STARKWARE

Scaling Computations on Blockchains with ZK-STARKs

Eli Ben-Sasson | co-founder & President | StarkWare | @elibensasson

Tl;dr

- Blockchains Rock! But lack Scale and Privacy

- ZK-STARKs solve both problems!
  a. Theoretically
  b. and practically,
  c. and accessibly, through Cairo

- So, come learn Cairo! 

Tl;dr

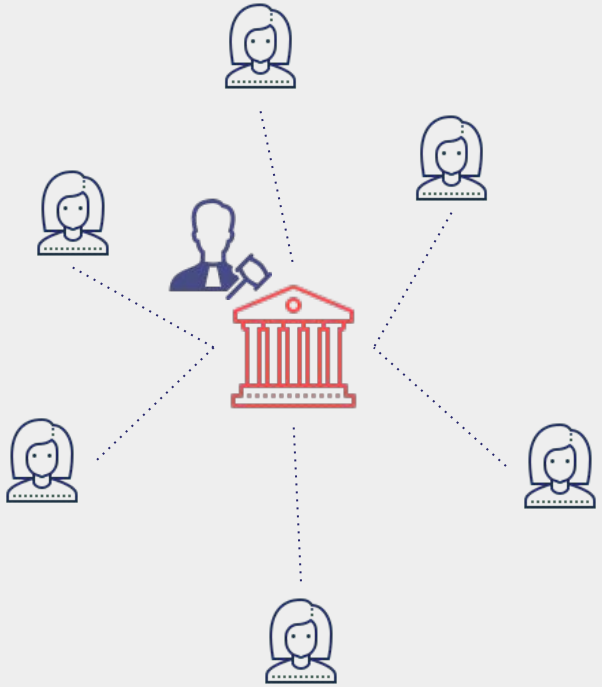
- **Blockchains Rock! But lack Scale and Privacy**

- ZK-STARKs solve both problems!
  a. Theoretically
  b. and practically,
  c. and accessibly, through Cairo

- So, come learn Cairo! 

