

# Lesson 16 - Auditing / Course Review

## Noir Example Projects

See [this list](#) of resources

- Anonymous proof of token ownership on Aztec (for token-gated access)
  - [Sequi](#)
  - [Cyclone](#)

Governance - MeloCafe

[Circuits](#) - Anonymous on-chain voting

# Verkle trees

<https://vitalik.ca/general/2021/06/18/verkle.html>

See [article](#)

and [Ethereum Cat Herders Videos](#)

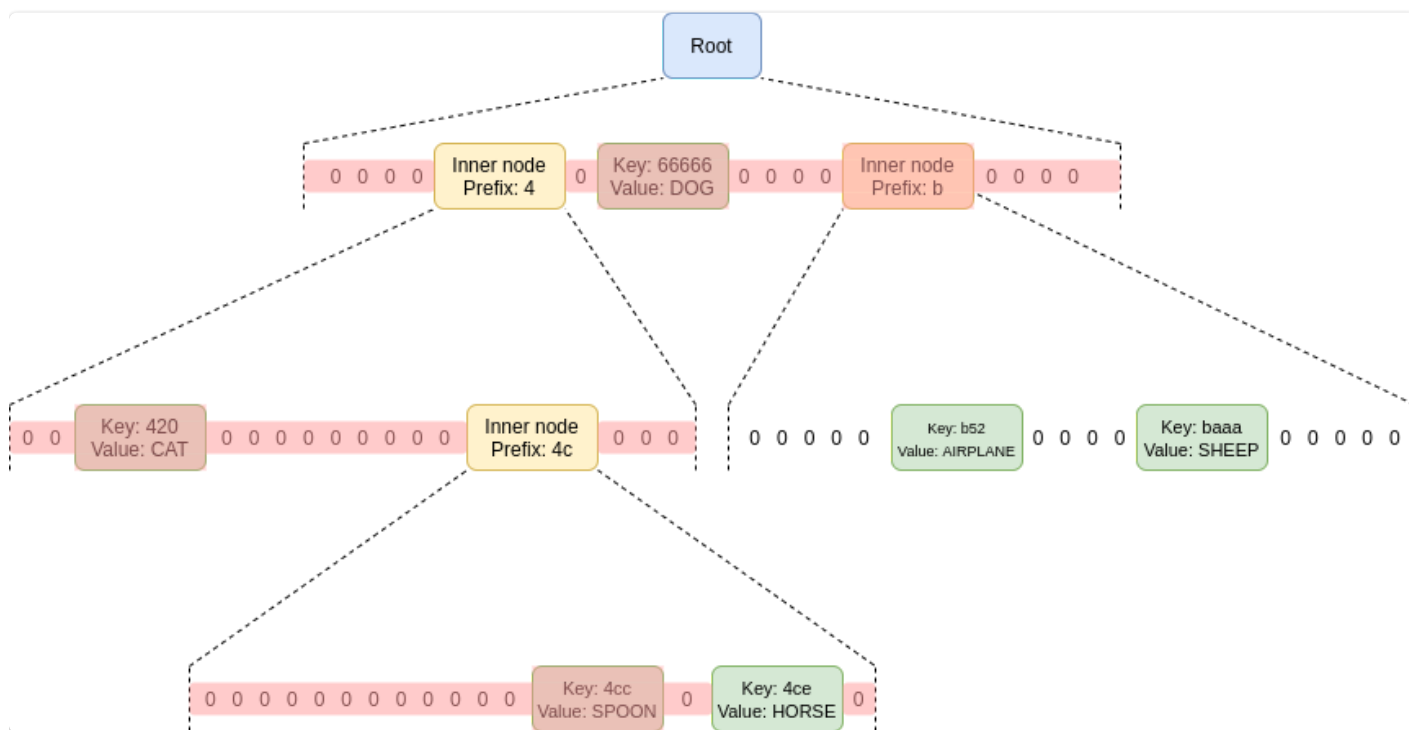
Like merkle trees, you can put a large amount of data into a Verkle tree, and make a short proof ("witness") of any single piece, or set of pieces, of that data that can be verified by someone who only has the root of the tree.

