CONSENSYS

Linea

zkAggregation and application to Linea

zkBelgrade — 06/2024

Youssef El Housni

Team

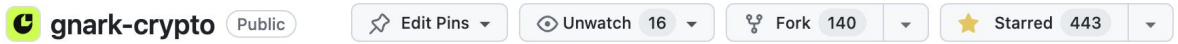
Who?

- Arya Pourtabatabaie
- Ivo Kubjas
- *Youssef El Housni*
- Thomas Piellard
- Gautam Botrel

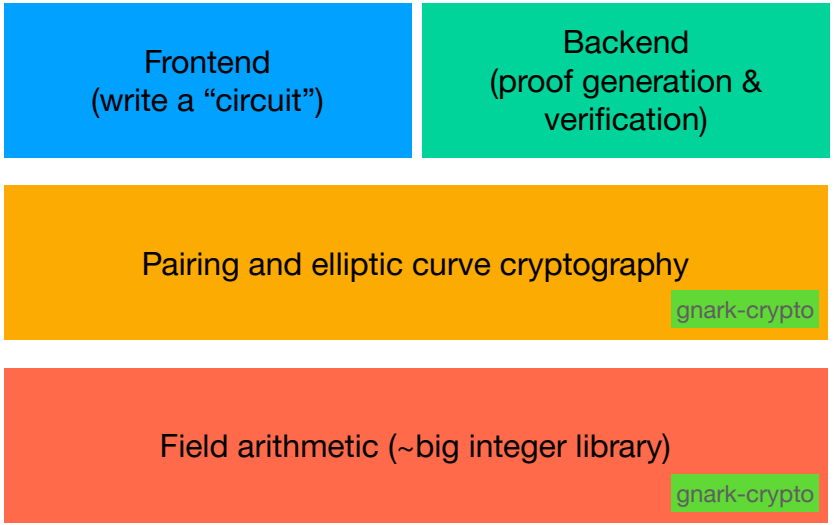
What?

We're building gnark, a fast and easy to use open source zkSNARK library, in Go.

and gnark-crypto, a fast cryptographic library, in Go.



gnark under the hood



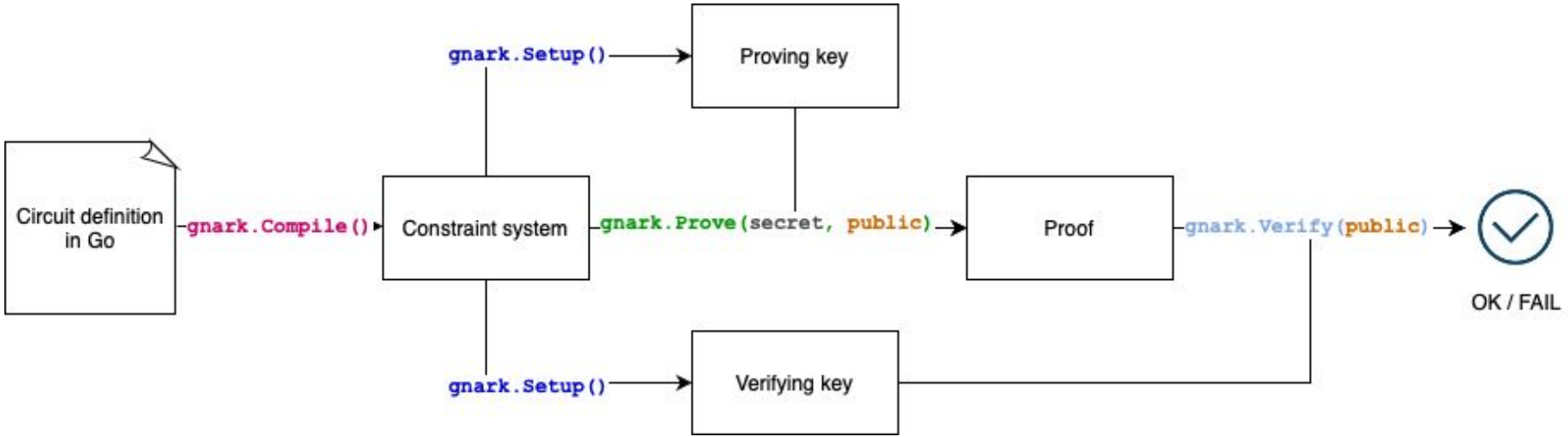
- Groth16, PLONK w/ KZG (or FRI)
- std: hashes, signatures, pairings, commitments…
- Native and non-native field arithmetic

- BN254, BLS12-381, BLS12-377/BW6-761, BLS24…
- Fast cryptographic primitives (MSM, pairings,...)
- KZG, FRI, Plookup…
- Sumcheck (GKR)

- 768-bit, 384-bit, 256-bit, goldilocks… on multi-targets
- SotA mul, Pornin’s inverse, FFT…

gnark workflow

```
pk, vk, err := groth16.Setup(ccs)
proof, err := groth16.Prove(ccs, pk, witness)
err := groth16.Verify(proof, vk, publicWitness)
```



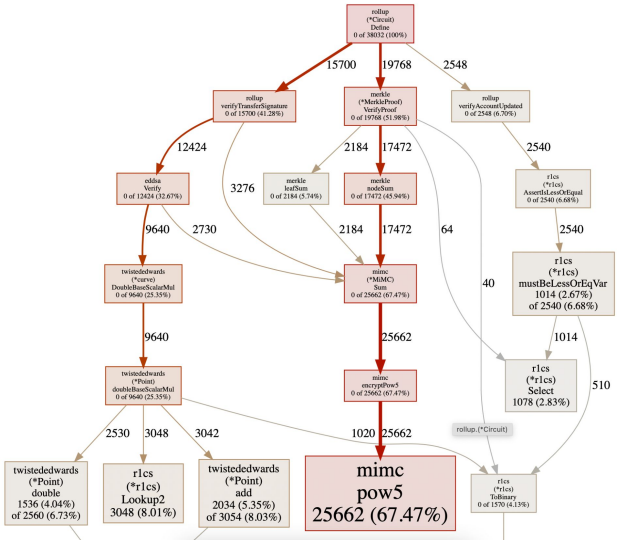

```
ccs, err = frontend.Compile(ecc.BN254.ScalarField(), r1cs.NewBuilder, &c)
ccs, err = frontend.Compile(ecc.BLS12_381.ScalarField(), scs.NewBuilder, &c)
```

gnark features

gnark

- No DSL, plain Go, **no dependencies**
- Compiles large circuit (seconds)
- Playground, constraints profiler, …
- multiple curves and backends
- MPC trusted setup
- Web2 and Solidity verification
- Several packages audited (by Algorand, EF, Worldcoin and Linea)
- One code base which performs well on:
  - Server (CPU)
  - Mobile (70% first place zprize)