Trusted Party
=
Delegated Accountability

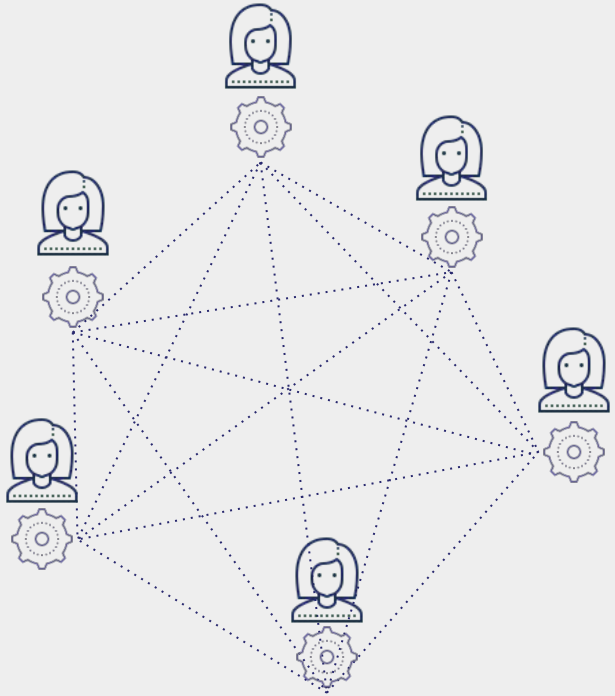
Trust central party/auditor



Blockchains
=
Inclusive Accountability

Verify, Don’t Trust

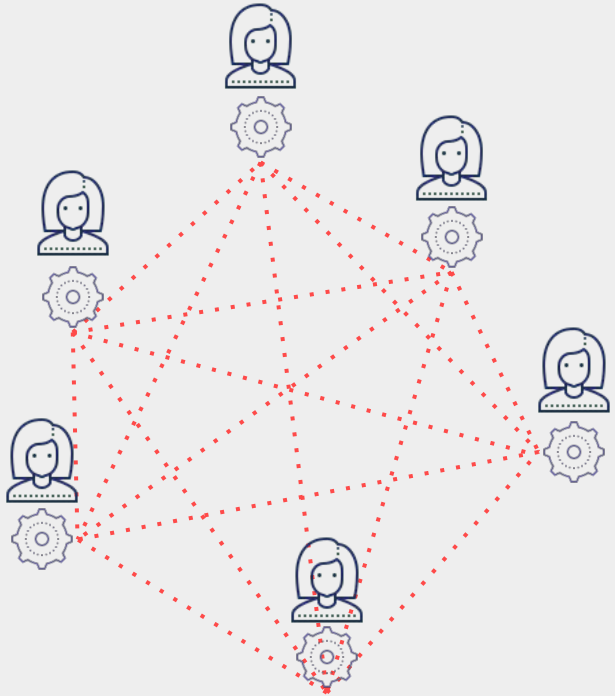
Verify (all transactions), don’t trust



Blockchains
=
Inclusive Accountability

Sacrifice Privacy & Scalability

Verify (all transactions), don't trust



STARKWARE

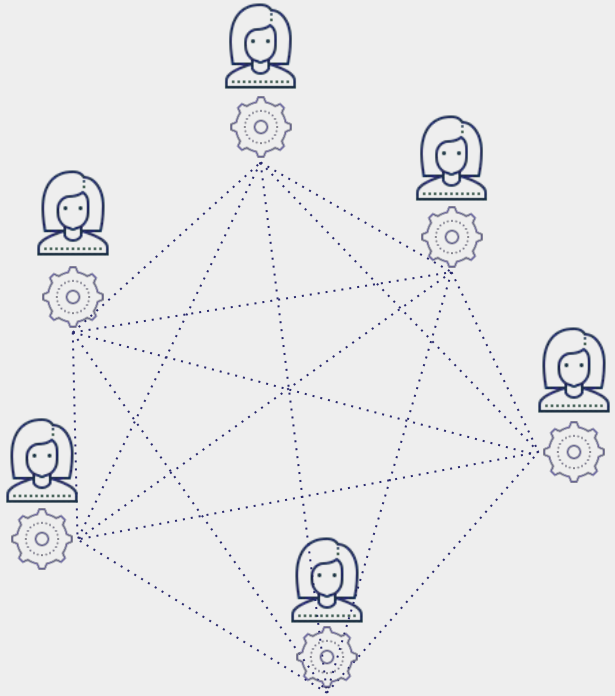
@starkwareltd | @elibensasson

Blockchains
=
Inclusive Accountability

Sacrifice Privacy & Scalability

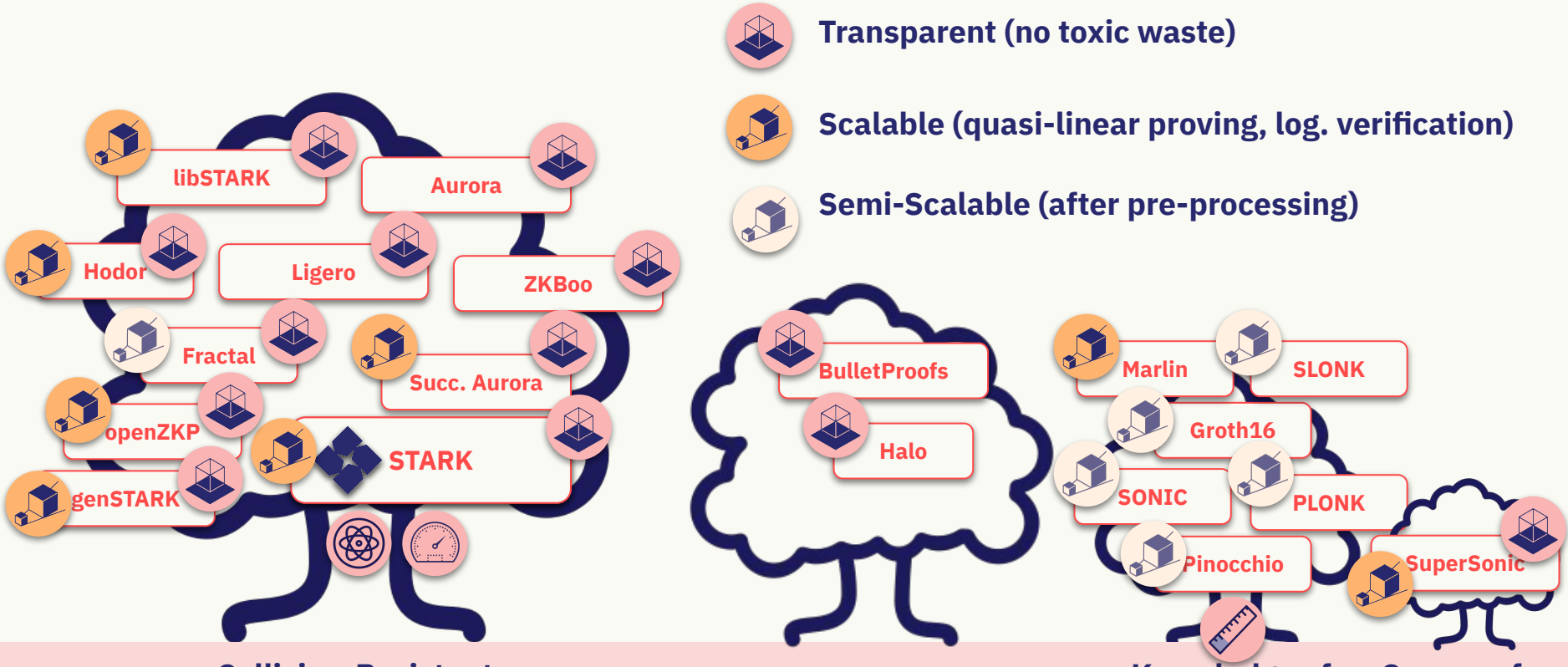
Scalable ZKPs solve both problems

Verify (all transactions), don't trust



@starkwareltd | @elibensasson

A Cambrian Explosion of ZKPs (read my post/watch video)

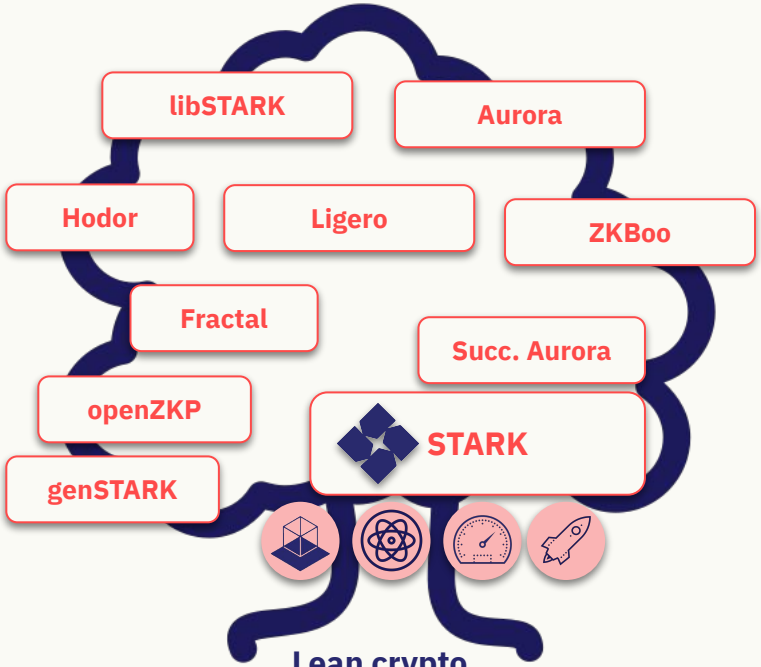


Summary (read my post/watch video)

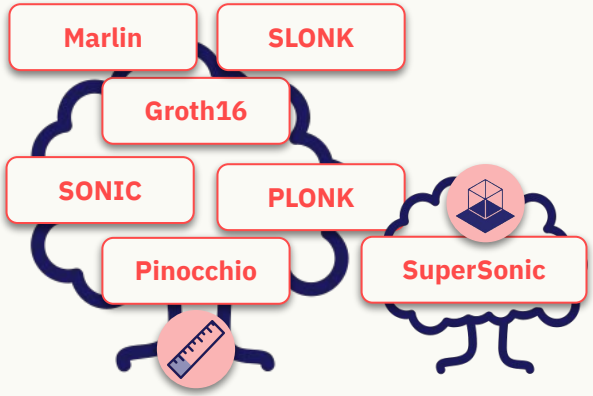
ZKP Cambrian explosion ongoing, expect more science!

ZKP members differ by (**i**) arithmetization, (**ii**) low-degreeness, and (**iii**) crypto assumptions

For short proofs, use **Groth16 SNARKs**.
For everything else, there's **STARKs**!



Lean crypto
Post quantum security
Fastest proving time
Future proofing (Lindsey)



