


The story of Cairo

Henri Lieutaud

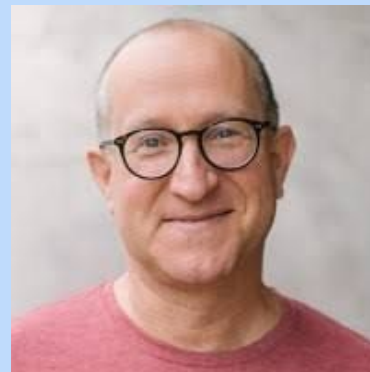
 <https://github.com/starknet-edu> | @henrilienri

 September 2023



What is provable code anyway?

STARK proofs



You know what could be cool?

If I had a 1 million degree polynomial; and I could prove to you that I know 1 million values of X where it goes to 0; but you wouldn't have to calculate the polynomial on each of the 1 million points, so you like, save time

STARK proofs

You know what could be cool?

If I had a 1 million degree polynomial; and I could prove to you that I know 1 million values of X where it goes to 0; but you wouldn't have to calculate the polynomial on each of the 1 million points, so you like, save time



STARK proofs

STARK is a family of cryptographic proof systems that can be used for privacy and scalability

STARK proves statements, saying that a computation was executed correctly

Examples for provable statements:

- The 1000th number in the Fibonacci sequence is X.
- I have 100 signed bank transactions that are valid.

Verification time is exponentially smaller than naive computation.



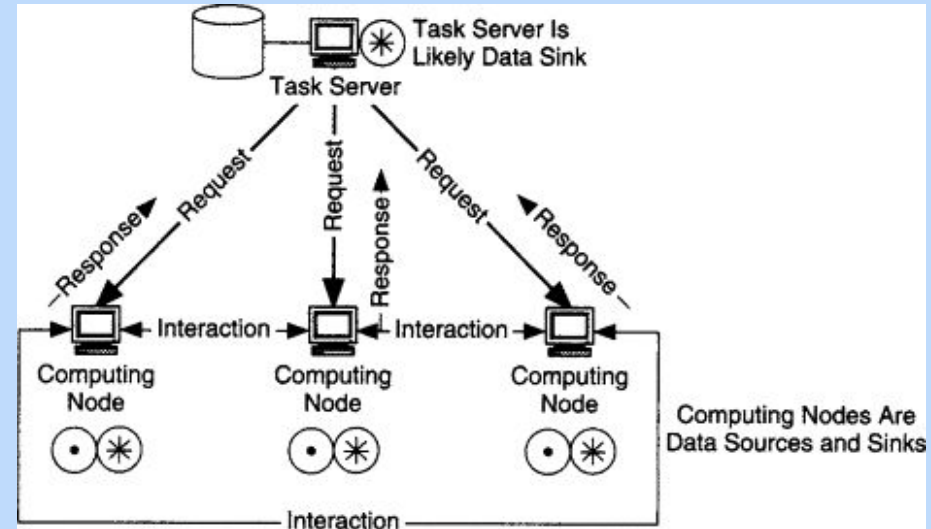
Why this matters

The internet -> Cloud computing -> Distributed computing

Distributed computing is a major step in our collective capacity to process information