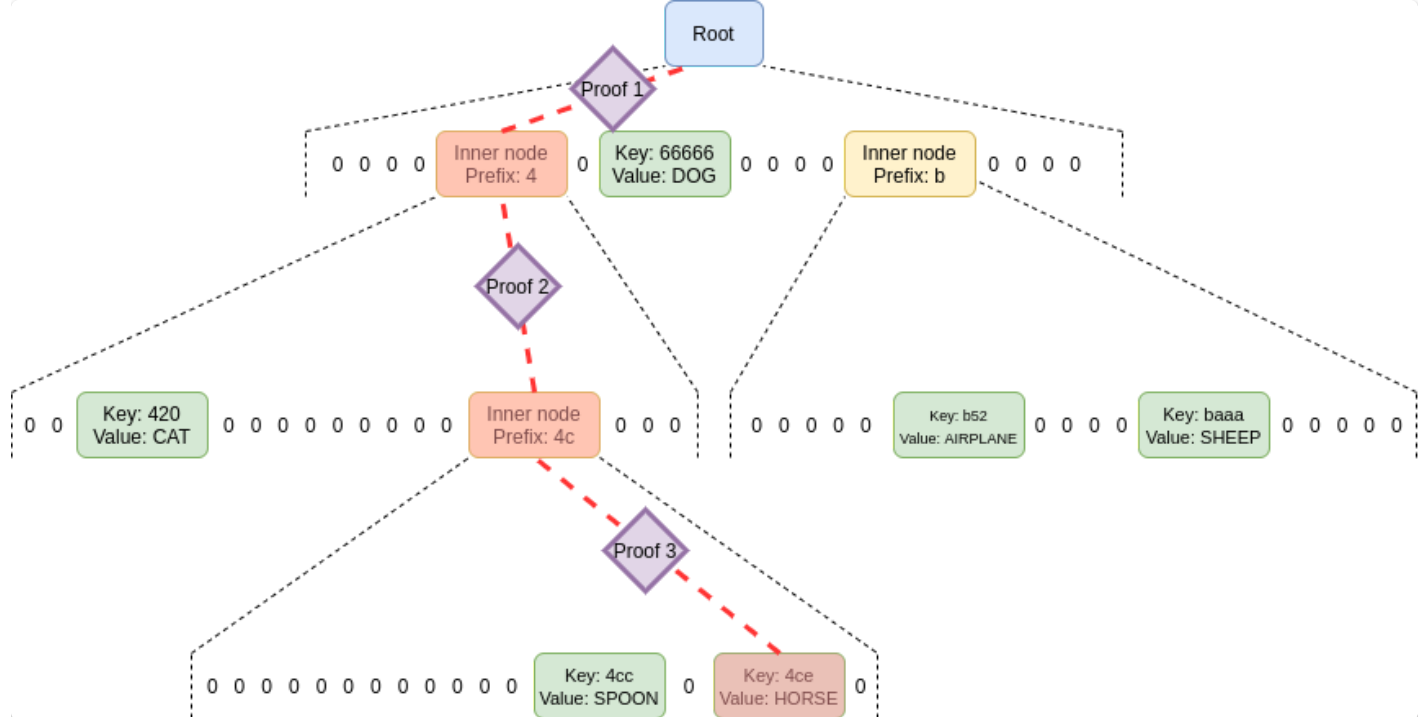
What Verkle trees provide, however, is that they are much more efficient in proof size. If a tree contains a billion pieces of data, making a proof in a traditional binary Merkle tree would require about 1 kilobyte, but in a Verkle tree the proof would be less than 150 bytes.

Verkle trees replace hash commitments with vector commitments or better still a polynomial commitment.

Polynomial commitments give us more flexibility that lets us improve efficiency, and the simplest and most efficient vector commitments available are polynomial commitments.

The number of nodes needed in a merkle proof is much greater than in a verkle proof





## Vector commitments vs. Hash

- Vector commitments: existence of an “opening”, a small payload that allow for the verification of a portion of the source data without revealing it all.
- Hash : verifying a portion of the data = revealing the whole data.