Proof length

STARKWARE

STARKWARE

ZK-STARKs solve (1) scale (2) privacy

Theoretically

STARKWARE

ZK-STARKs solve (1) scale (2) privacy

Theoretically and Practically

STARKWARE

ZK-STARKs solve (1) scale (2) privacy

Theoretically and Practically and Accessibly

@starkwareltd | @elibensasson

STARKWARE

ZK-STARKs solve (1) scale (2) privacy

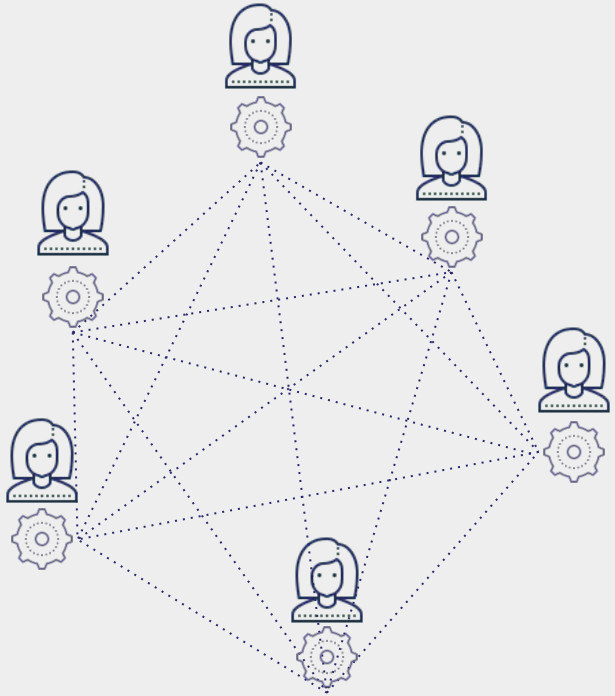
Theoretically

Blockchains
=
Inclusive Accountability

Sacrifice Privacy & Scalability

ZK-STARKs solve both problems

Verify (all transactions), don't trust



@starkwareltd | @elibensasson

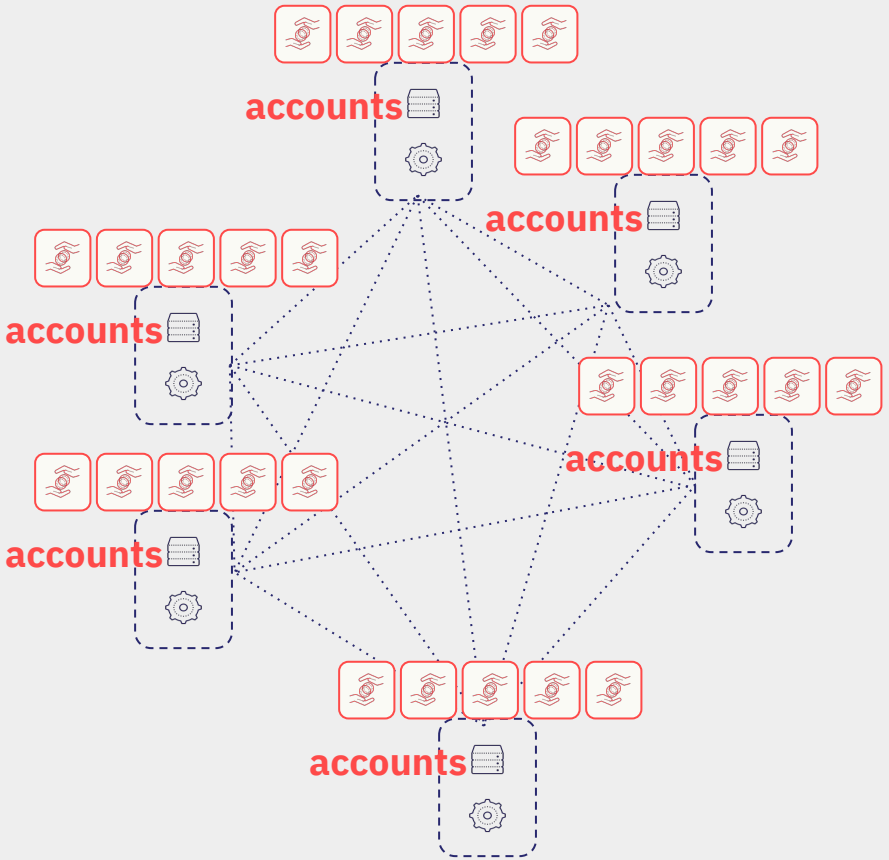
Blockchains
=
Inclusive Accountability

Sacrifice Privacy & Scalability

ZK-STARKs solve both problems

STARKWARE

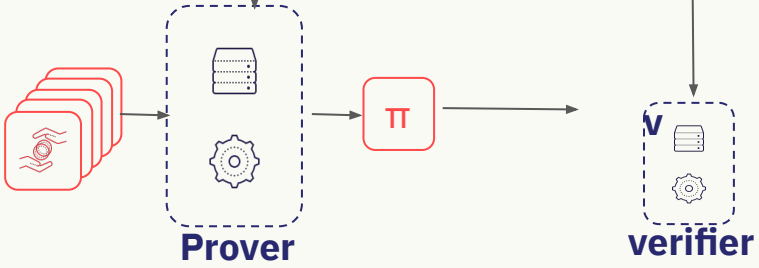
Verify (all transactions), don't trust



ZK-STARK

Privacy (Zero Knowledge, ZK)
Prover's private inputs are shielded

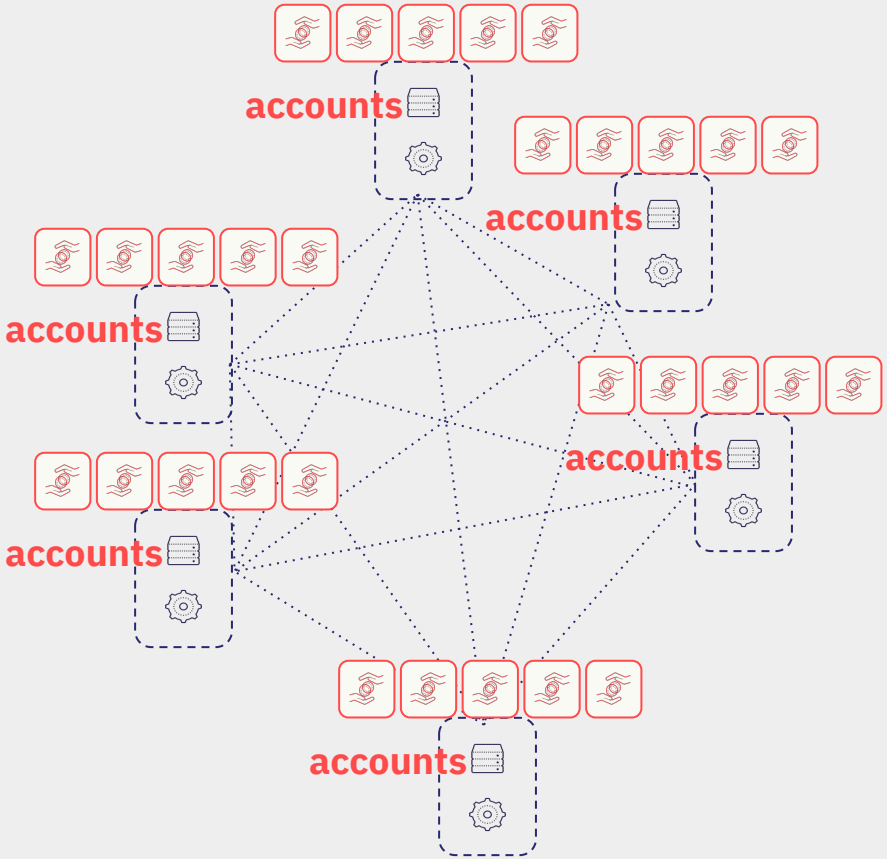
Scalability
Exponentially small verifier running time*
Nearly linear prover running time*



