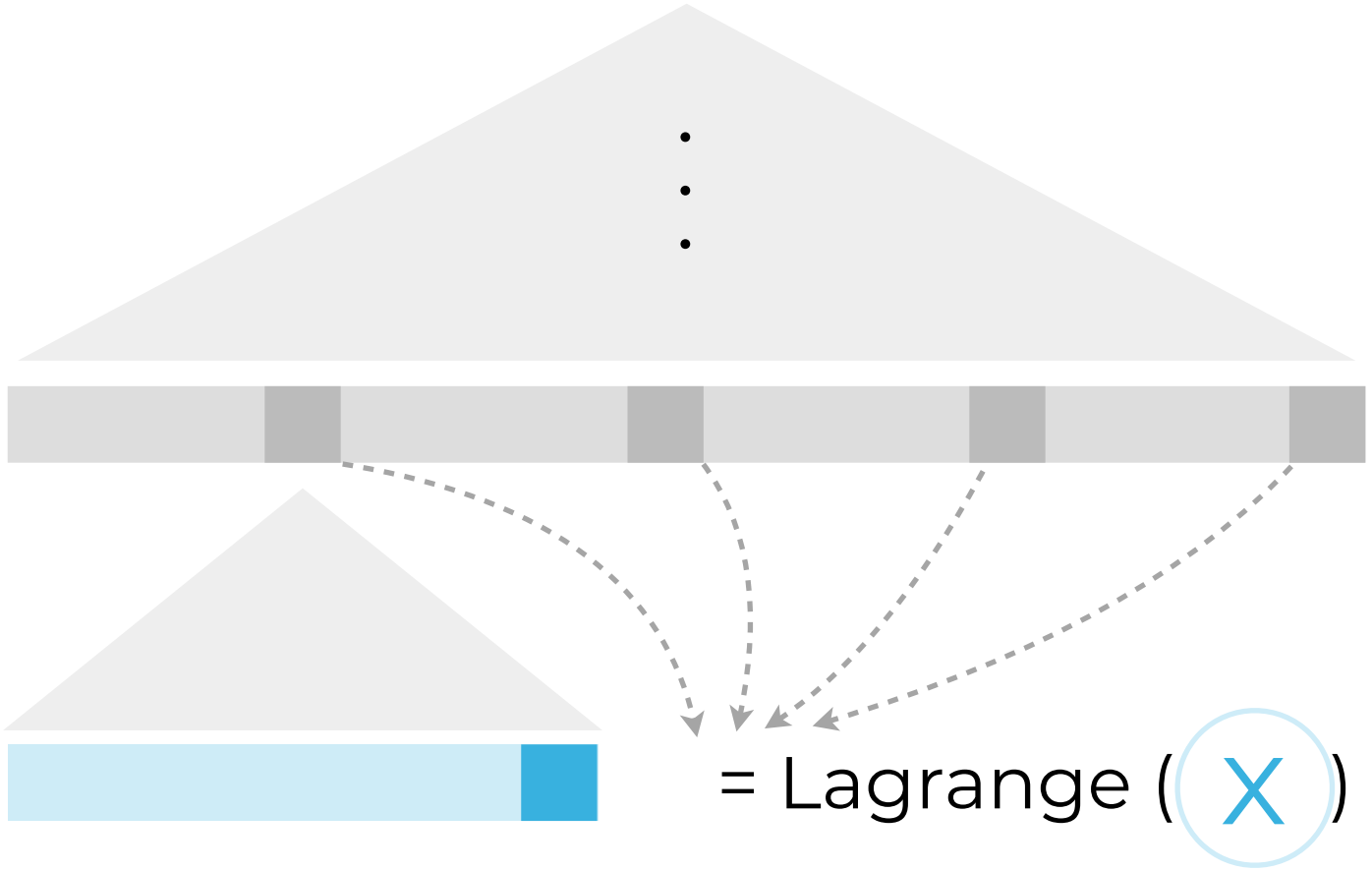
Commit phase

constant polynomial

© QEDIT 2019

Fast Reed Solomon IOP Proximity

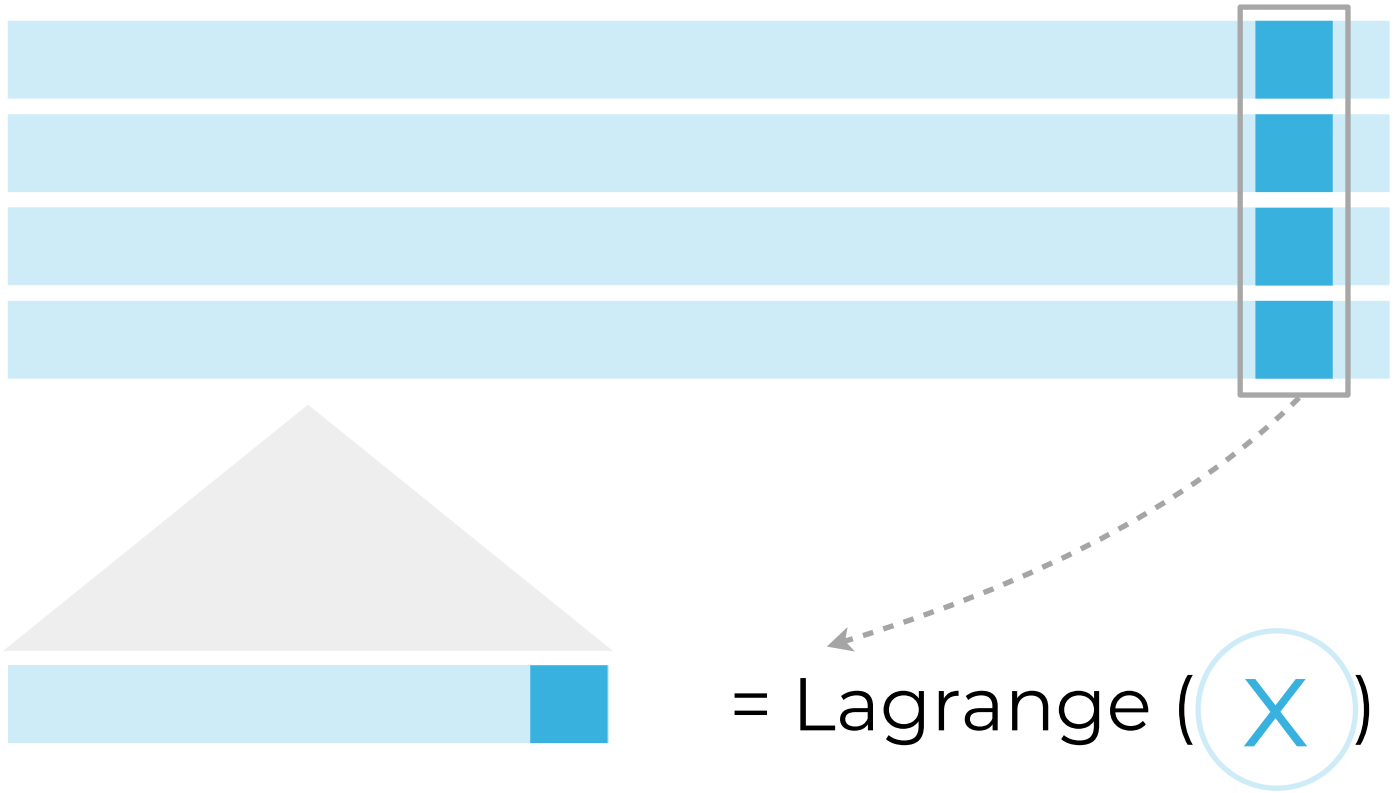
Query phase



© QEDIT 2019

Fast Reed Solomon IOP Proximity

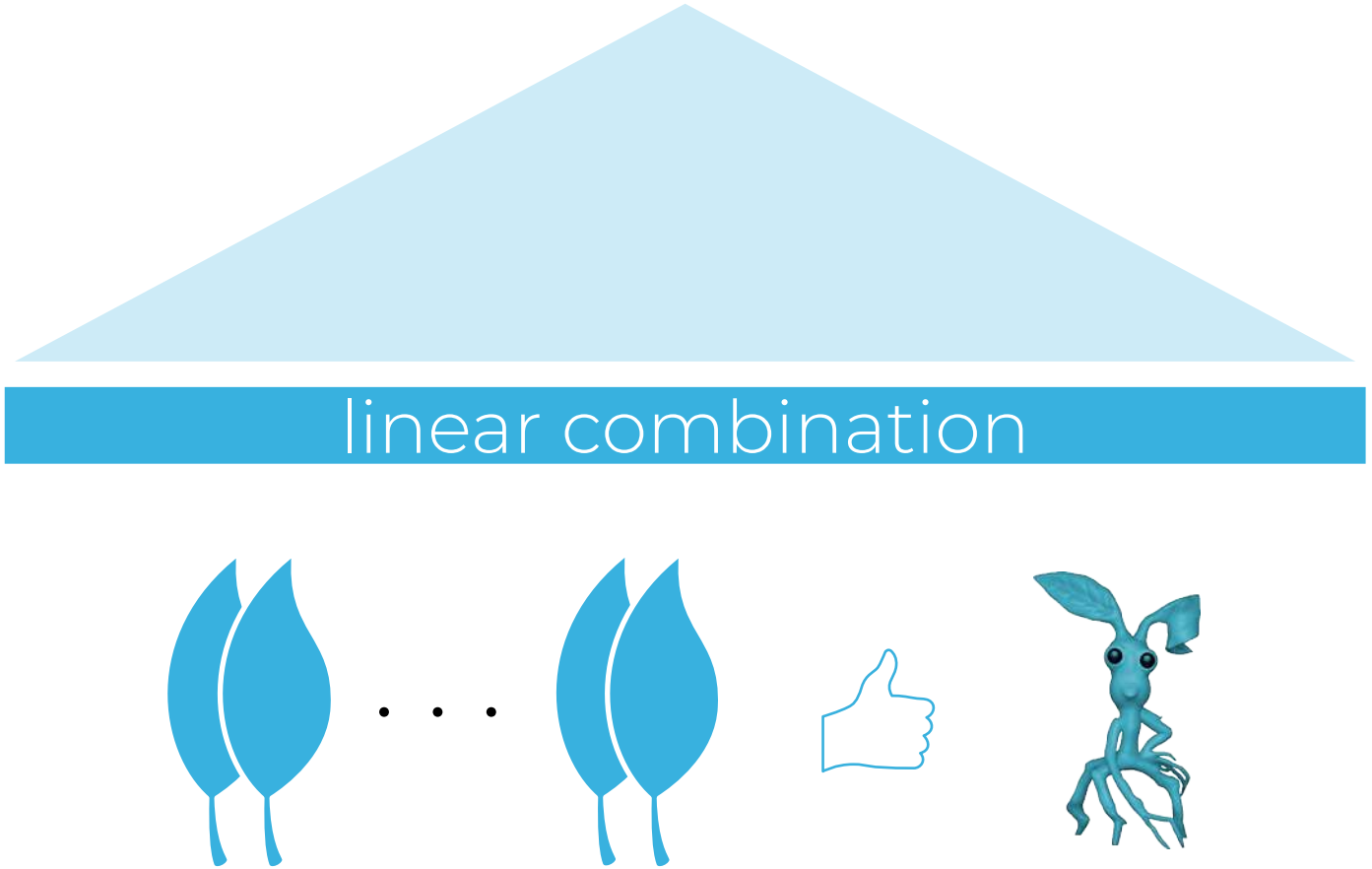
Query phase



© QEDIT 2019

Fast Reed Solomon IOP Proximity

Query phase



© QEDIT 2019

/ GuildOfWeavers / genSTARK