## Proof sizes

### Merkle

Leaf data +  
 15 sibling  
 32 bytes each  
 for each level (~7)

= ~3.5MB for 1K leaves

### Verkle

Leaf data +  
 commitment + value + index  
 32 + 32 + 1 bytes  
 for ~4 levels  
 + small constant-size data

= ~ 150K for 1K leaves

# ZKP Bridges

With succinct proofs, zkBridge not only guarantees strong security without external assumptions, but also significantly reduces on-chain verification cost. zkBridge provides a modular design supporting a base layer with a standard API for smart contracts on one chain to obtain verified block headers from another chain using snarks. By separating the bridge base layer from application-specific logic, zkBridge makes it easy to enable additional applications on top of the bridge, including message passing, token transfer, etc..

# Audit

A number of auditing companies are active in this space

Peckshield, Chainsecurity, Open Zeppelin

Nethermind are building their audit team, and have created tools and guidelines.

Consensys Dilligence have [announced](#) a partnership with Starkware for auditing Cairo contracts.

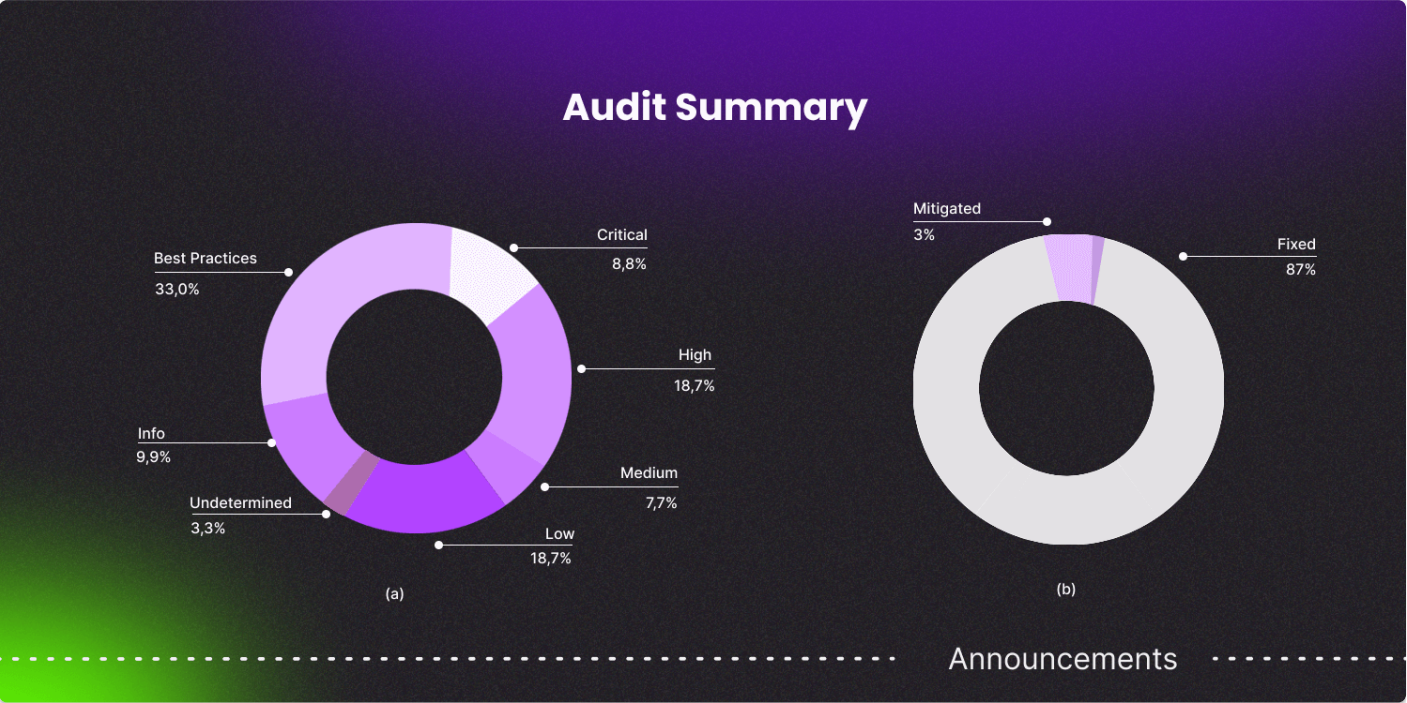
## Security Guidelines

See this [article](#) from ctrlc03 an internal auditor and developer within the EF's PSE team.