*With respect to size of computation

STARKWARE

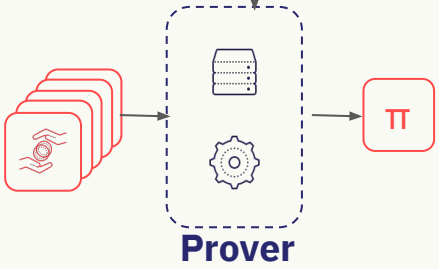
Verify (all transactions), don't trust



ZK-STARK

Privacy (Zero Knowledge, ZK)
Prover's private inputs are shielded

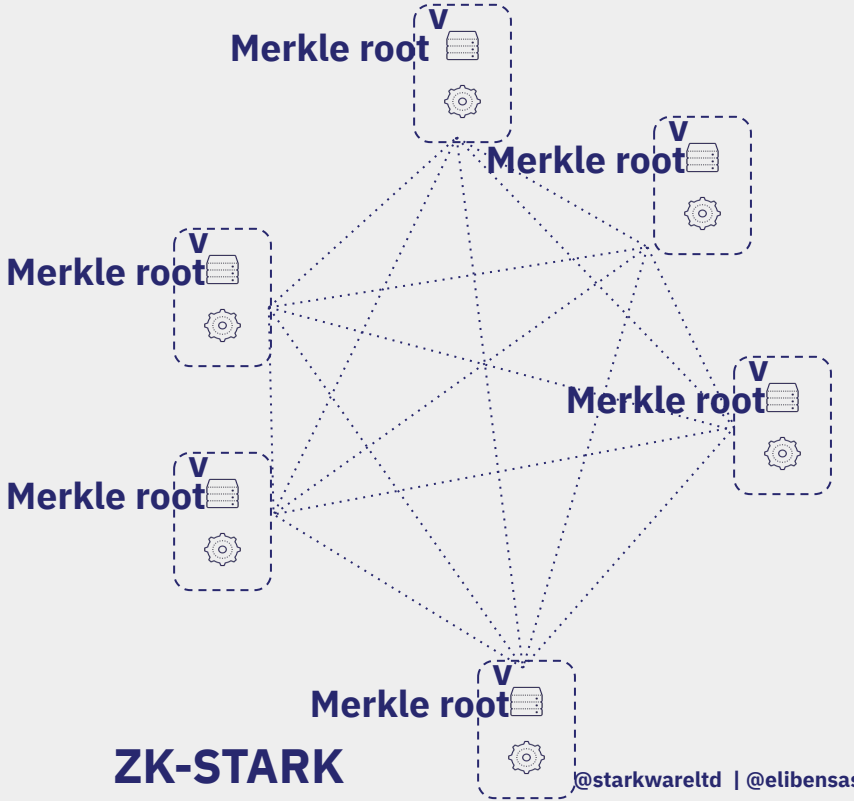
Scalability
Exponentially small verifier running time*
Nearly linear prover running time*



*With respect to size of computation

STARKWARE

Verify STARK batch, don't trust

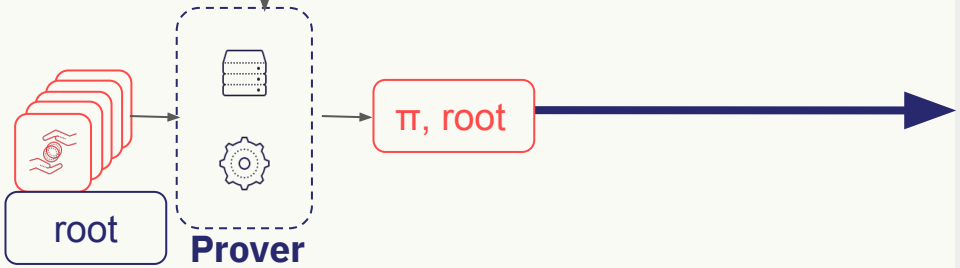


@starkwareltd | @elibensasson

ZK-STARK

Privacy (Zero Knowledge, ZK)
Prover's private inputs are shielded

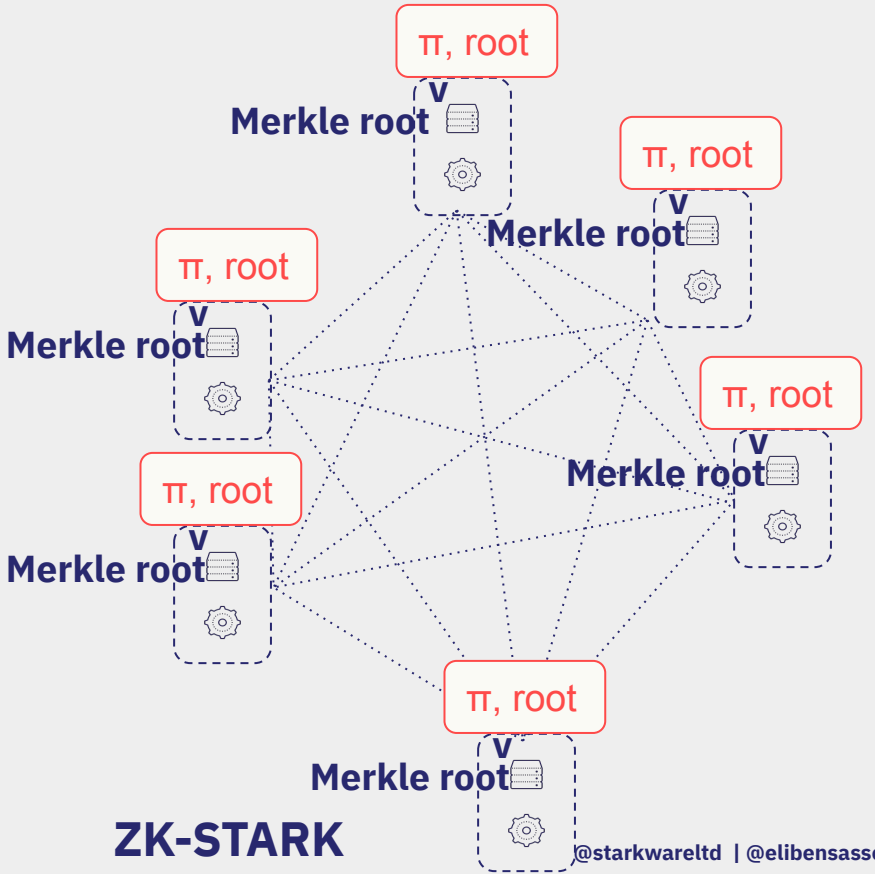
Scalability
Exponentially small verifier running time*
Nearly linear prover running time*