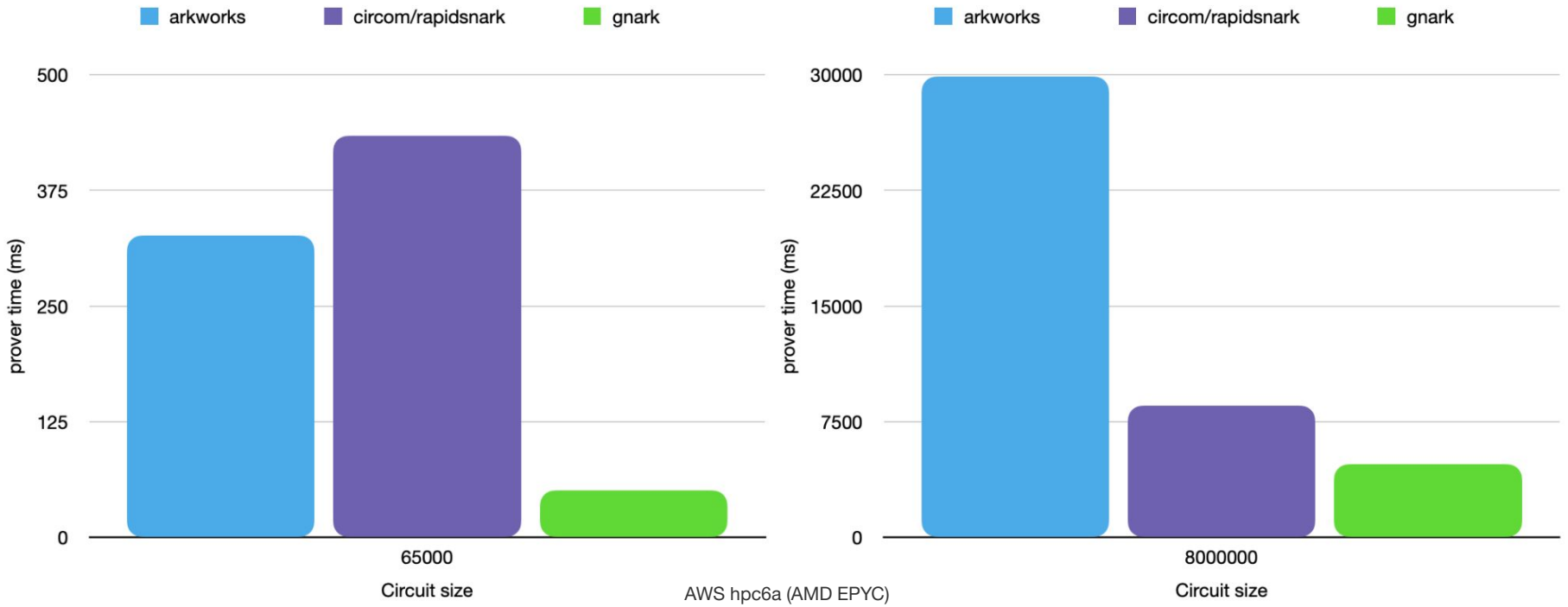
```
func (circuit *Circuit) Define(api frontend.API) error {
    // compute x**3 and store it in the local variable x3.
    x3 := api.Mul(circuit.X, circuit.X, circuit.X)

    // compute x**3 + x + 5 and store it in the local variable res
    res := api.Add(x3, circuit.X, 5)

    // assert that the statement x**3 + x + 5 == y is true.
    api.AssertIsEqual(circuit.Y, res)
```