This process currently relies on trust, relations and intermediaries

Computational proof do to code execution what blockchains to do value transfers



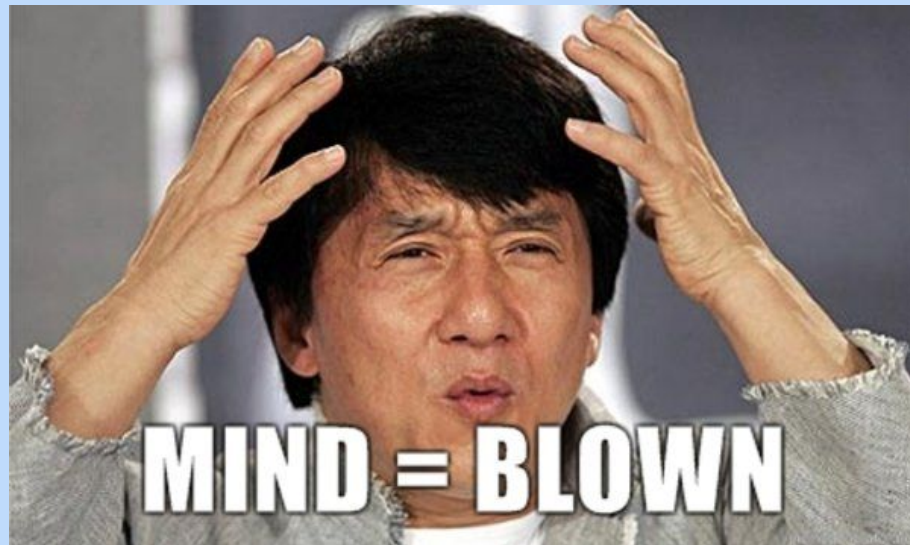
Why this matters

This is much, much bigger than the blockchain industry

ZK rollups are one application of provable code

Industries get started where there is a lack of offering for niche users

The only ecosystem paranoid enough to require computational proofs is the blockchain space (and for good reason)



How was Cairo created?

How Cairo was born

Relying on Stark proofs for provable computation means that your code is expressed in the form of polynomials

Finding the right polynomials for your program is a time consuming, and requires specific skills

How Cairo was born

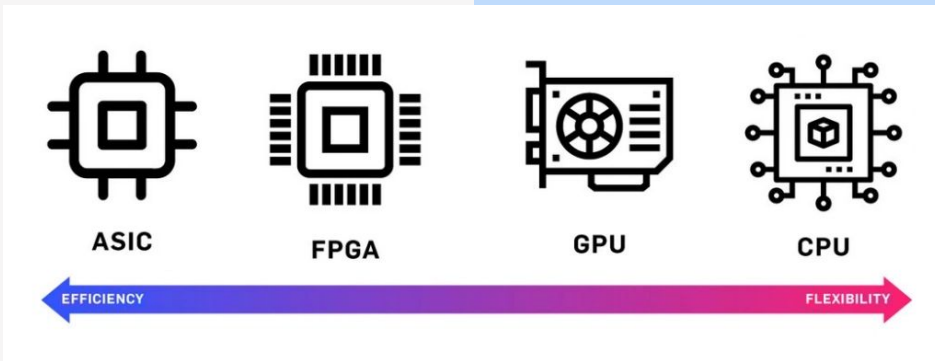
We can make a parallel to designing computer chips

ASIC are specialized, fast, expensive to make and very hard to design.

CPUs are general, slower, cheaper and easier to program for. A single architecture for multiple purposes.

Efficiency vs Flexibility

c0	c1	c2	c3	c4	c5
IA	Step0_A	Step1_A	Step2_A	Step3_A	Step4
Step9_A	Step0_A	Step1_A	Step2_A	Step3_A	Step4
Step9_A	Step0_A	Step1_A	Step2_A	Step3_A	Step4
Step9_A	Step0_A	Step1_A	Step2_A	Step3_A	Step4
Step9_A	Step0_A	Step1_A	Step2_A	Step3_A	Step4
Step9_A	Step0_A	Step1_A	Step2_A	Step3_A	Step4
Step9_A	Step0_A	Step1_A	Step2_A	Step3_A	Step4
Step9_A	Step0_A	Step1_A	Step2_A	Step3_A	Step4
Step9_A	Step0_A	Step1_A	Step2_A	Step3_A	Step4
Step9_A	Step0_A	Step1_A	Step2_A	Step3_A	Step4



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

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

Step 1: Building STARKs manually (ASIC)

- Designing the trace:
 - How many columns?
 - Assign a meaning to trace cells (i.e. X_{4i} represent the transfer amount of transaction i)
- Adding constraints:
 - Only polynomial constraints of low degree.
- Manual optimizations.
 - How to measure efficiency?
 - Trace space (number of cells) - Prover time and memory
 - Constraint size - Verifier time
- Not ideal for complex statements
 - This gets very complex very fast