*With respect to size of computation

STARKWARE

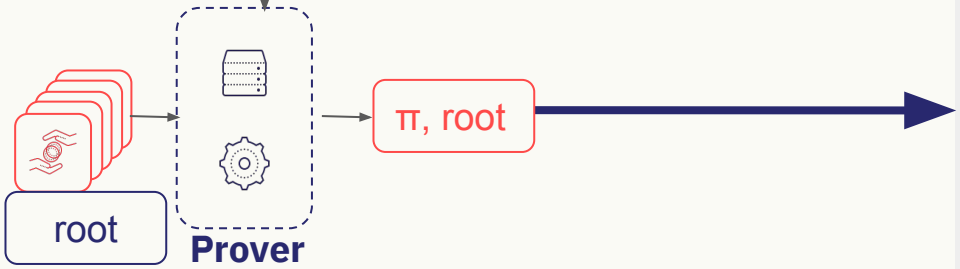
Verify STARK batch, don't trust



ZK-STARK

Privacy (Zero Knowledge, ZK)
Prover's private inputs are shielded

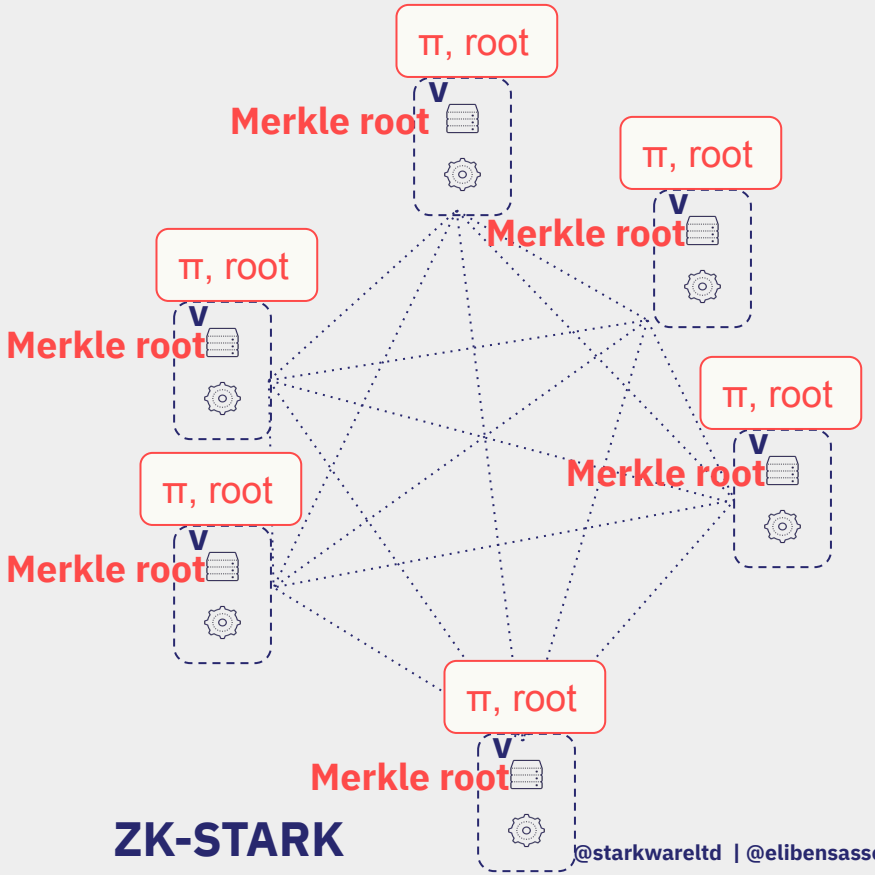
Scalability
Exponentially small verifier running time*
Nearly linear prover running time*



*With respect to size of computation

STARKWARE

Verify STARK batch, don't trust



ZK-STARK

Privacy (Zero Knowledge, ZK)
Prover's private inputs are shielded

Scalability
Exponentially small verifier running time*
Nearly linear prover running time*

Universality
Applicability to general computation

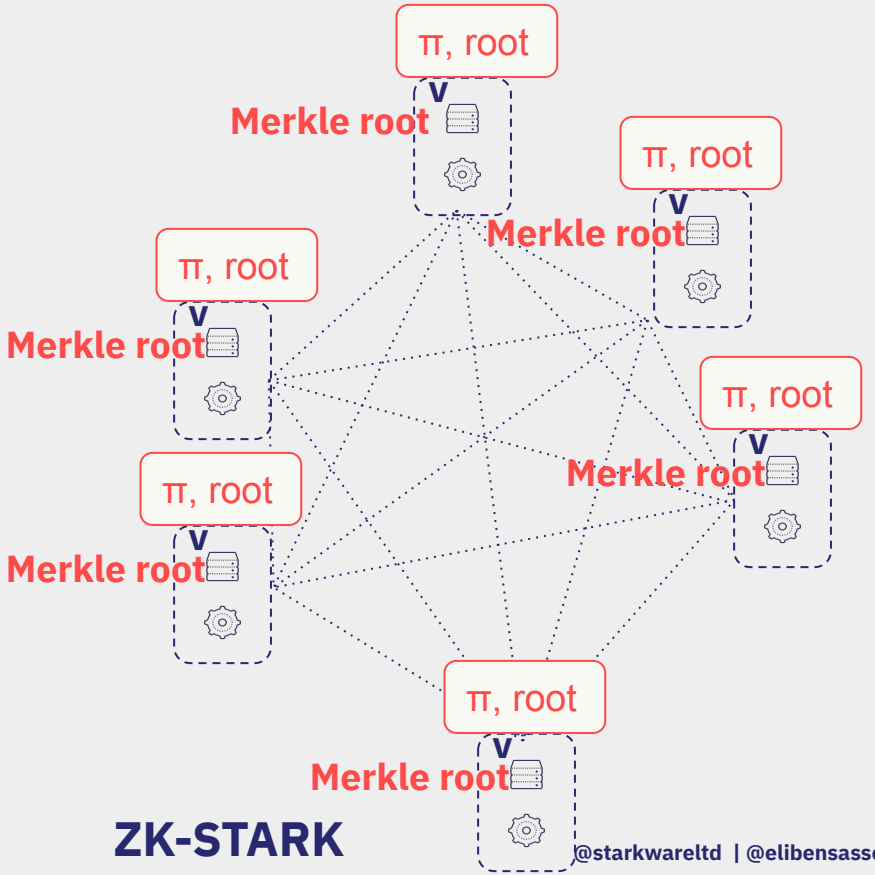
Transparency
No toxic waste (i.e. no trusted setup)

Lean & Battle-Hardened Cryptography
e.g. post-quantum secure

*With respect to size of computation

STARKWARE

Verify STARK batch, don't trust



ZK-STARK

@starkwareltd | @elibensasson

ZK-STARK

Privacy (Zero Knowledge, ZK)
Prover's private inputs are shielded

Scalability
Exponentially small verifier running time*
Nearly linear prover running time*

Universality
Applicability to general computation

Transparency
No toxic waste (i.e. no trusted setup)

Lean & Battle-Hardened Cryptography
e.g. post-quantum secure

*With respect to size of computation

STARKWARE

ZK-SNARK

Privacy (Zero Knowledge, ZK)
Prover's private inputs are shielded

Succinctness
Exponentially small verifier running time **only post processing**
Arbitrary prover running time

Universality
Applicability to general computation

Non-Interactive
Setup may be (i) > computation time, (ii) trusted (toxic waste)

*With respect to size of computation

@starkwareltd | @elibensasson

ZK-STARK > ZK-SNARK

Privacy (Zero Knowledge, ZK)
Prover's private inputs are shielded

Scalability
Exponentially small verifier running time*
Nearly linear prover running time*

Universality
Applicability to general computation

Transparency
No toxic waste (i.e. no trusted setup)

Lean & Battle-Hardened Cryptography
e.g. post-quantum secure

*With respect to size of computation

Fiat-Shamir/Micali/BCS16

Privacy (Zero Knowledge, ZK)
Prover's private inputs are shielded

Succinctness
Exponentially small verifier running time **only post processing**
Arbitrary prover running time

Universality
Applicability to general computation

Non-Interactive
Setup may be (i) > computation time, (ii) trusted (toxic waste)

*With respect to size of computation

STARKWARE

@starkwareltd | @elibensasson

STARKWARE

ZK-STARKs solve (1) scale (2) privacy

Theoretically and Practically

Transactions per second (TPS)

Naive Replay	StarkEx (DeversiFi)
	