Constraints profiler

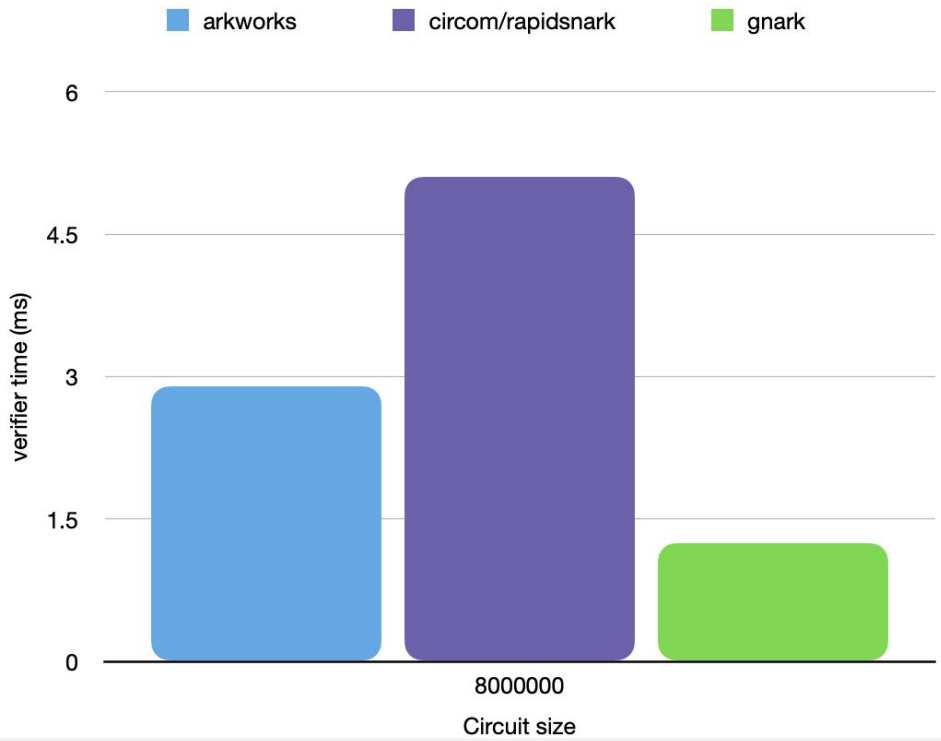
gnark is very fast

Groth16 SNARK prover on BN254: MSM, FFT, parallelism



gnark is very fast

Groth16 SNARK verifier: Pairing on BN254

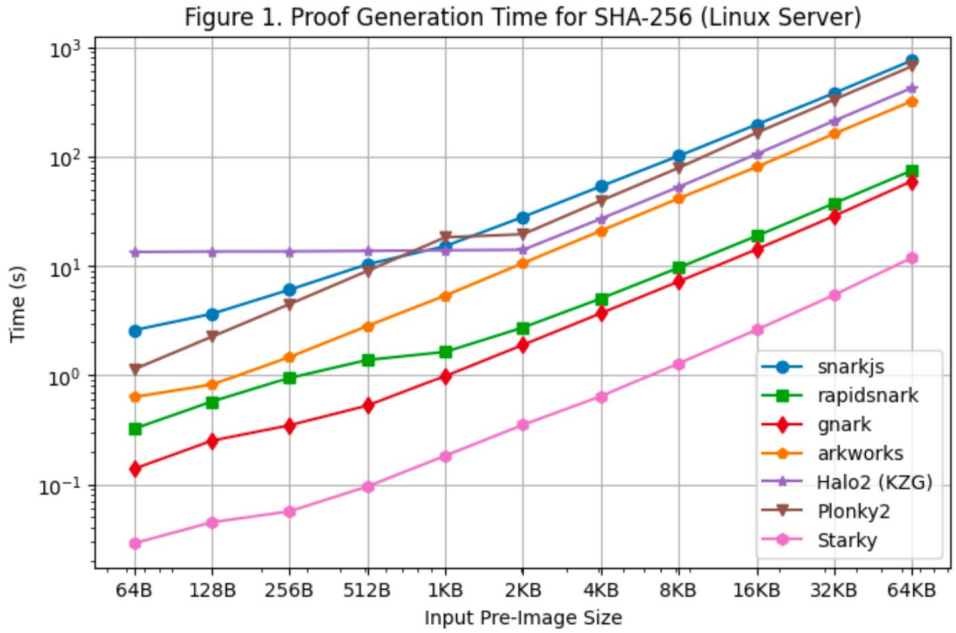


AWS hpc6a (AMD EPYC)

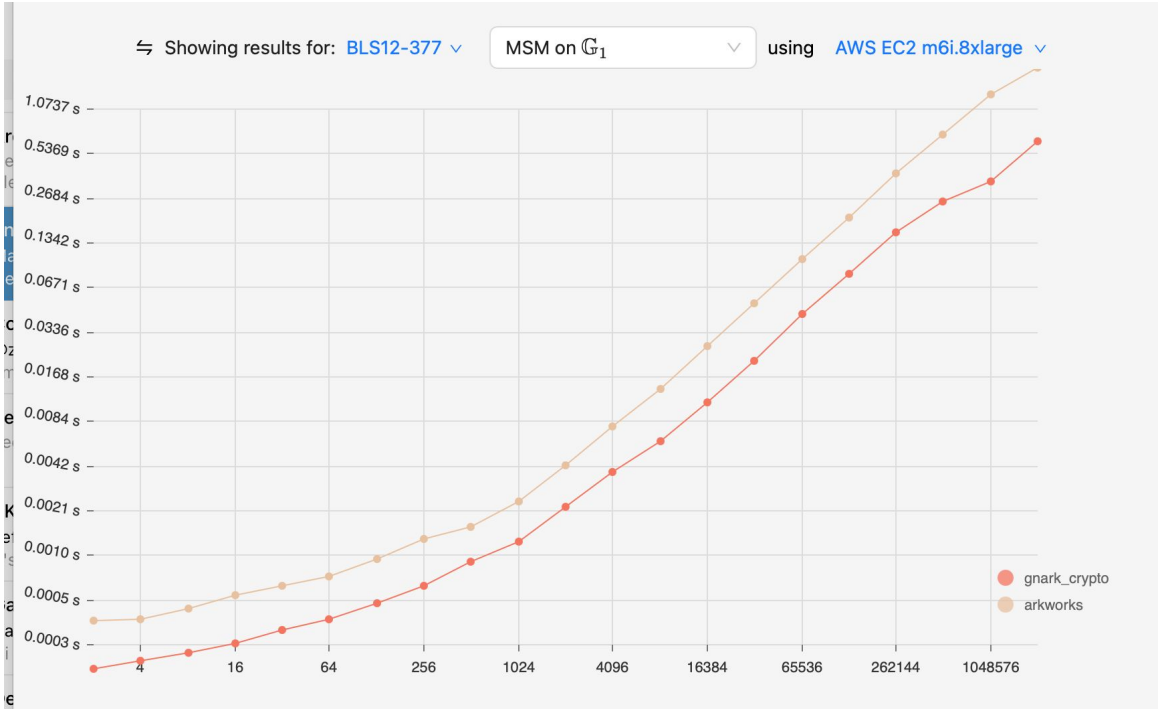
gnark is very fast (celer benchmark)