/ elibensasson / libSTARK

/ matter-labs / hodor

© QEDIT 2019

genSTARK example

Pedersen commitments

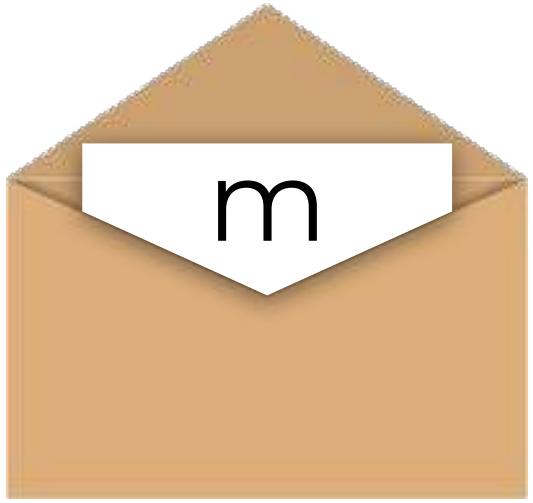
$$[H^r \cdot G^m]_p$$

© QEDIT 2019

genSTARK example

Pedersen commitments

$$[H^r \cdot G^m]_p$$



© QEDIT 2019

genSTARK example

Pedersen commitments

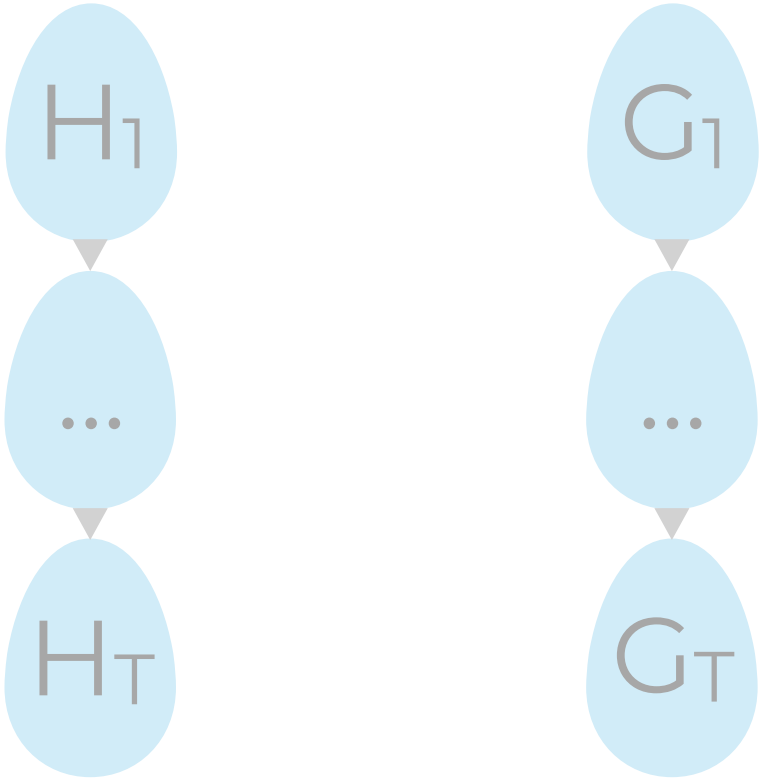
$$[H^r \cdot G^m]_p$$