This [article](#) also gives some vulnerabilities in cairo.

# Example audits of cairo contracts

ZKX - [overview](#) of audit



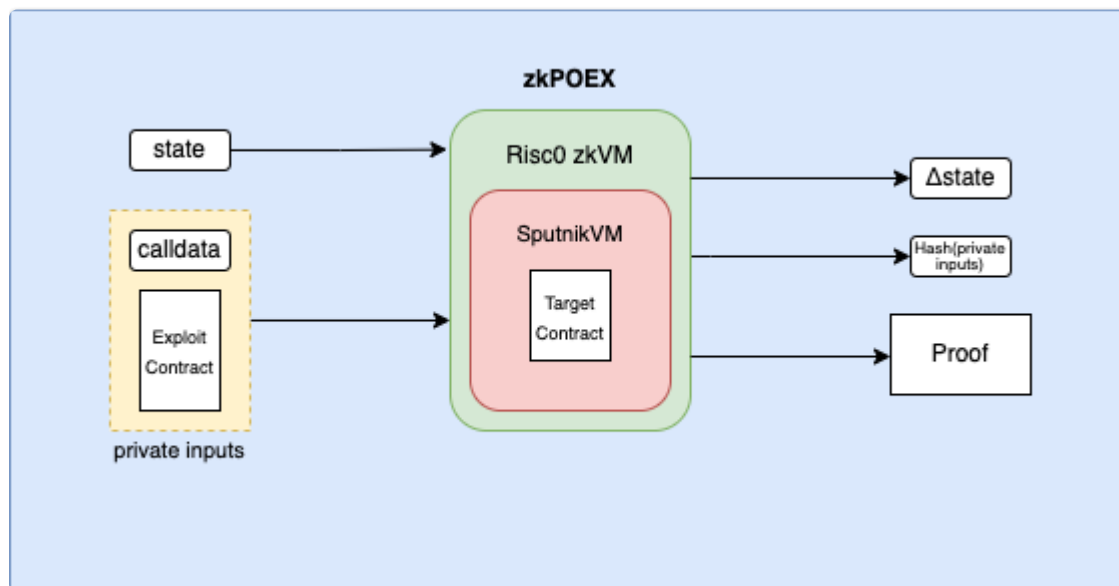
Maker DAO - [Chain Security Report](#)

## Static Analysis

Armana [Repo](#)

### Using ZKPs to verify exploits

See the recent [project](#) from ETHDenver



## Technologies Used

The project utilises the following technologies:

- [Risc0](#): A General Purpose Zero-Knowledge VM that allows to prove and verify any computation. The RISC Zero ZKVM is a verifiable computer that works like a real embedded RISC-V microprocessor, enabling programmers to write ZK proofs like they write any other code.
- [SputnikVM](#): A high-performance, modular virtual machine for executing Ethereum smart contracts.

# Useful talks from Devcon

ZKP Workshop [video](#)

Zk Application showcase [video](#)

The KZG Ceremony [video](#)

zkEVM Technical details [video](#)

Vampire SNARK from Nethermind [video](#)

Shielded Voting and Threshold encryption [video](#)

Self Sovereign Identity [video](#)

Are your ZKPs correct ? [video](#)

ZKP Performance [video](#)

Proving execution in zkEVM [video](#)

ZKP Introduction [video](#)