SHA2 preimage knowledge

https://github.com/celer-network/zk-benchmark



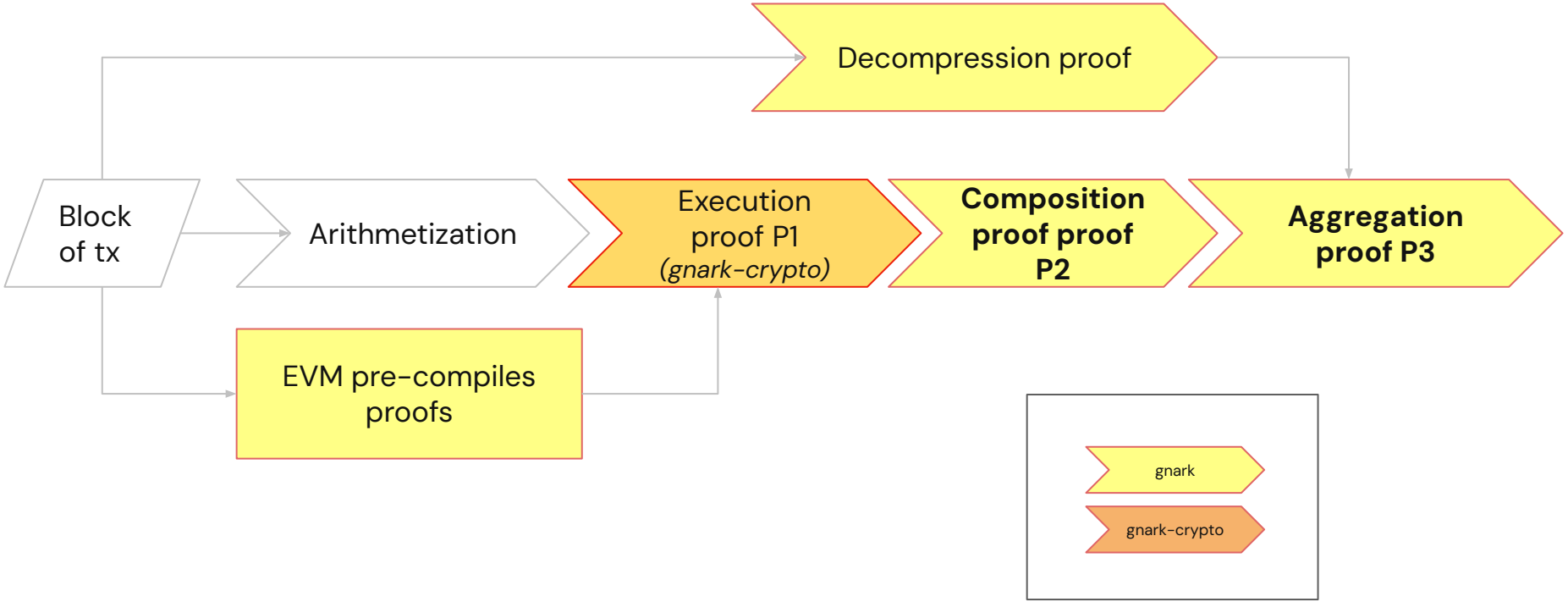
gnark is very fast (zka.lc benchmark)



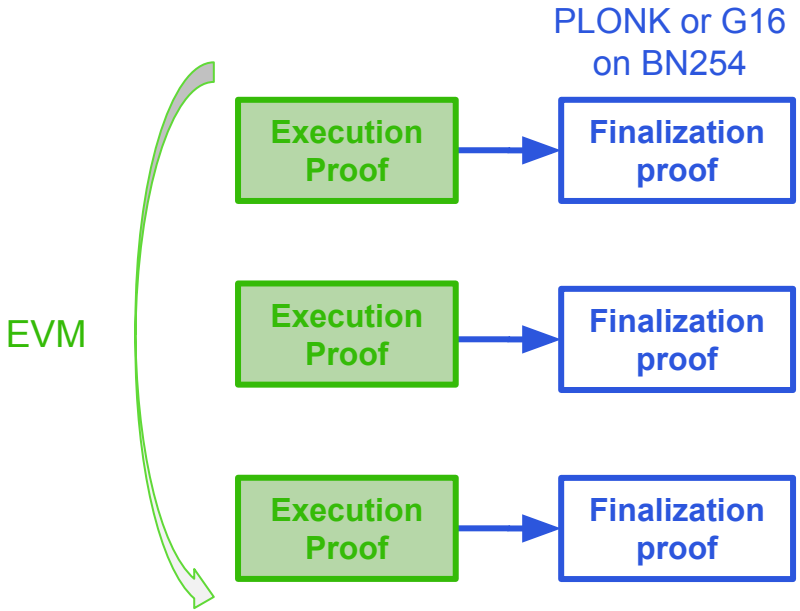
gnark applications

- **zkEVMs (Linea)**
- zkVM (SP1)
- Rollups (zkBNB)
- Binance proof of solvency
- Proof system (Worldcoin Groth16)
- zkBridge (Celer)
- zkCoprocessor (Brevis, Lagrange)
- *Classical cryptography: Algorand, EIP-4844 (go-kzg) and EIP-2537 (geth)*
- Blockchain voting (Vocodoni)
- Ingonyama (hardware accelerator): GPU support for Groth16 and PlonK.
- Some formal verification (Lean) on gnark circuits.
- Base: keystores, FFLONK
- …