© QEDIT 2019

genSTARK example

Pedersen commitments

$$[H^r \cdot G^m]_p$$

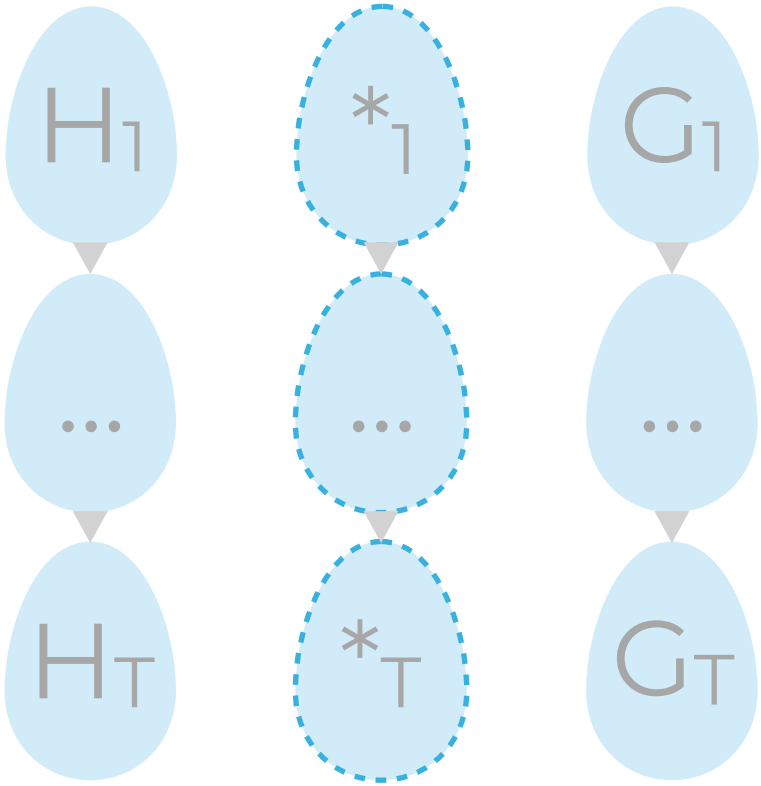


© QEDIT 2019

genSTARK example

Pedersen commitments

$$[H^r \cdot G^m]_p$$

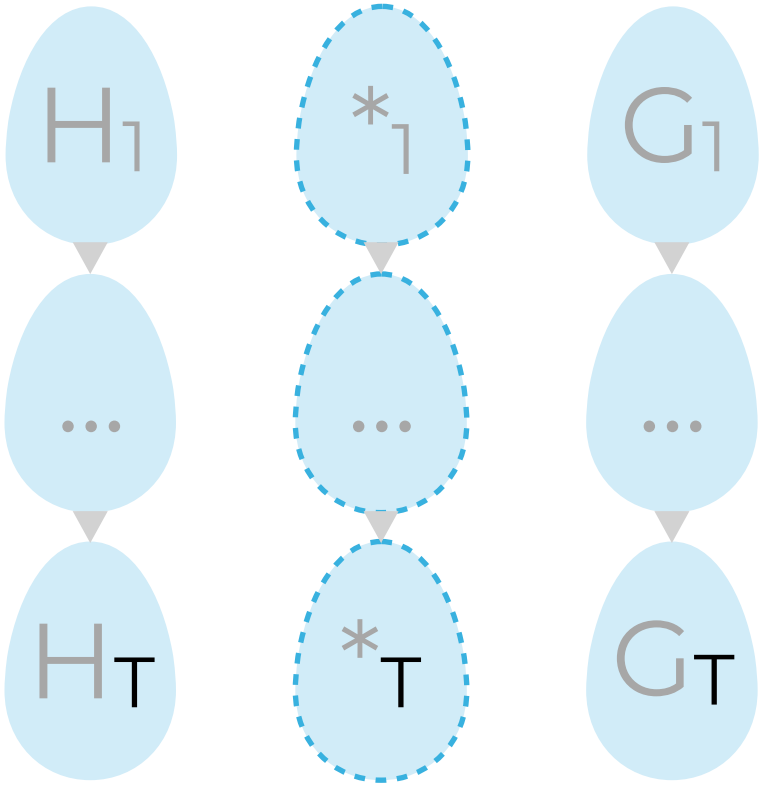


© QEDIT 2019

genSTARK example

Pedersen commitments

$$[H^r \cdot G^m]_p$$



© QEDIT 2019

genSTARK complexity

λ / log (e T/d)