4	**9,000**
trading	trading
	
30	**18,000**
payments	payments

Reddit Bake-off 2020

One

STARK Proof

On-chain data
Ethereum mainnet

300K
Transactions

315
Gas / Transaction

3,000
TPS

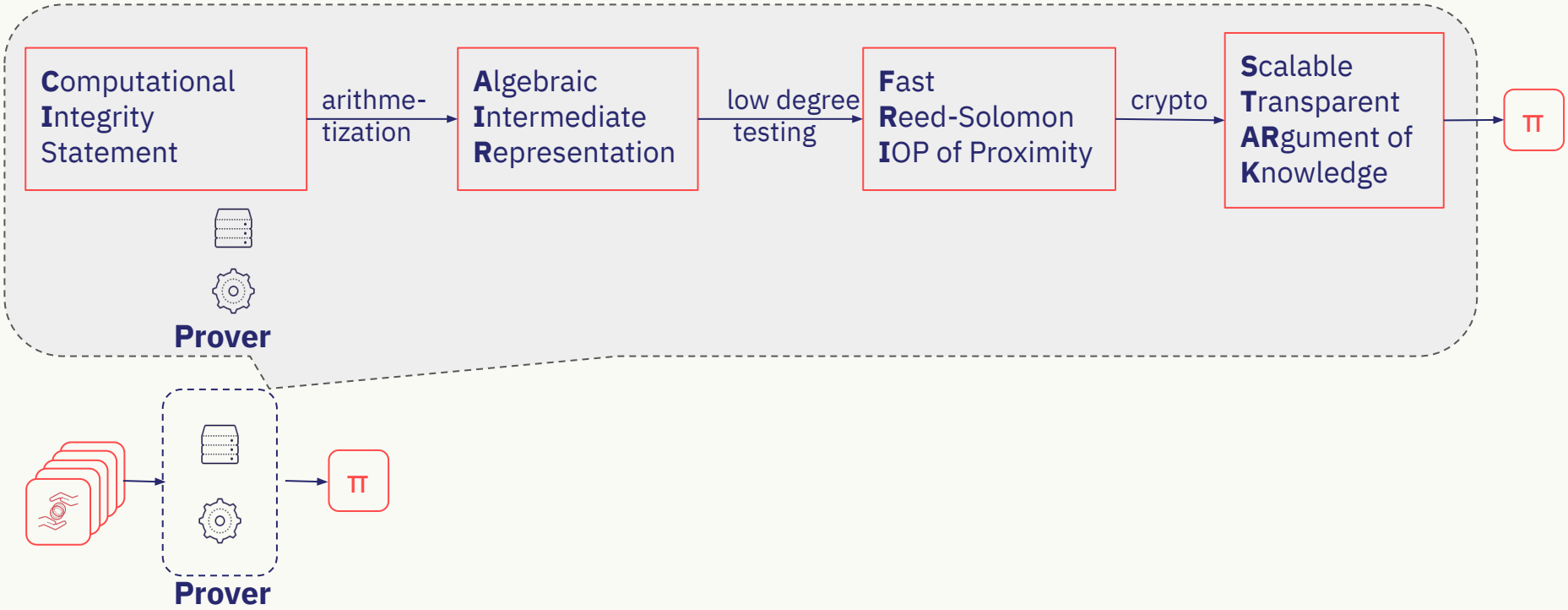
8
Blocks

No Trusted Setup

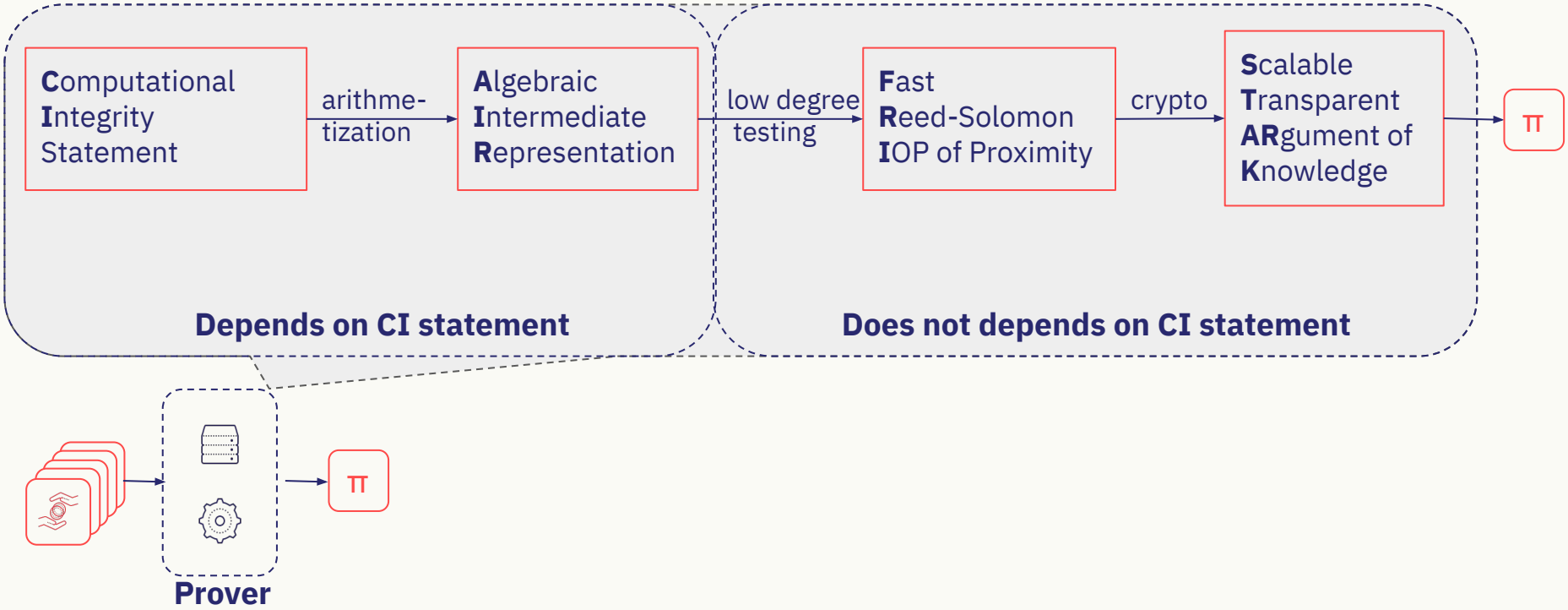
7 Wallets Supported

STARKWARE

How to build an AIR-FRI STARK



How to build an AIR-FRI STARK



How to build an AIR-FRI STARK

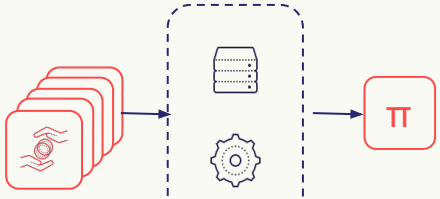
Computational
Integrity
Statement

arithme-
tization

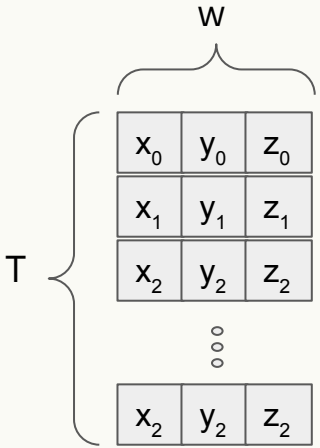
Algebraic
Intermediate
Representation

π

Depends on CI statement



Transition function: constraints on trace



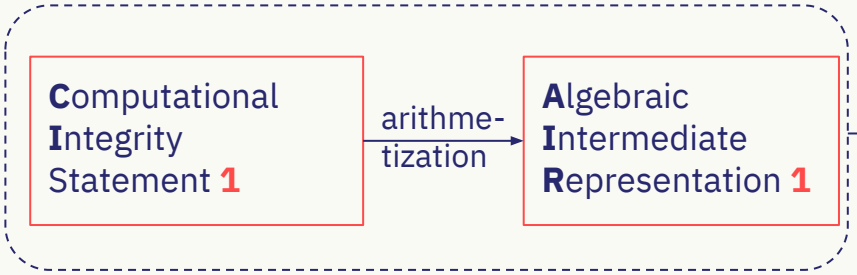
Constraints

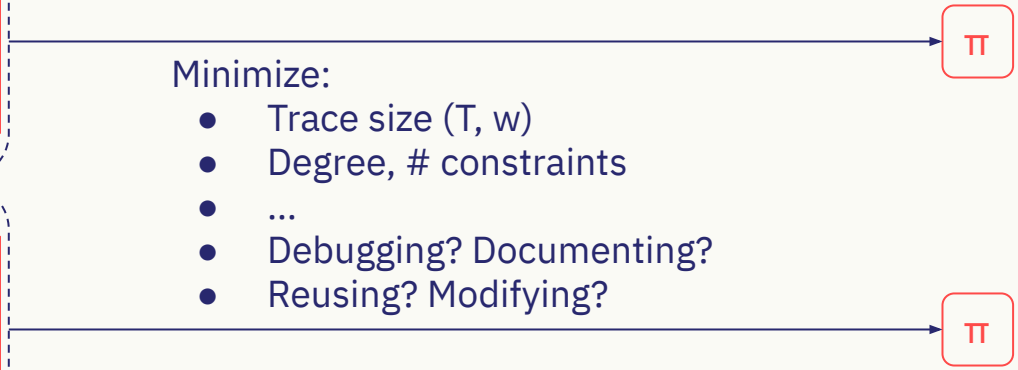
- $X_i^2 - Y_{i+2} = 0$ for i = 0,2,4,...
- $X_i Y_{i+1} = 1$ for i = 1,9,17,...
- ...

STARKWARE

AIR Visualizer

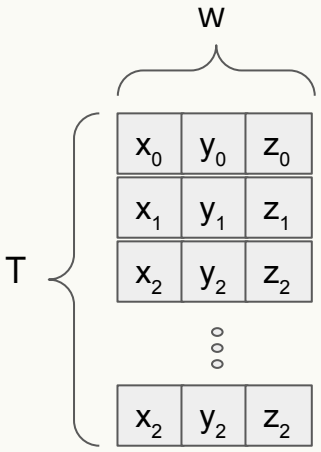
ASIC-like STARK





Minimize:

- Trace size (T, w)
- Degree, # constraints
- ...
- Debugging? Documenting?
- Reusing? Modifying?



Constraints

- $X_i^2 - Y_{i+2} = 0$ for i = 0,2,4,...
- $X_i Y_{i+1} = 1$ for i = 1,9,17,...
- ...

CPU AIR - CAIRo

Computational **I**ntegrity Statement **1**

Computational **I**ntegrity Statement **2**

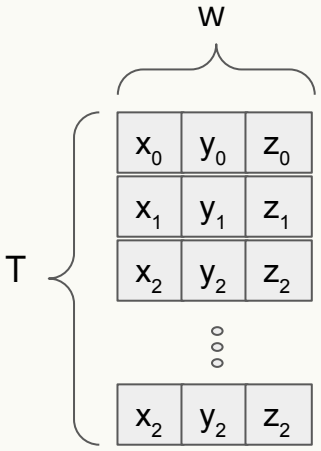
High Level Programming Lang

High Level Programming Lang

Compiler

One AIR to Rule them All

π



One AIR to Rule Them All

w < 50
# constraints < 100
Degree = 2
Variable T (depends on prog)

Cairo Theory

Cairo is 1st