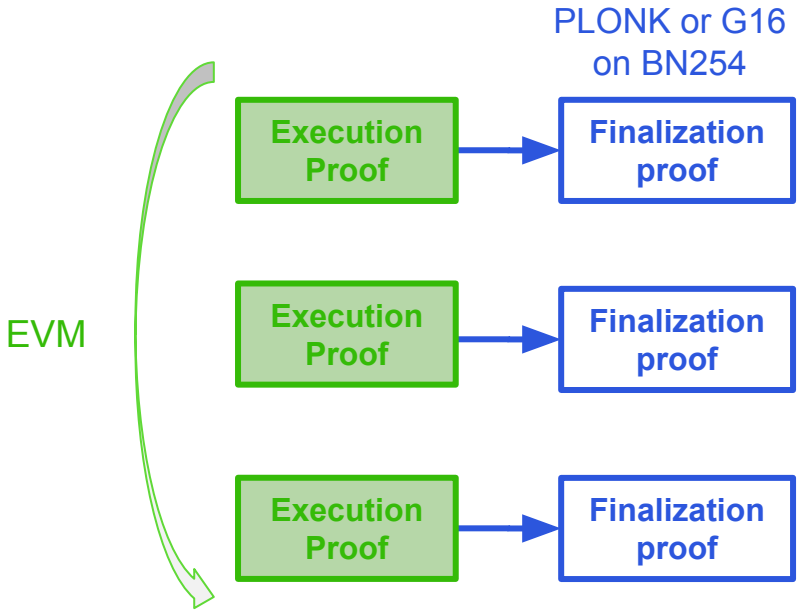
gnark in Linea



zk-proving the EVM



zk-proving the EVM



Execution Proof

- Polygon → plonky2
- zkSync → plonky2 (*Boojum*)
- Scroll → Halo2-KZG

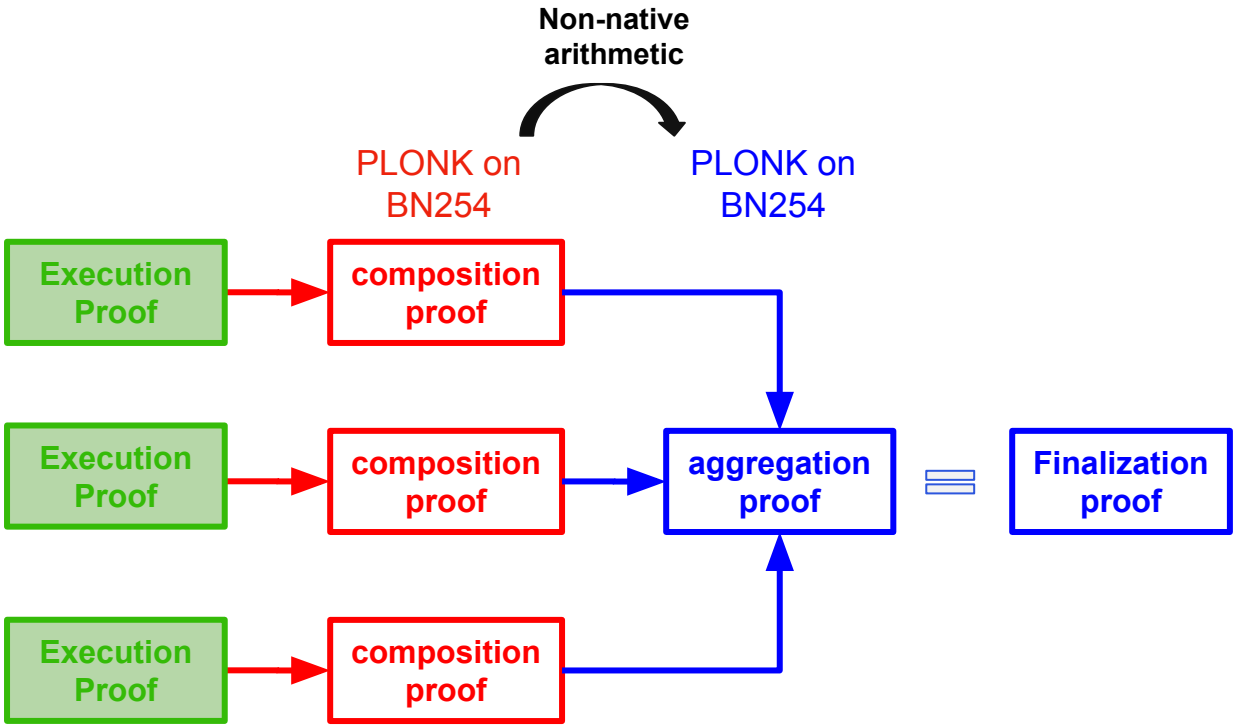
- **Linea → Vortex**

Vortex: A List Polynomial Commitment and its Application to Arguments of Knowledge

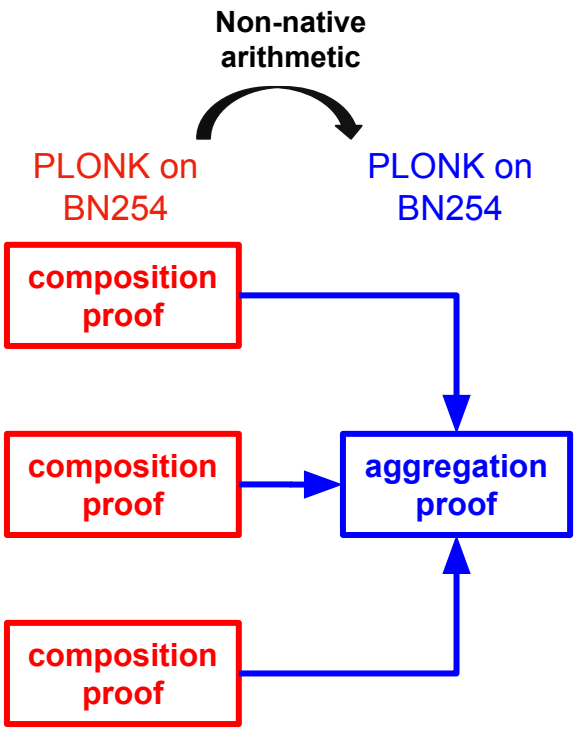
Alexandre Belling, Azam Soleimanian, and Bogdan Ursu

Linea, Prover Team
{firstname.lastname}@consensys.net

zkAggregation



zkAggregation



Consensys / **gnark**

Bottlenecks:

- Non-native field arithmetic

  gnark / std / math / **emulated** /

  https://hackmd.io/@ivokub/SyJRV7ye2
  https://hackmd.io/@ivokub/SJZOLa382

- MSM in-circuit

  gnark / std / algebra / emulated / sw_emulated / **point.go**

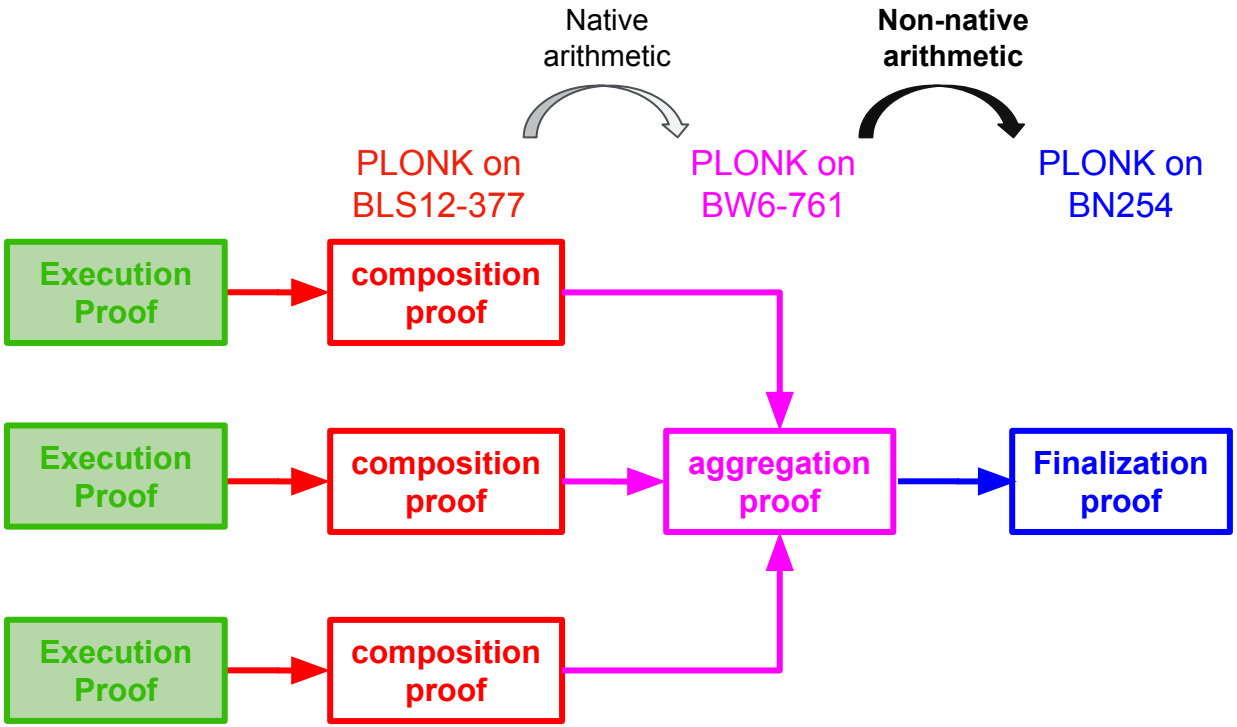
  optimized version of algorithm 1 of [Halo] (Section 6.2 and appendix C)
  [Halo]: https://eprint.iacr.org/2019/1021.pdf