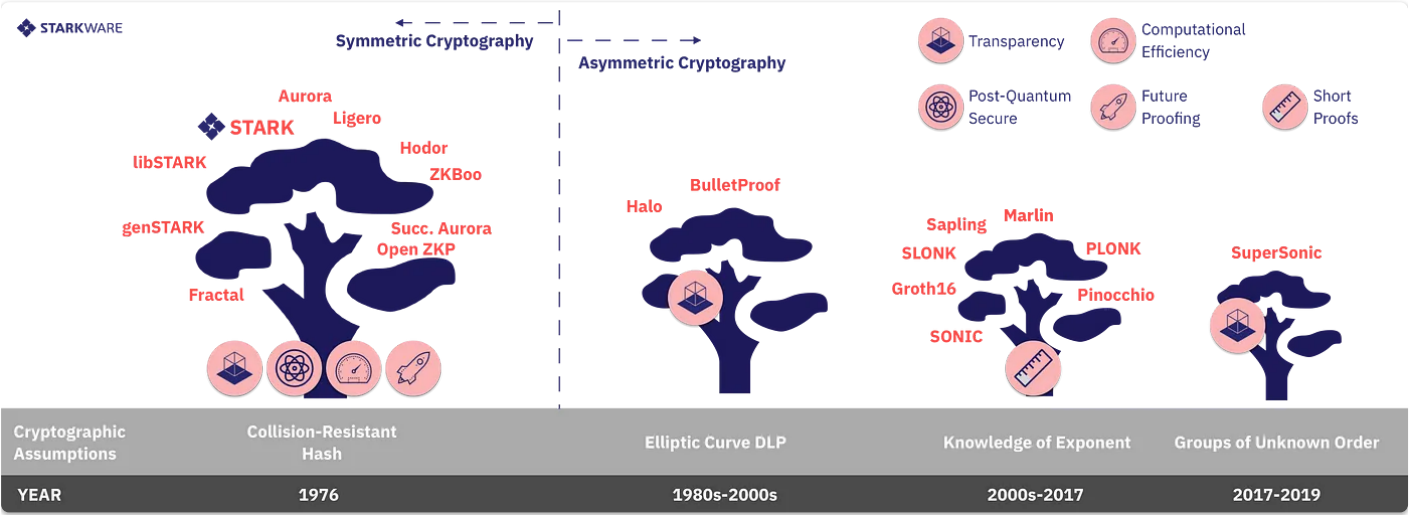
Autonomous Worlds workshop [video](#)



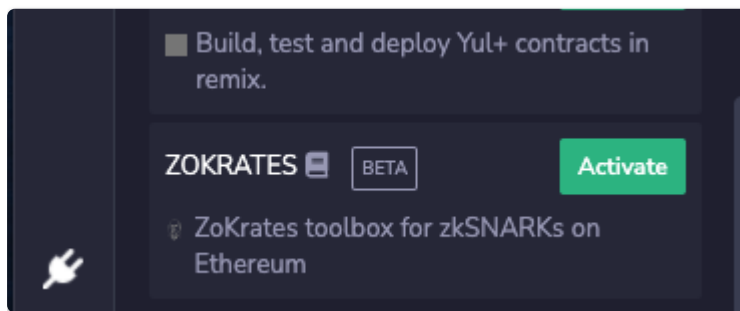
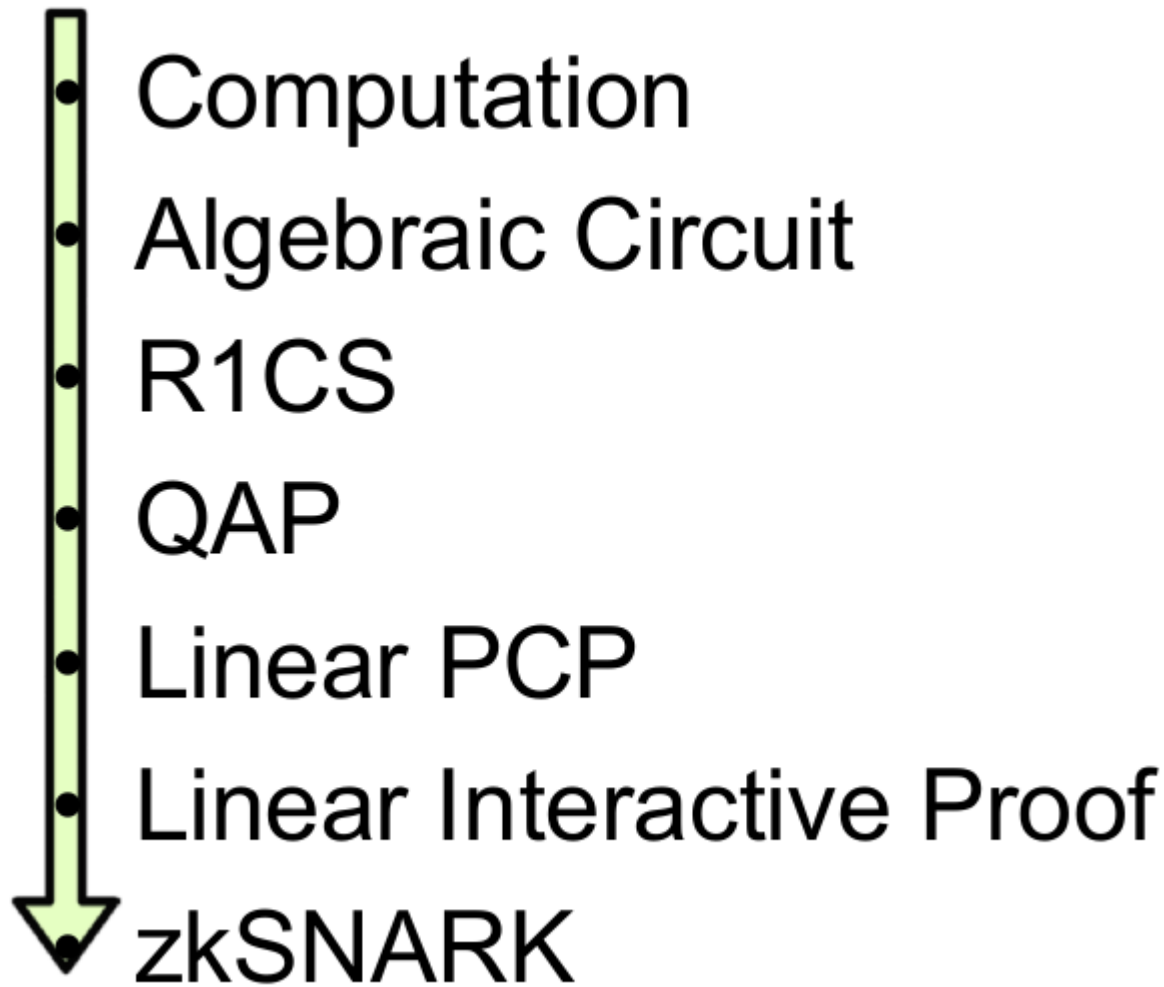
# Course Review