© QEDIT 2019

genSTARK complexity

$O(w\ T\ e \log (T\ e+|F|))$

$\lambda / \log (e\ T/d)$

© QEDIT 2019

genSTARK complexity

$O(w\ T\ e \log (T\ e+|F|))$

$O(q \log^2 e\ T)$

$\lambda / \log (e\ T/d)$

© QEDIT 2019

genSTARK complexity

$O(w\ T\ e \log (T\ e+|F|))$

$O(q \log^2 e\ T)$

$O(q \log^2 e\ T)$

$\lambda / \log (e\ T/d)$

© QEDIT 2019

genSTARK complexity

$O(w\ T\ e \log (T\ e+|F|))$

$O(q \log^2 e\ T)$

$O(q \log^2 e\ T)$

$O(e\ T)$

$\lambda / \log (e\ T/d)$

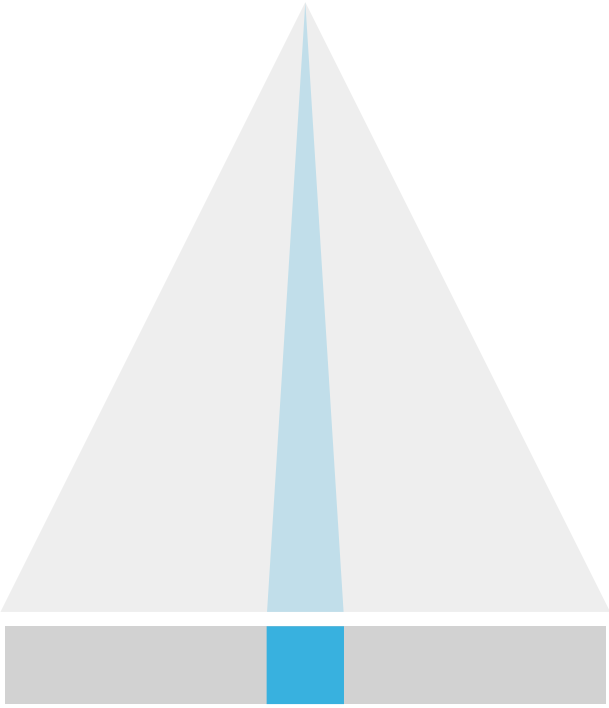
© QEDIT 2019

genSTARK benchmark

Membership

- 32 commitments
- 256 bit field size
- 256 bit exponent

- 15 s prover
- 2 s verifier
- 700 KB proof
- 600 KB trees
- 200 MB RAM



© QEDIT 2019

genSTARK benchmark

Bulk

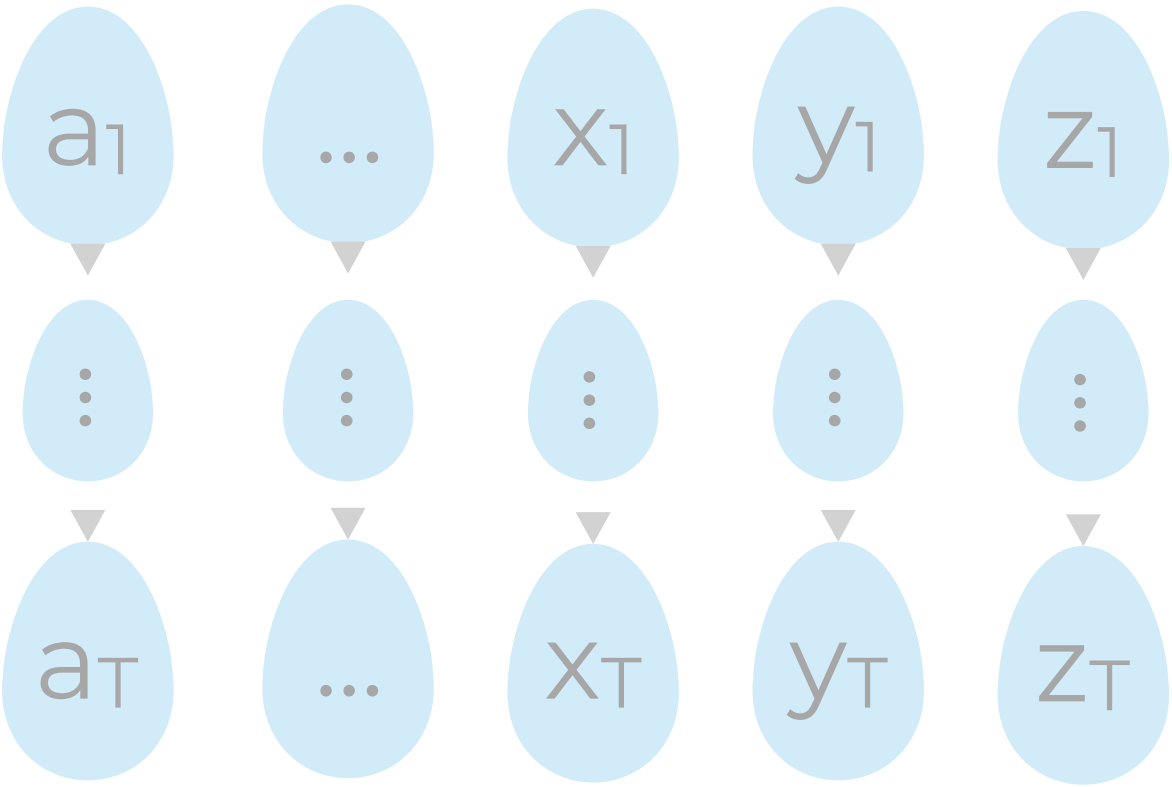
- 1024 commitments
- 32 bit field size
- 256 bit exponent

- 200 s prover
- 1 ms verifier
- 2 MB proof
- 500 KB trees
- 3 GB RAM

© QEDIT 2019

genSTARK benchmark

Tradeoff



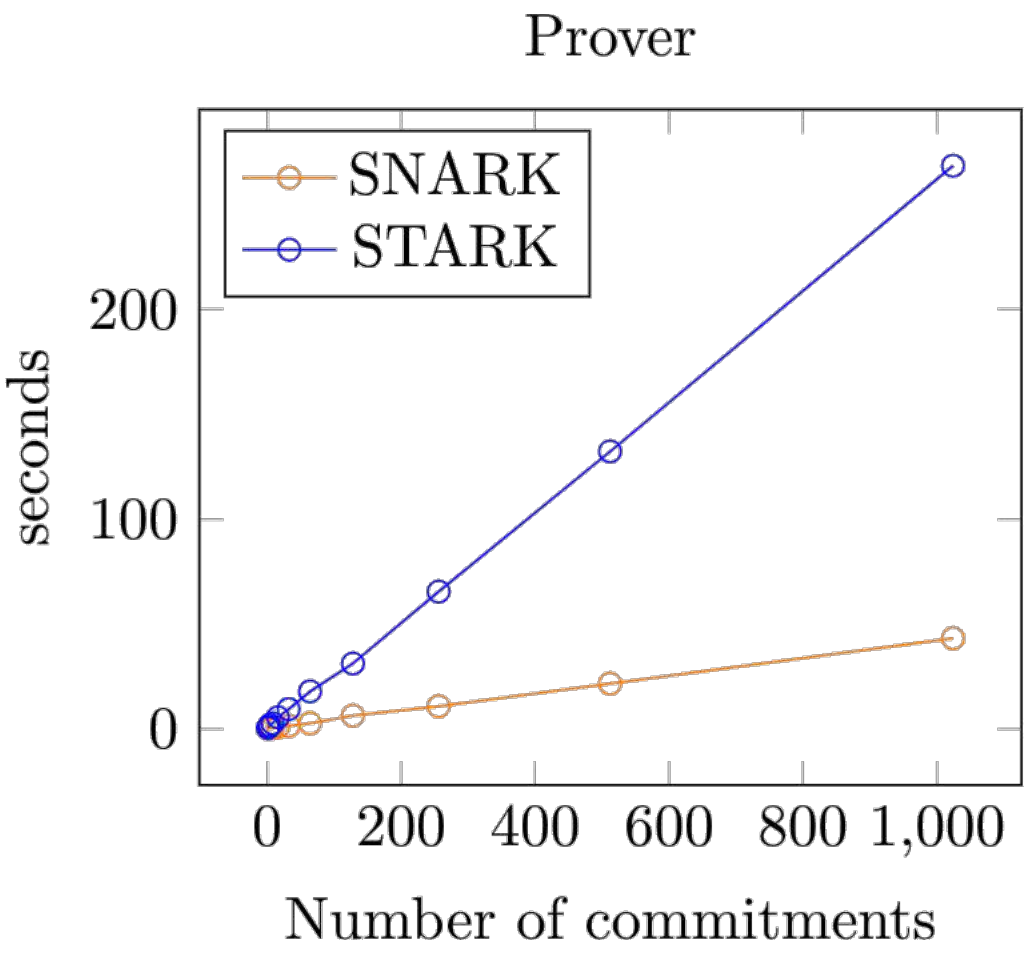
© QEDIT 2019

STARK vs SNARK

an eggs to apples comparison

STARK

- JavaScript genSTARK
- Amateur library
- 256 bit field
- Transparent
- Polylog proof in steps