Step 3: DSLs for AIRs (GPU)

- Use a higher level language to generate trace and constraints
Why? More similar to programming. Flexible.
- Drawbacks compared to real programming:
 - No memory
 - Branching is expensive - both branches use trace space
 - No recursion / general loops. Only constant size loops
 - Not Turing complete
 - All iterations take trace space
 - STARKs can handle a variable number of instances by repeating the basic block
 - Not all programs are repetitive in nature
 - For example, executing transactions and then compressing all the outputs

Step 4: Cairo - CPU Air

- One AIR, One verifier.
- StarkWare's CPU AIR implementation. Cairo stands for CpuAIR (O).
- Universal Machine. Turing complete. Von Neumann
- Pay (with trace cells) only for what actually ran
- Complex non repetitive logic (conditionals / recursion)
- Efficient

In a real world application, the cairo version with exact same logic was about 20-30% more expensive compared to the handwritten AIR version.

With logical optimizations enabled by Cairo, was actually cheaper!

Step 5 - Cairo language

High level language (easier to learn)

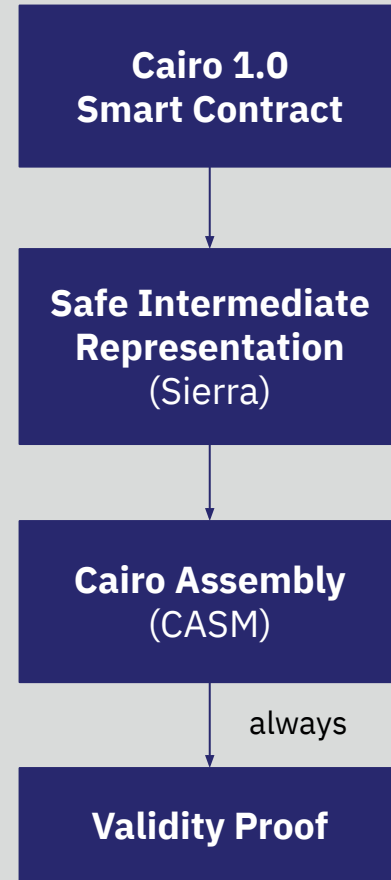
Syntax inspired by Rust (strongly typed)

Compiles to Sierra

Safe Intermediate Representation

Even if a program fail a proof is generated

Sequencers are ALWAYS compensated



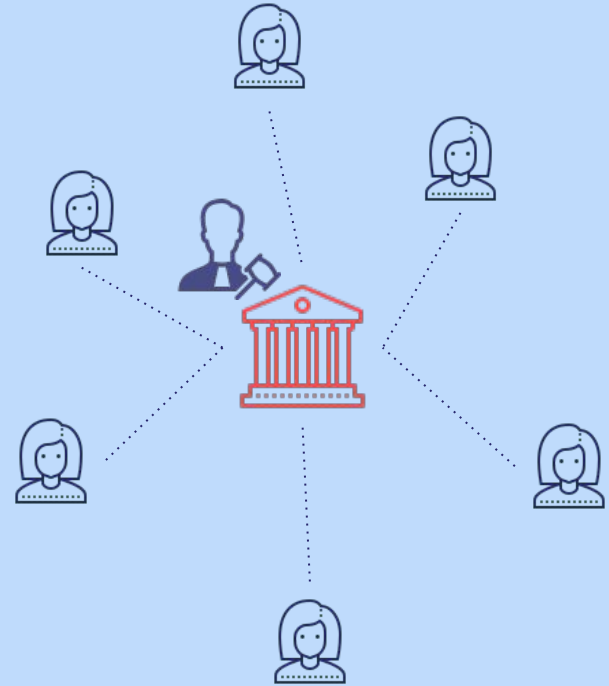
How do we use it

The blockchain paradox

Bitcoin was launched in 2008, Ethereum in 2015.