## Lesson 1 & 2

- Introductory maths & cryptography
- General ZKP theory



	SNARKs	STARKs	Bulletproofs
Algorithmic complexity: prover	$O(N * \log(N))$	$O(N * \text{poly-log}(N))$	$O(N * \log(N))$
Algorithmic complexity: verifier	$\sim O(1)$	$O(\text{poly-log}(N))$	$O(N)$
Communication complexity (proof size)	$\sim O(1)$	$O(\text{poly-log}(N))$	$O(\log(N))$
- size estimate for 1 TX	Tx: 200 bytes, Key: 50 MB	45 kB	1.5 kb
- size estimate for 10.000 TX	Tx: 200 bytes, Key: 500 GB	135 kb	2.5 kb
Ethereum/EVM verification gas cost	$\sim 600k$ (Groth16)	$\sim 2.5M$ (estimate, no impl.)	N/A
Trusted setup required?	YES 😞	NO 😊	NO 😊
Post-quantum secure	NO 😞	YES 😊	NO 😞
Crypto assumptions	Strong 😞	Collision resistant hashes 😊	Discrete log 😞



## Lesson 3

- ZKP use cases / rollups

## Lessons 4 - 5

- Starknet / Cairo / Warp

# Cairo

## Other blockchains developers:



**Rust is better**



**NOOOOOOOO! Solidity  
is far better**

## Starknet developers:



**Cairo is hell**



**Yes**

## Lesson 6

- Confidential tokens



Zcash is a privacy-protecting, digital currency built on strong science.