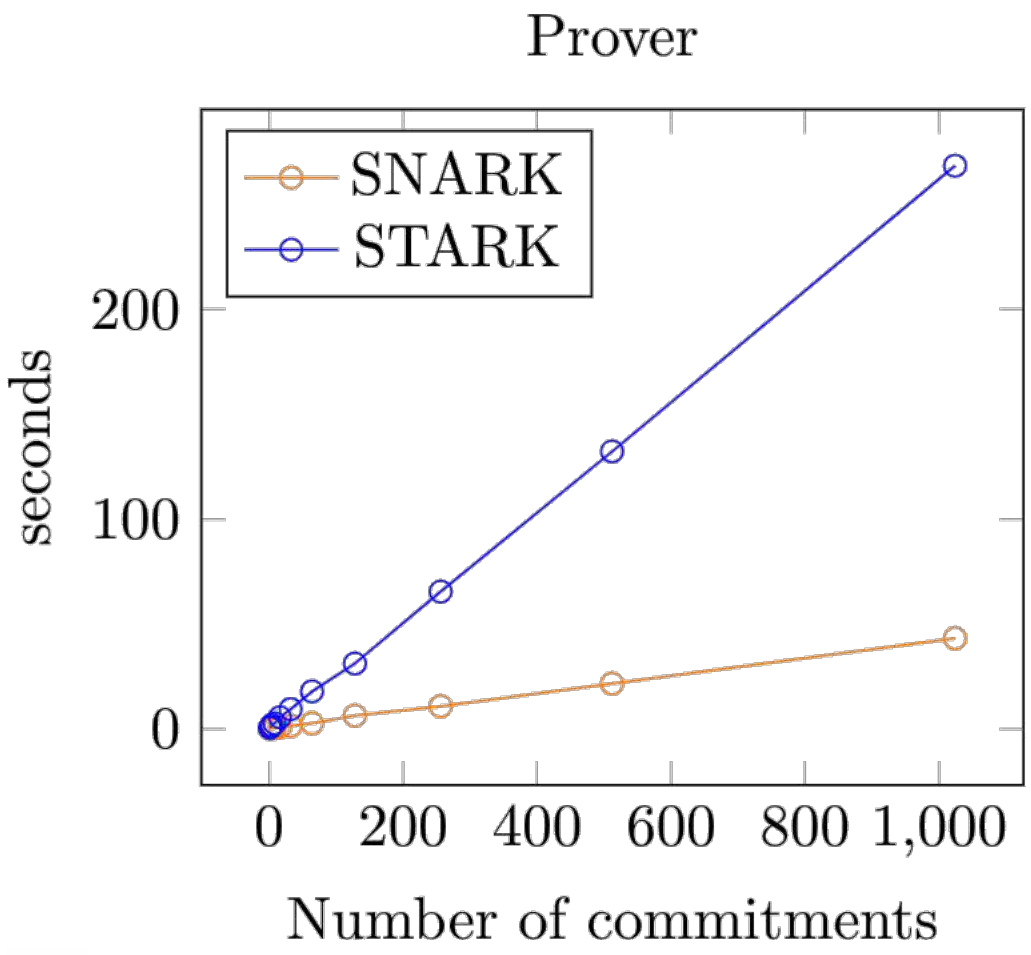


SNARK

- Rust librustzcash
- Optimized library
- JubJub elliptic curve
- Trusted setup
- Constant proof size

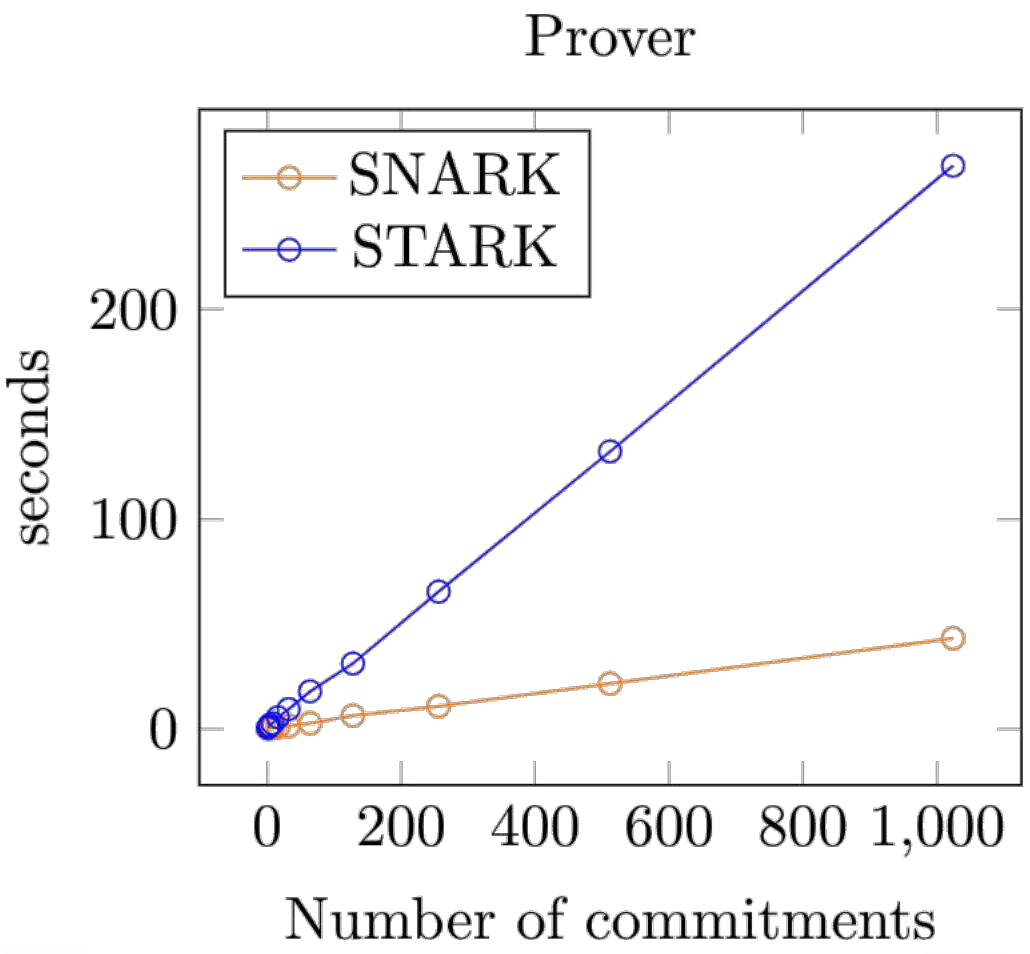
© QEDIT 2019

So, should we put our **coins** here?



© QEDIT 2019

So, should we put our **coins** here?



Never put all **eggs** in one basket!

© QEDIT 2019

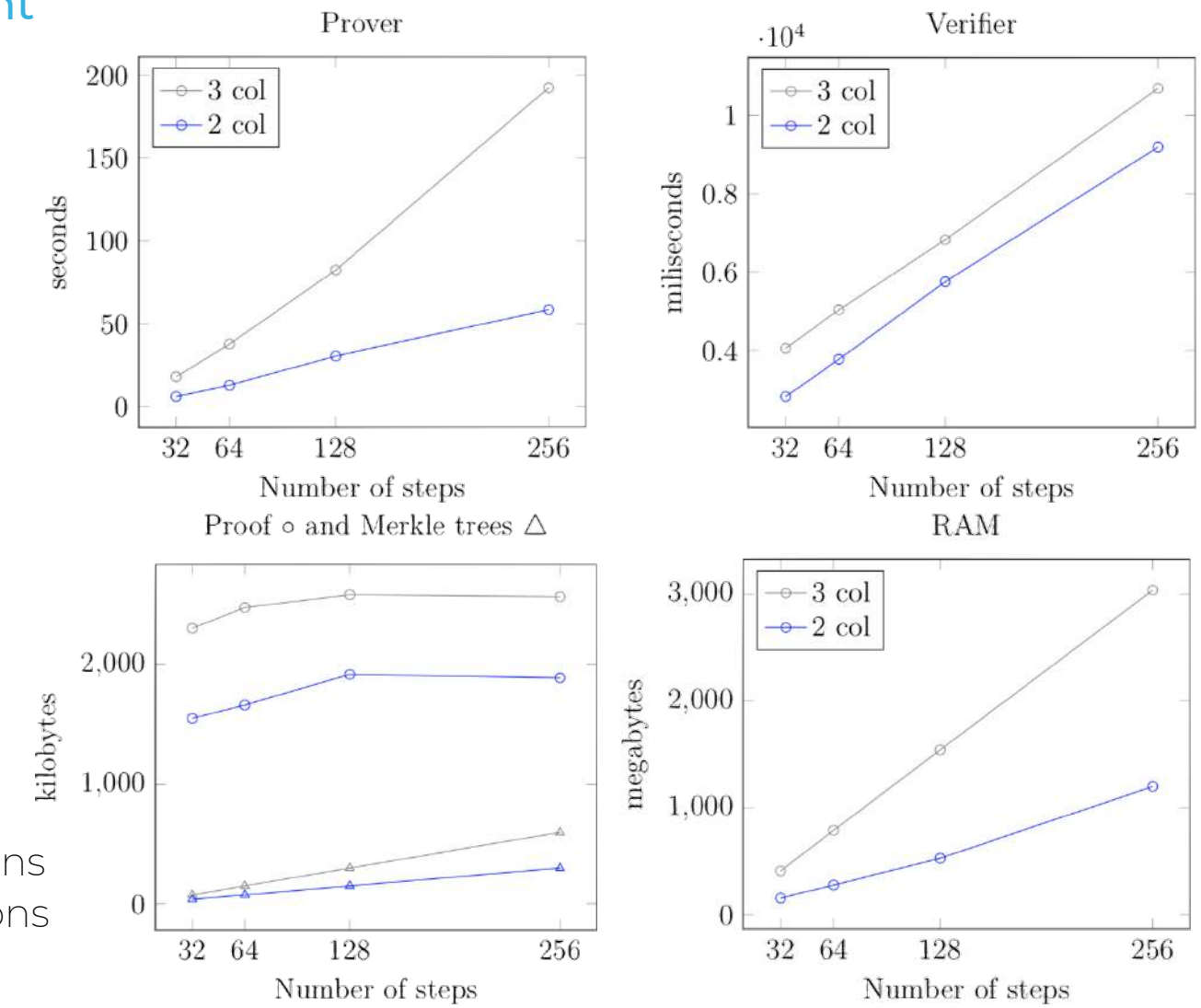
So, should we put our **coins** here?