Why so slow, compared to banks, credit cards, Alipay?

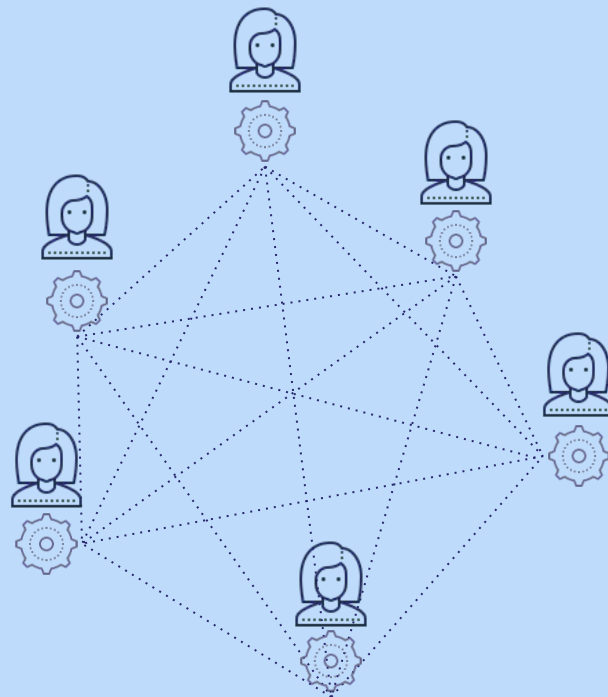
**Trusted Party
(e.g., Banks)
=
Delegated
Accountability**