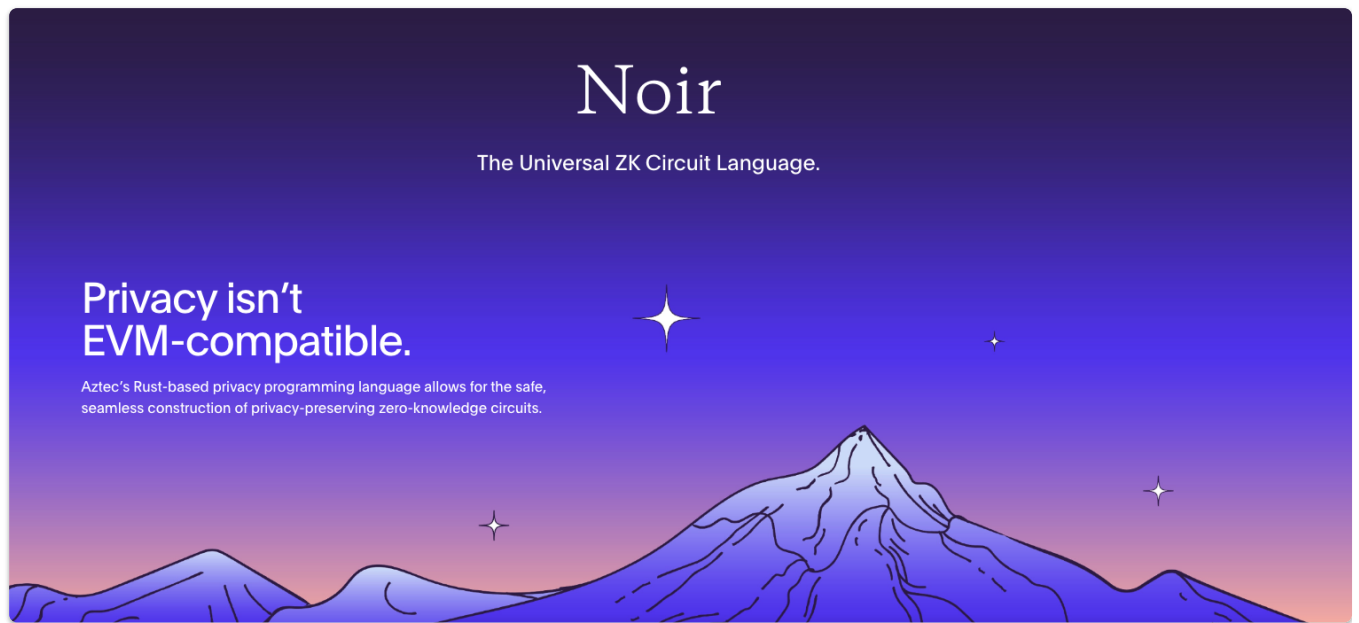


Also Nightfall , ZKDai



## Lesson 7

- Noir



## Lesson 8

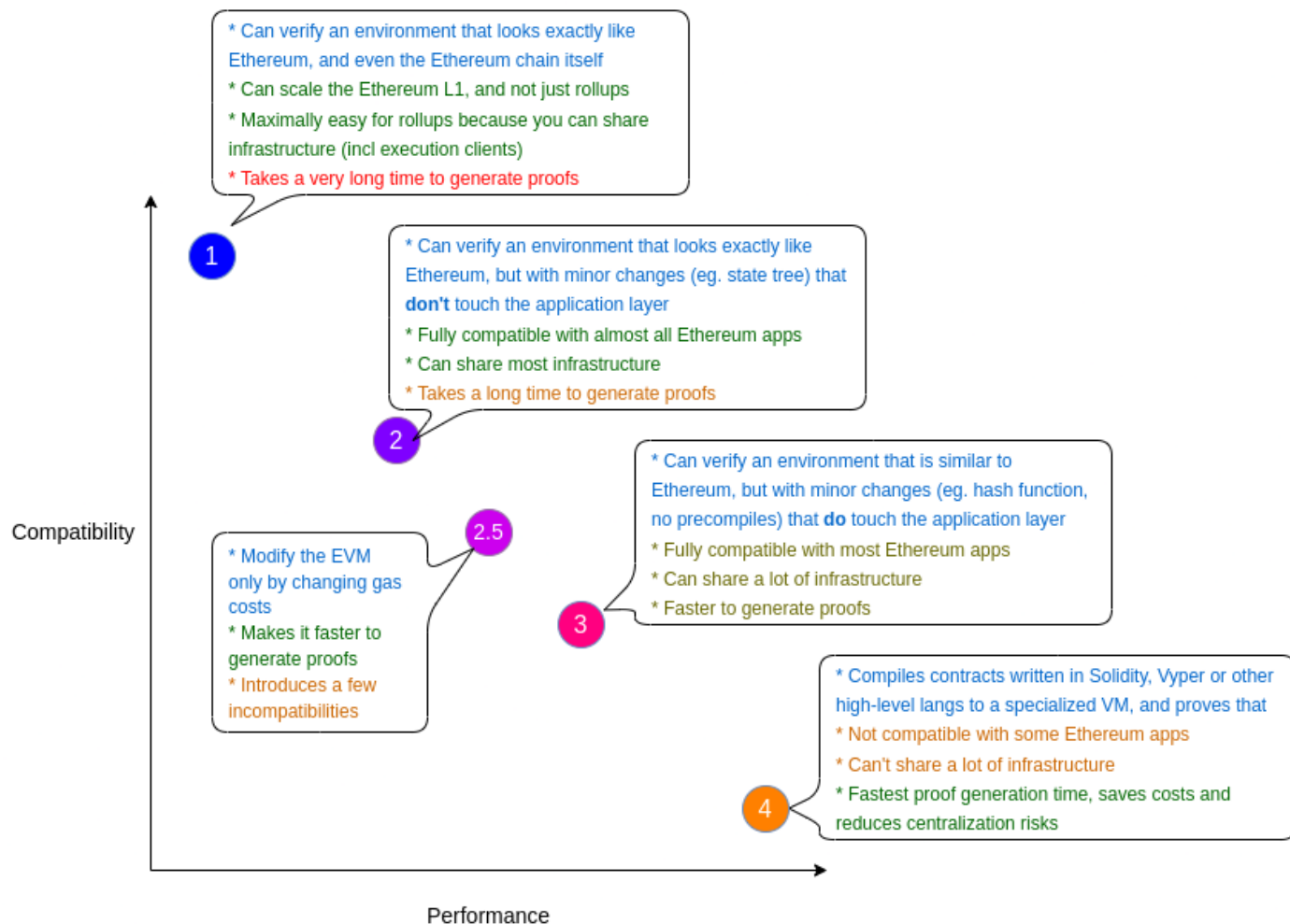