any Other Inquiry Left (**OIL**)?

Never put all **eggs** in one basket!

genSTARK benchmark

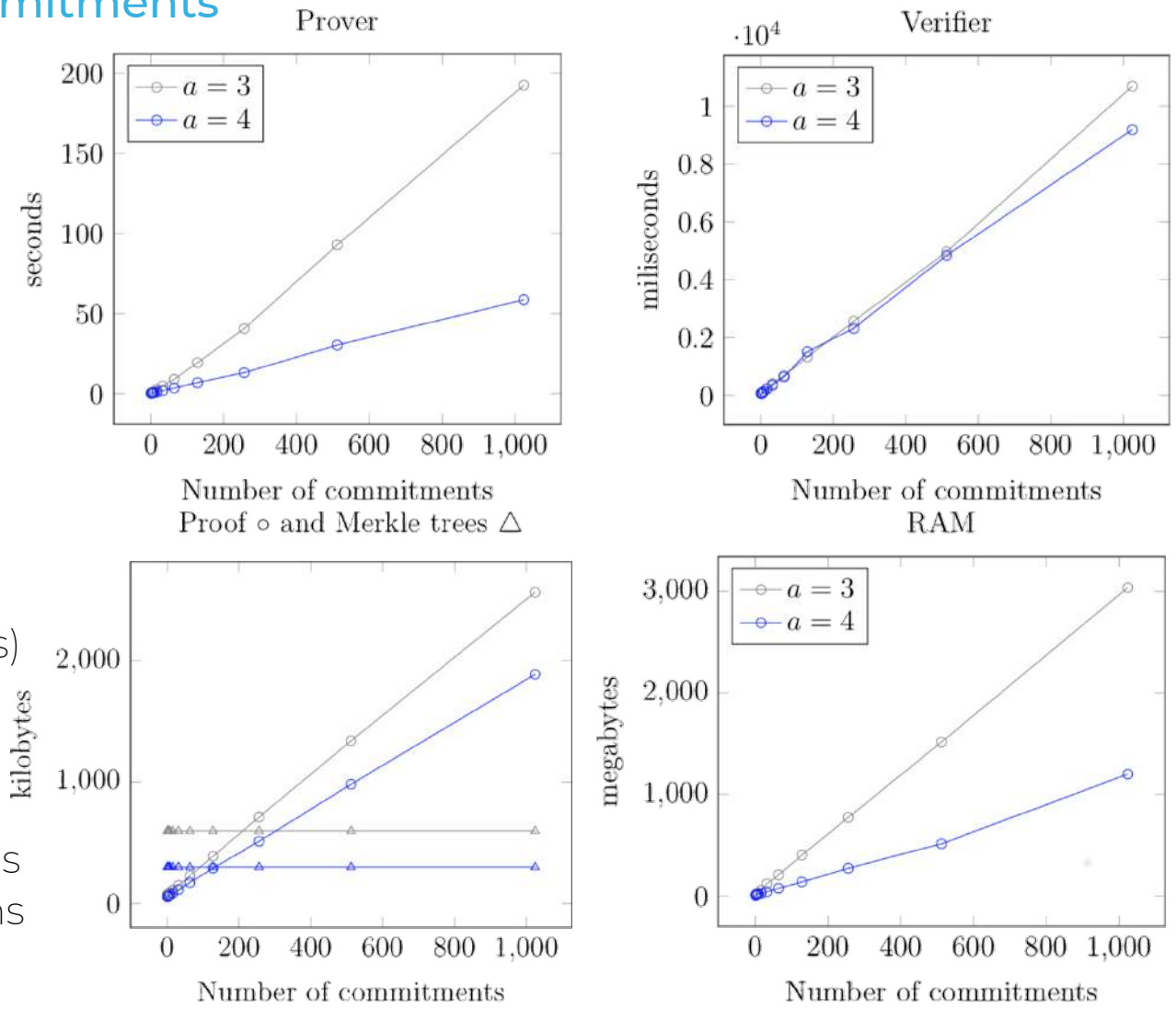
Bits in exponent



- Steps (rows)
- 1024 Pedersen
- 32 bit field
- 80 queries
- 3 col 3 assertions
- 2 col 4 assertions