- Pairing in-circuit

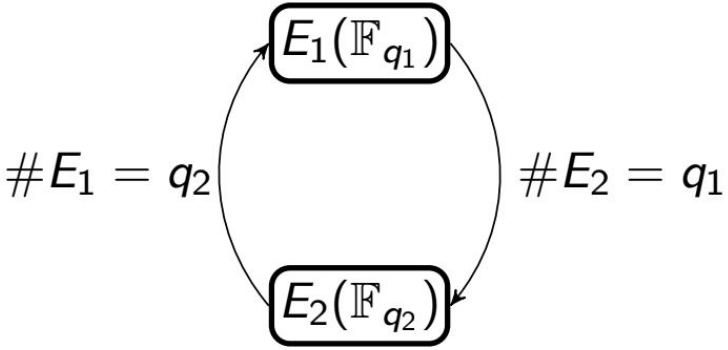
  gnark / std / algebra / emulated / **sw_bn254** /

  Pairings in R1CS: https://eprint.iacr.org/2022/1162.pdf
  On Proving Pairings: https://eprint.iacr.org/2024/640.pdf

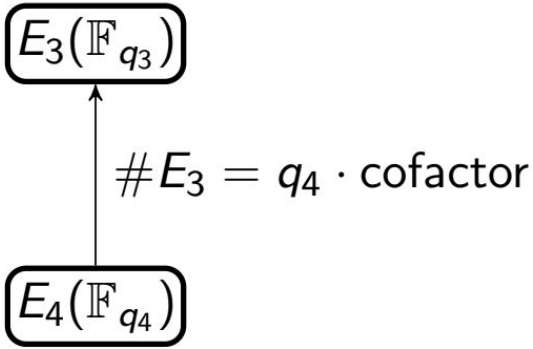