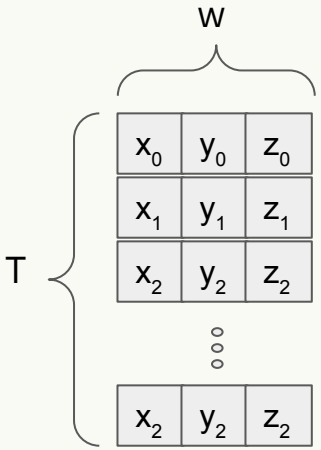
- **Universal** Von Neumann **ST**ARK

Scalability
Exponentially small verifier running time*
Nearly linear prover running time*

Transparency
No toxic waste (i.e. no trusted setup)

Universality
Applicability to general computation

- Universal Von Neumann verifier on blockchain (Ethereum Mainnet)



One AIR to Rule Them All

w < 50
# constraints < 100
Degree = 2
Variable T (depends on prog)

Cairo Theory

Cairo is 1st

- **Universal** Von Neumann **ST**ARK

Scalability
Exponentially small verifier running time*
Nearly linear prover running time*

Transparency
No toxic waste (i.e. no trusted setup)

Universality
Applicability to general computation

- Universal Von Neumann verifier on blockchain (Ethereum Mainnet)

Prior Works on Universal ZKPs

SNARK: Pinocchio, TinyRAM, vnTinyRAM, Buffet, Pequin, jSNARK, ZEXE,...

STARK: TinyRAM, DiStaff, AirScript, ...

Bulletproofs: zkVM, Spacesuit, ...

See zkp.science for partial list

STARKWARE

ZK-STARKs solve (1) scale (2) privacy

Theoretically and Practically and Accessibly, through Cairo

@starkwareltd | @elibensasson

Hello Cairo!

- Cairo* as MVL - Minimal Viable Language for production STARK systems
  - **Goldilocks principle:** architecture is “just right” balance of expressibility and STARK prover efficiency (minimal trace size)
  - **Neither too hot:** only 3 registers (PC, allocation pointer, frame pointer), minimal instruction set, ...
  - **Nor too cold:** supports functions, recursion, branching, conditionals, random memory, ...