genSTARK benchmark

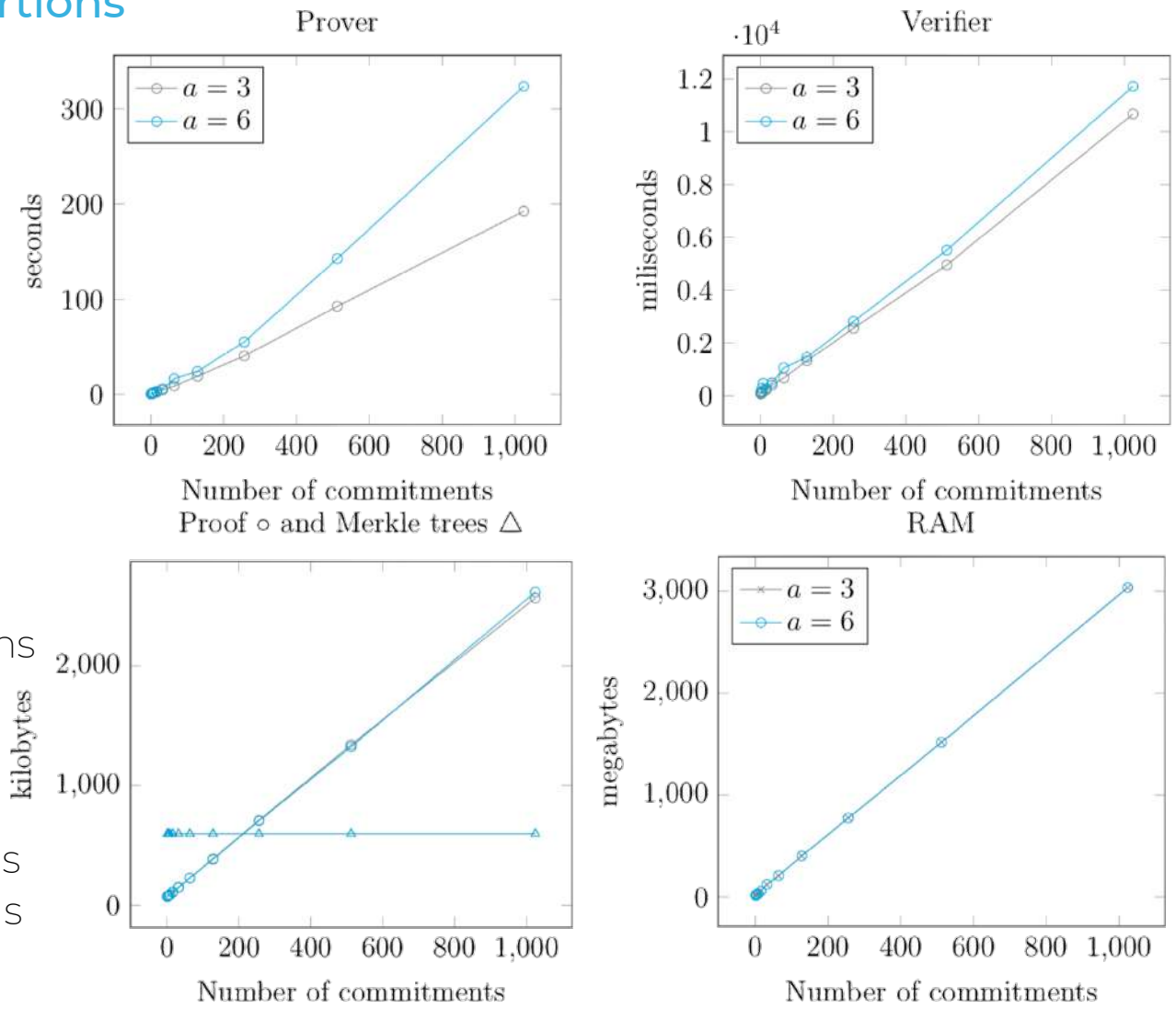
Number of commitments



- Width (columns)
- 256 exponent
- 32 bit field
- 80 queries
- 3 col 3 assertions
- 2 col 4 assertions

genSTARK benchmark

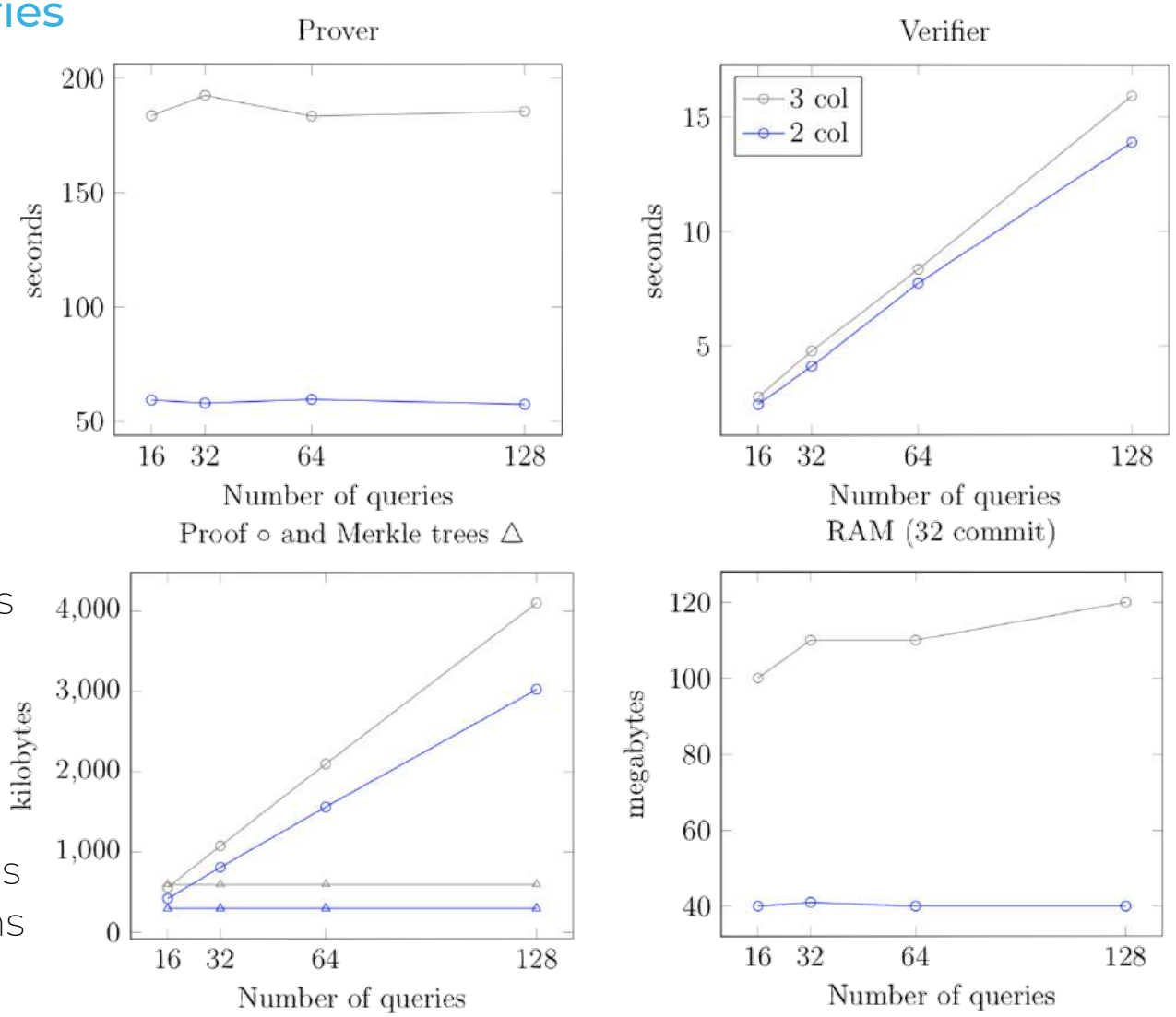
Number of assertions



- Multiple columns
- 256 exponent
- 32 bit field
- 80 queries
- 3 col 3 assertions
- 3 col 6 assertions