1-layer 2-chain zkAggregation



2-cycle and 2-chain

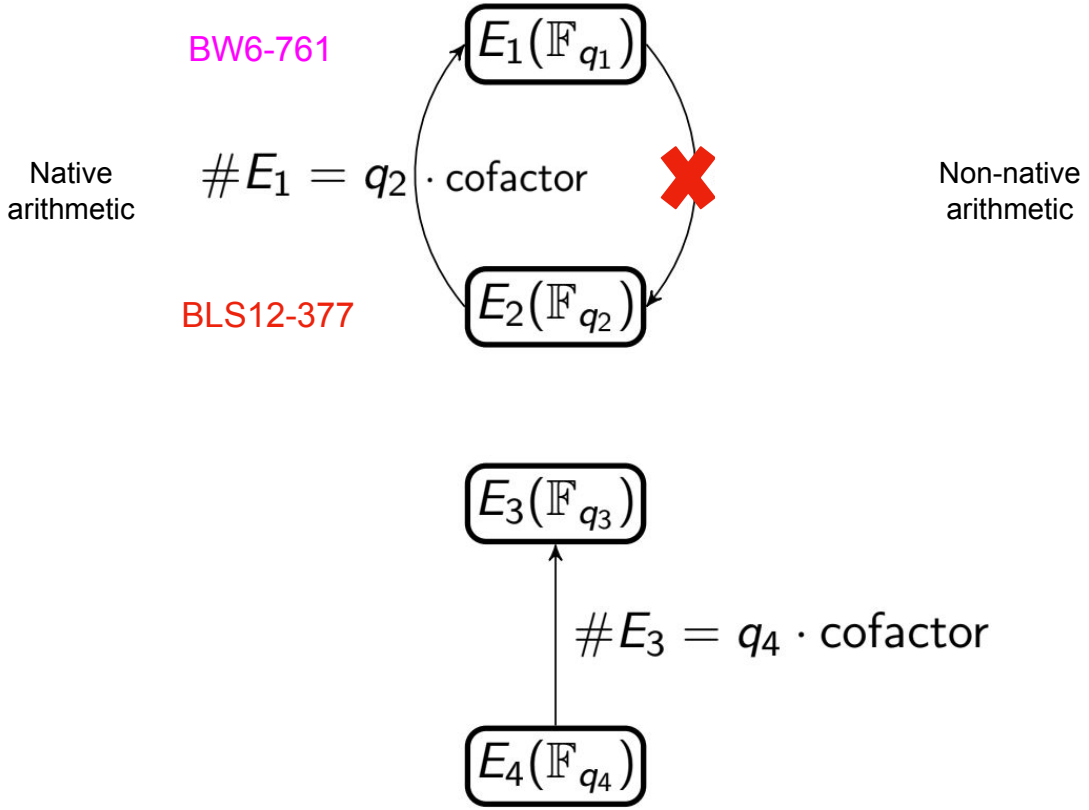


BW6-761

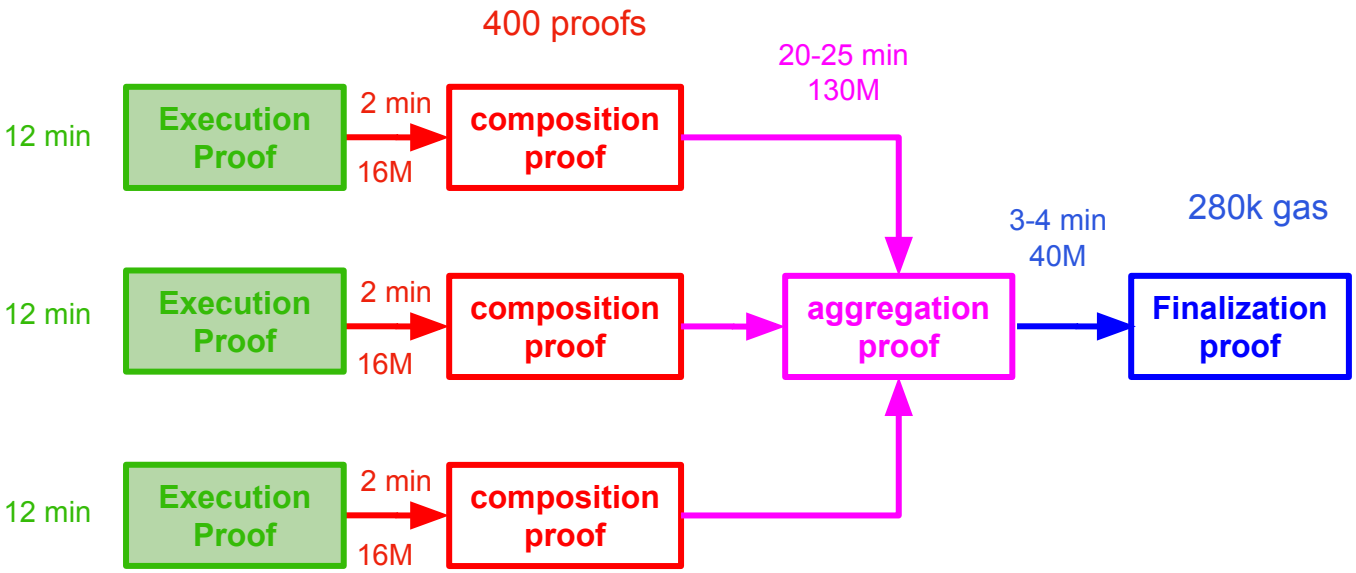


BLS12-377

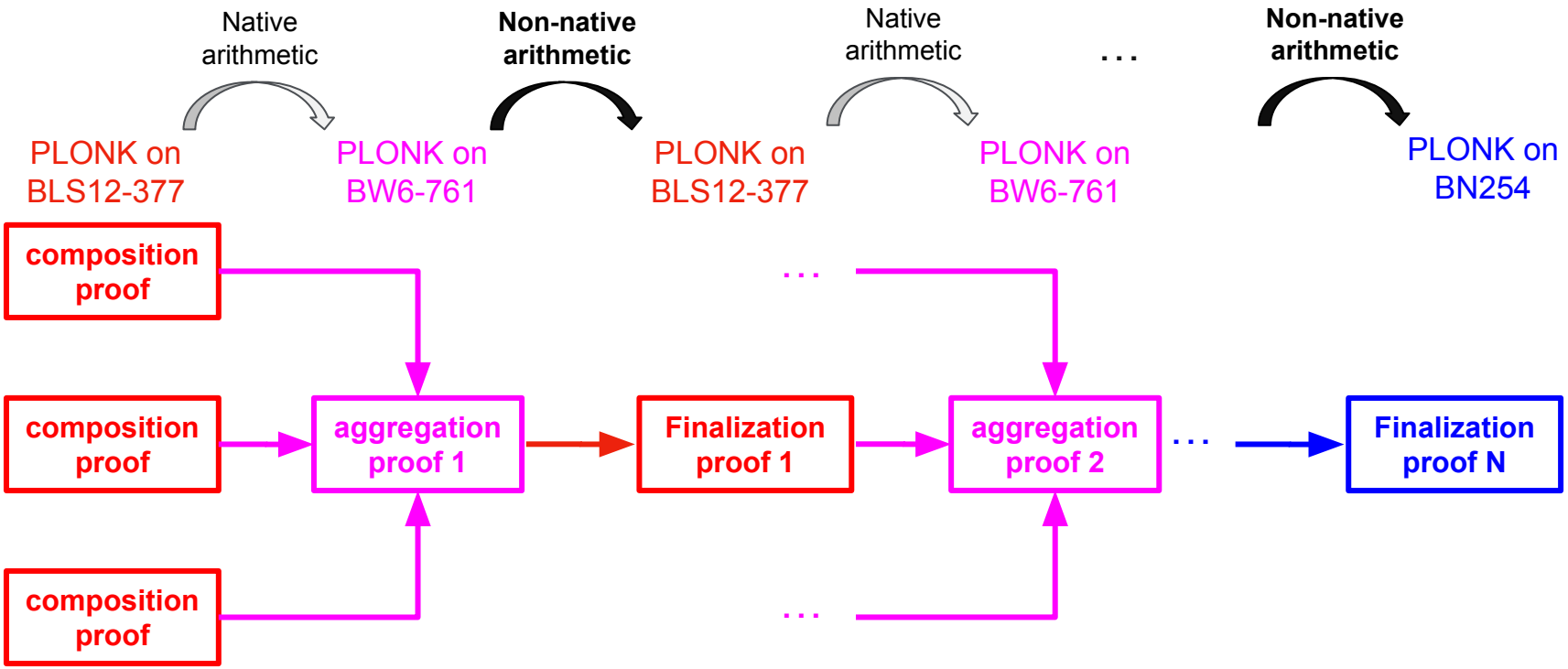
2-chain or *non-native* 2-cycle?