Snippets

```
func fib(x, y, n) -> (res):
    if n == 0:
        return (y)
    end

    let (res) = fib(y, x + y, n - 1)
    return (res)
end
```

Library functions

```
let (pedersen_ptr, result) = pedersen_hash(
    pedersen_ptr, x, y)
```

```
from starkware.cairo.common.math import assert_le
let (range_check_ptr) = assert_le(range_check_ptr, lower, value)
```

Cairo - Production Grade STARKs for Blockchain

DeversiFi

SPOT TRADING

iMMUTABLE

NFT TRADING

δY / δX

PERPETUAL TRADING

Cairo - Production Grade STARKs for Blockchain

Compiler

Translate Cairo code into bytecode

Virtual Machine

Runs the Cairo program and turns it into a trace

IED Extensions

Syntax highlighting for VIM & Visual Studio Code

Tracer

Visual rendering of the program run

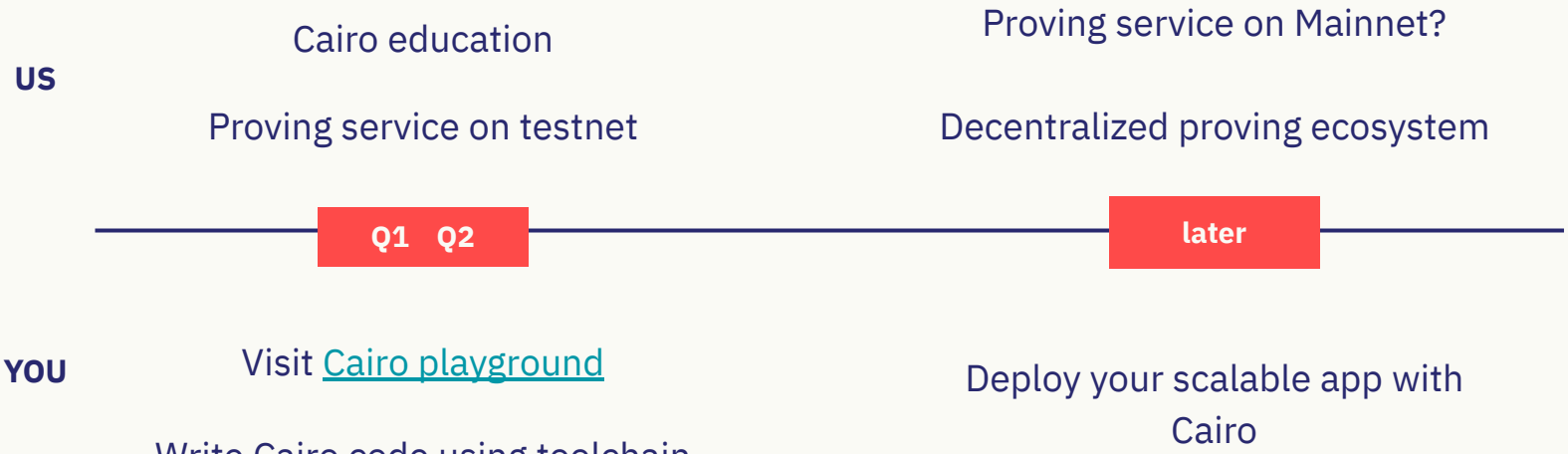
Application Code

StarkEx 2.0 code same code that powers DeversiFi on mainnet

Cairo - Production Grade STARKs for Blockchain

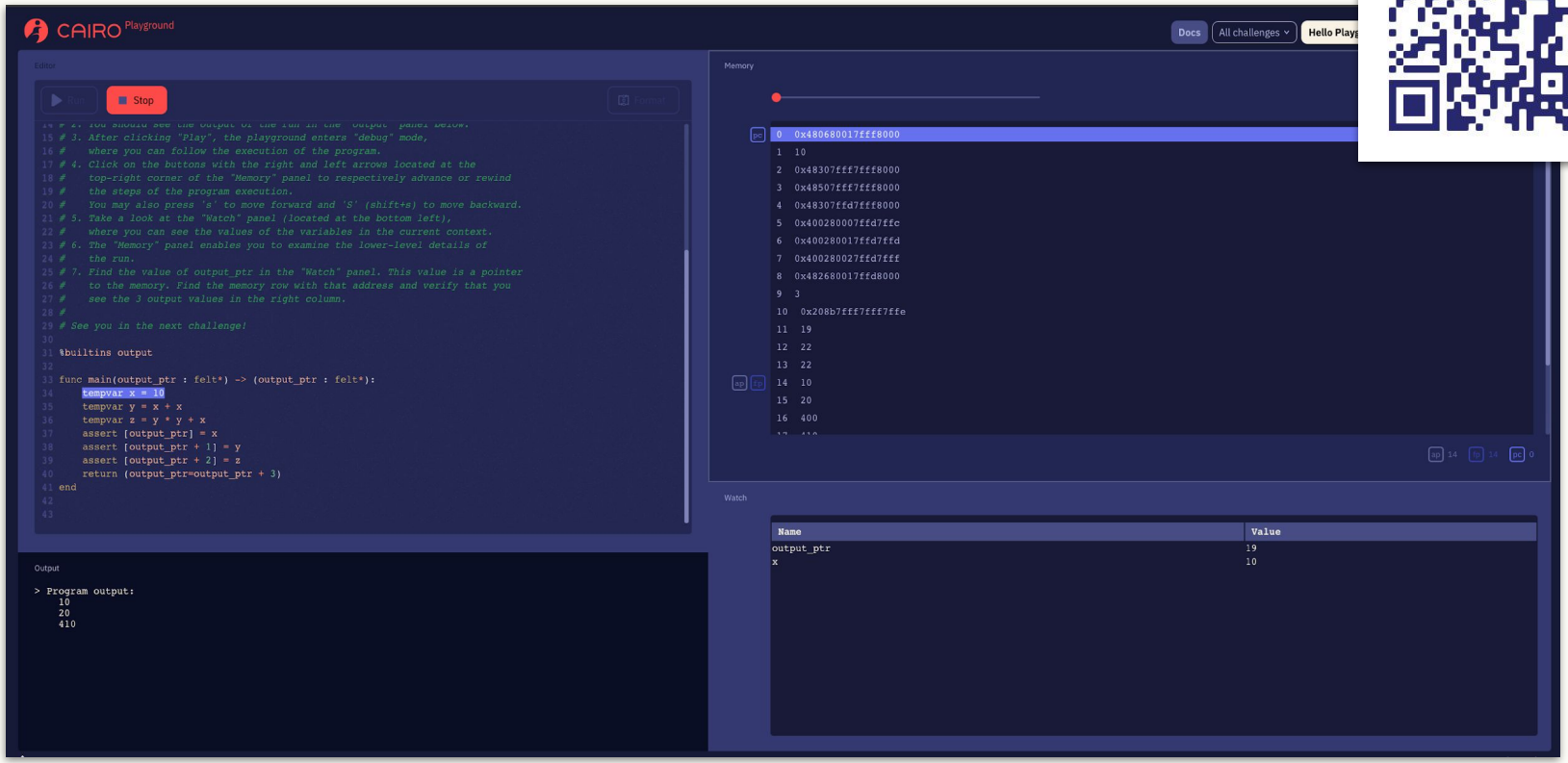
- Efficiency
  - Theoretical estimate: Cairo 20-30% more expensive in proving time than ASIC-AIR
  - Real World experience: Cairo trace size << ASIC-AIR trace size
  - Why? High level language allows devs to code complex optimizations safely

Cairo - Next Steps



Cairo - Come & Play

try it now!



Further resources

STARK Theory

- STARK paper
- FRI paper
- ethSTARK documentation
- Latest FRI soundness

STARK math education

- Blog posts
- Crowdcast STARK @ home series

Cairo resources

- Documentation
- Blogs [1, 2]
- Main page
- Discord server

STARK code

- ethSTARK (open source)
- Ziggy (pq-secure signature, open source)
- Cairo StarkEx code