genSTARK benchmark

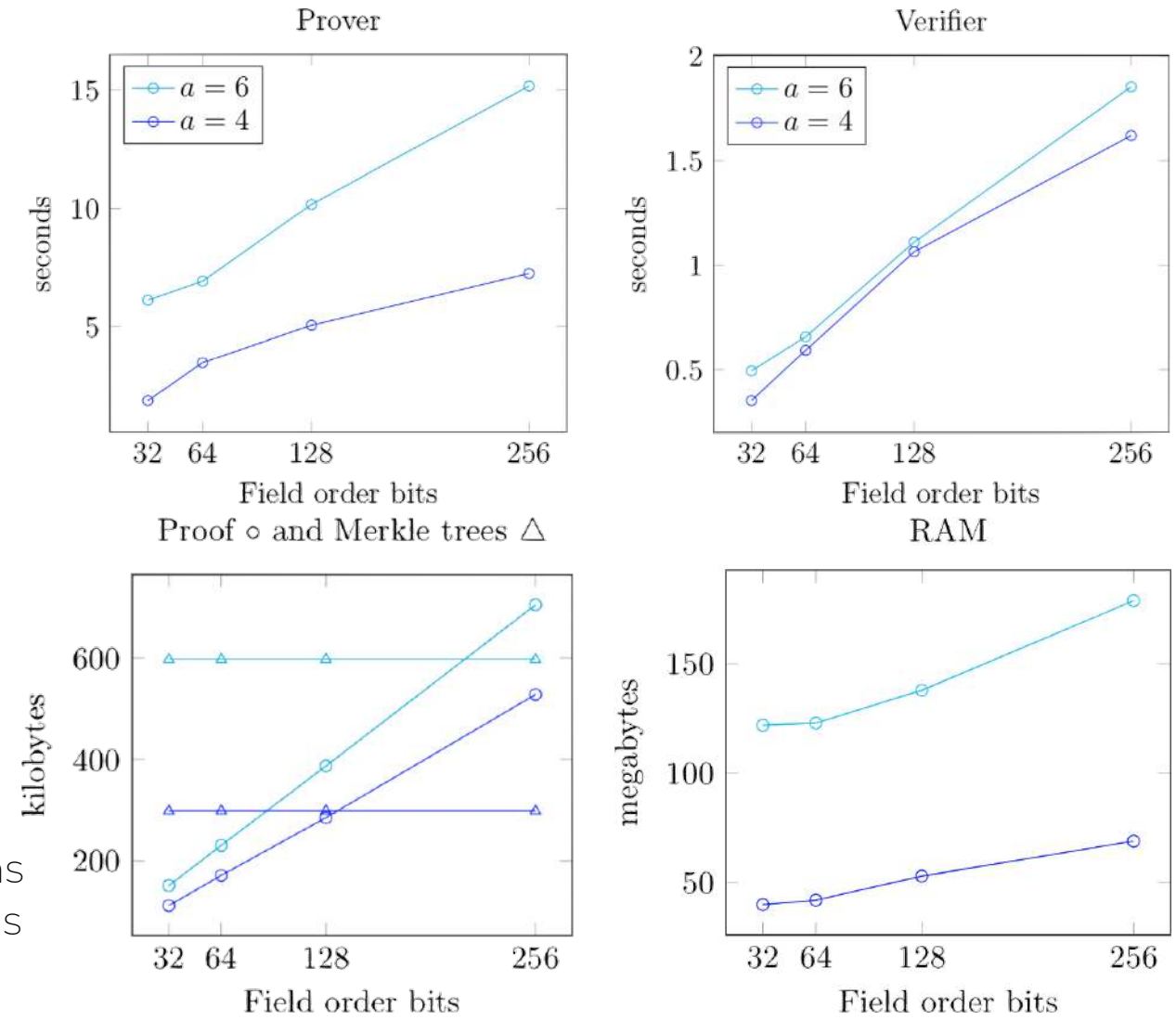
Number of queries



- Checking points
- 256 exponent
- 1024 Pedersen
- 32 bit field
- RAM only 32
- 3 col 3 assertions
- 2 col 4 assertions

genSTARK benchmark

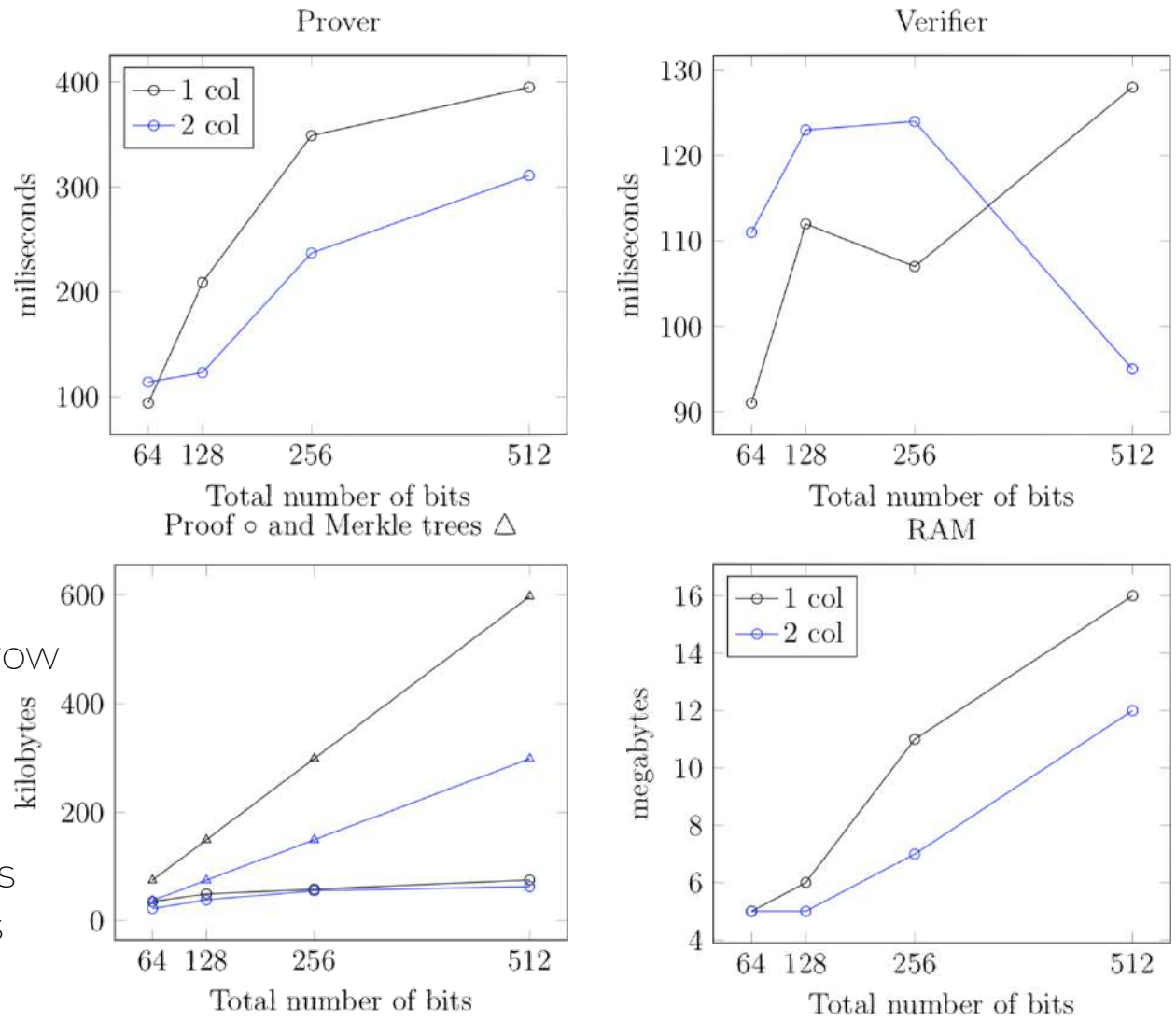
Field size



- Element size
- 256 exponent
- 32 Pedersen
- 80 queries
- 2 col 4 assertions
- 3 col 6 assertions

genSTARK benchmark

Execution trace



- Trade-off col vs row
- 1 Pedersen
- 256 bit field
- 80 queries
- 2 col 4 assertions
- 1 col 2 assertions