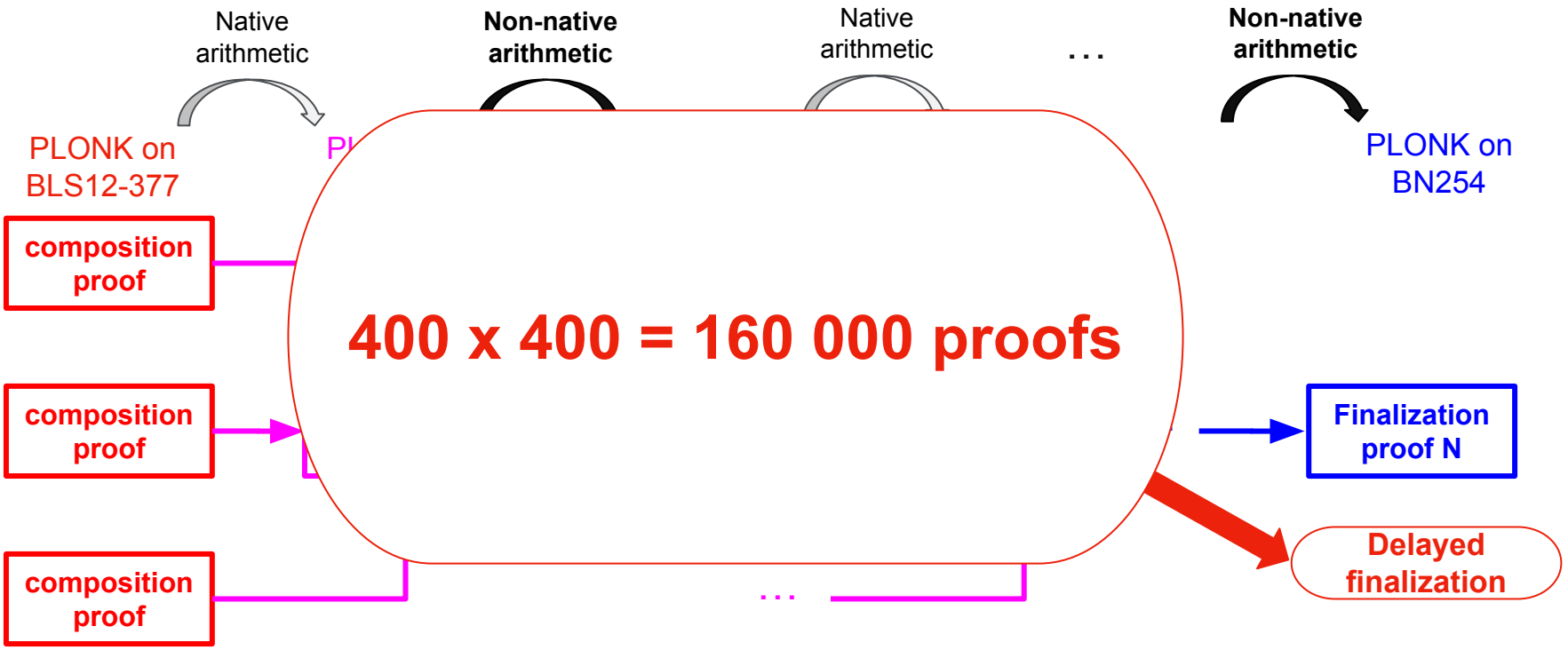
Benchmarks (hp6a)



Multi-layer 2-chain aggregation?

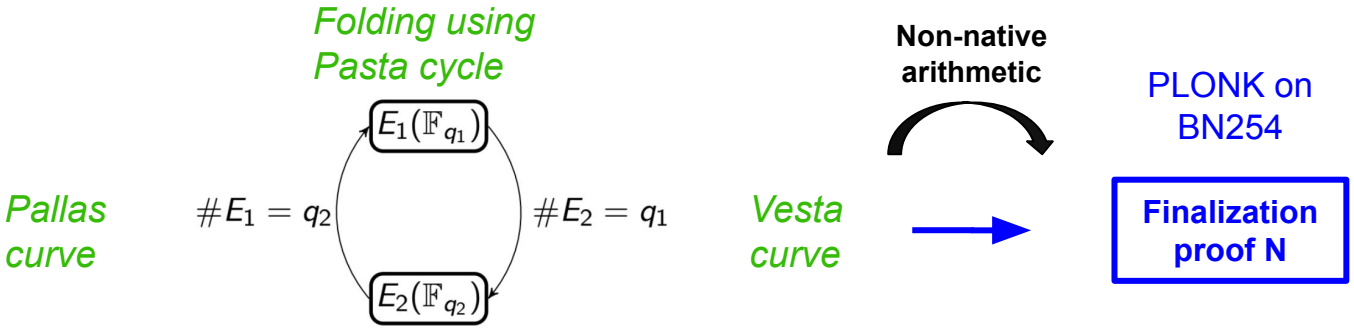


Multi-layer 2-chain aggregation?



zkFolding instead of zkAggregation ?

Folding schemes (e.g. Nova https://eprint.iacr.org/2021/370) work over a cycle of non-pairing-friendly curves.



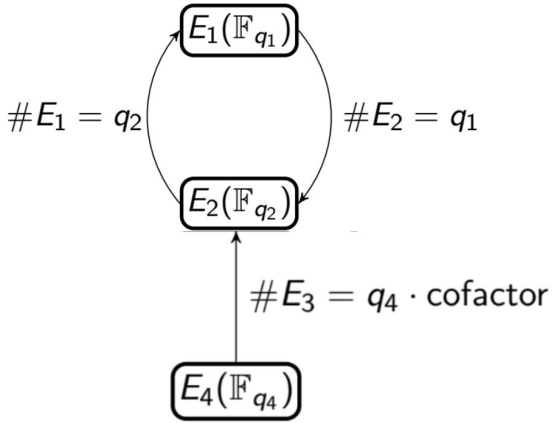
Question: is it possible to find a non-pairing-friendly cycle on top of BN254, BLS12-381 or any SNARK-pairing-friendly curve?

zkFolding instead of zkAggregation ?

Families of prime-order endomorphism-equipped embedded curves on pairing-friendly curves

Antonio Sanso[1] and Youssef El Housni[2]

[1] Ethereum Foundation
[2] Linea




```
sage: hex(r0)
'0x40000000e18820ac7e4ae010935bb29483628260db62ef544865b1c000000001'
sage: hex(p0)
'0xaaaaaaaae30cb2d5ddbe0944aad1b96788db962bb21454db5c12fca0d6c205a3271689e66595fc8a55ac51118872aaab'
sage: hex(r1)
'0x40000000e18820ac7e4ae010935bb2938362825f1852adfcd931154000000003'
sage: E0 = EllipticCurve(GF(p0), [0,-3])
sage: E1 = EllipticCurve(GF(r0), [0,11])
sage: E2 = EllipticCurve(GF(r1), [0,22])
sage: E0.order() % r0 == 0
True
sage: E1.order() == r1
True
sage: E2.order() == r0
True
```

Questions?

gnark@consensys.net
youssef.elhousni@consensys.net

X: @gnark_team, @YoussefElHousn3
GH: @yelhousni

linea.build
play.gnark.io
github.com/consensys/gnark
github.com/consensys/gnark-crypto