Trust central party/auditor

Blockchains = Inclusive Accountability

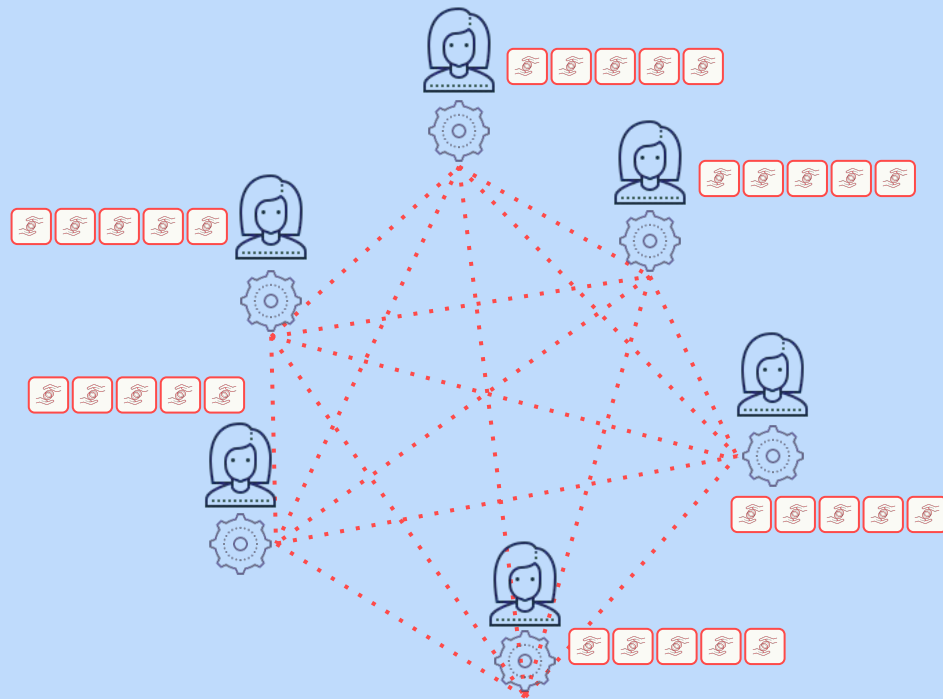
Verify, Don't Trust



Verify (all transactions), don't trust

Blockchains = Inclusive Accountability

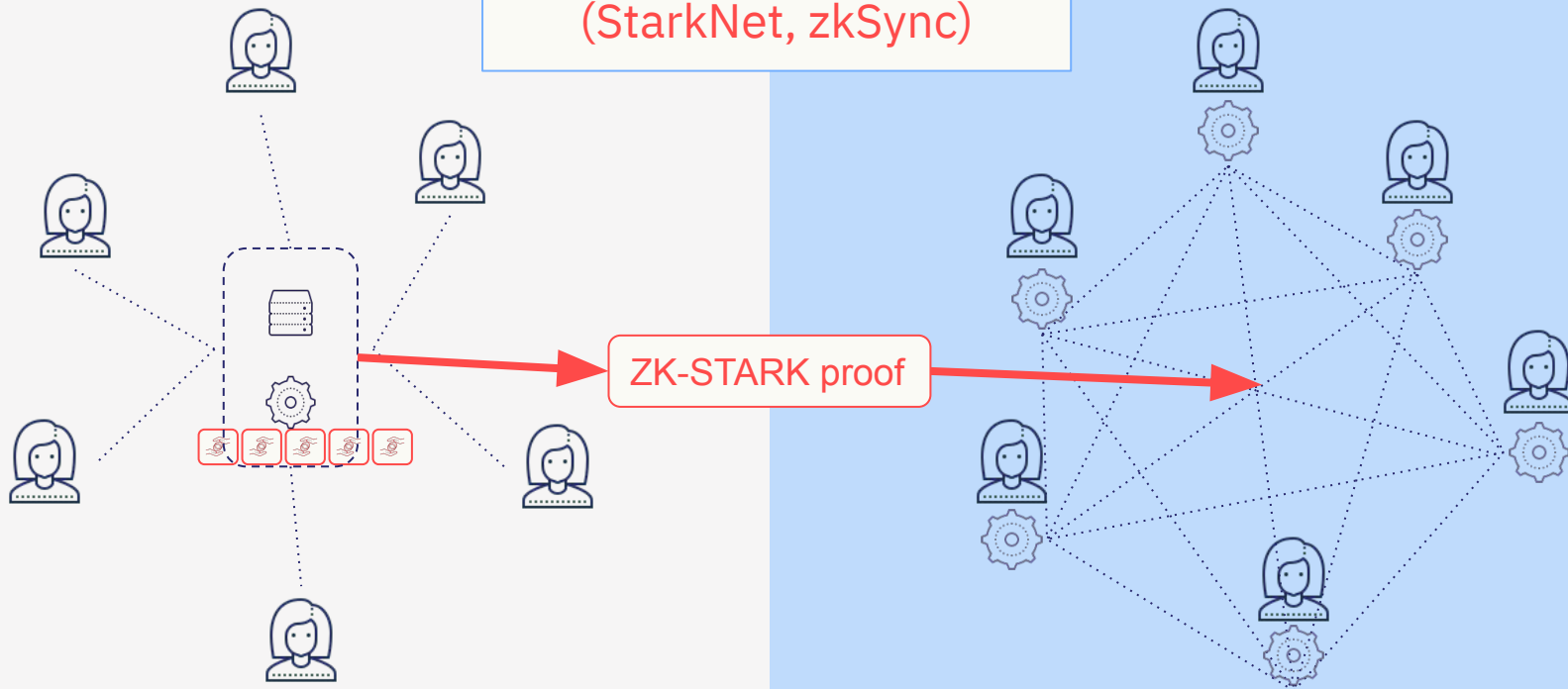
Sacrifice Privacy & Scalability



Verify (all transactions), don't trust

Validity proofs in blockchain networks

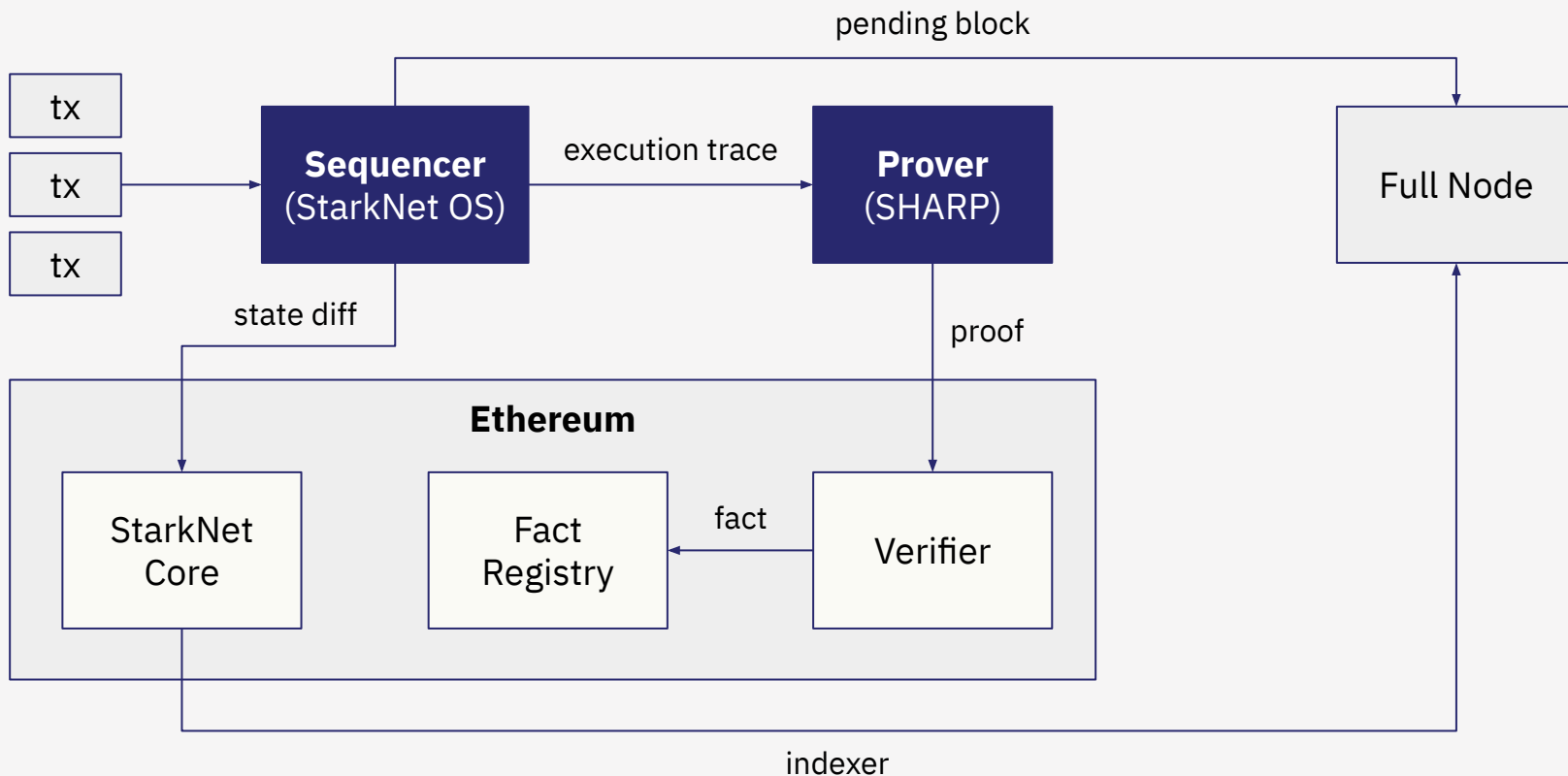
Validity Proofs (StarkNet, zkSync)



Big computer to *generate* proof
Expensive, semi inclusive (~mining)

Smol computer to *verify* proof
Fast, inclusive

StarkNet's Architecture



Summary

Why StarkNet?

Combines the power of provable execution with the transparency of an open ledger

Execution, proof generation and proof verification as a service open to anyone

Relies on rock solid Ethereum for security and availability


Execute once, verify everywhere



Thanks!

Henri Lieutaud

 @henrliehenrli

 September 2023



StarkNet edu newsletter