- Cairo contracts

## Lesson 9

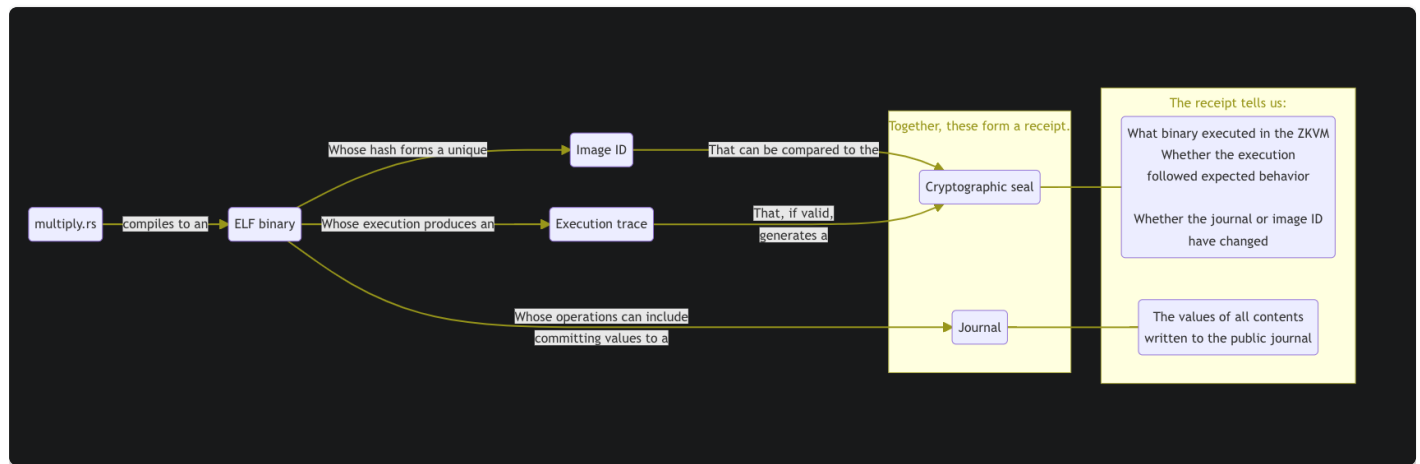
- Mina
- zkApps



## zkEVM solutions

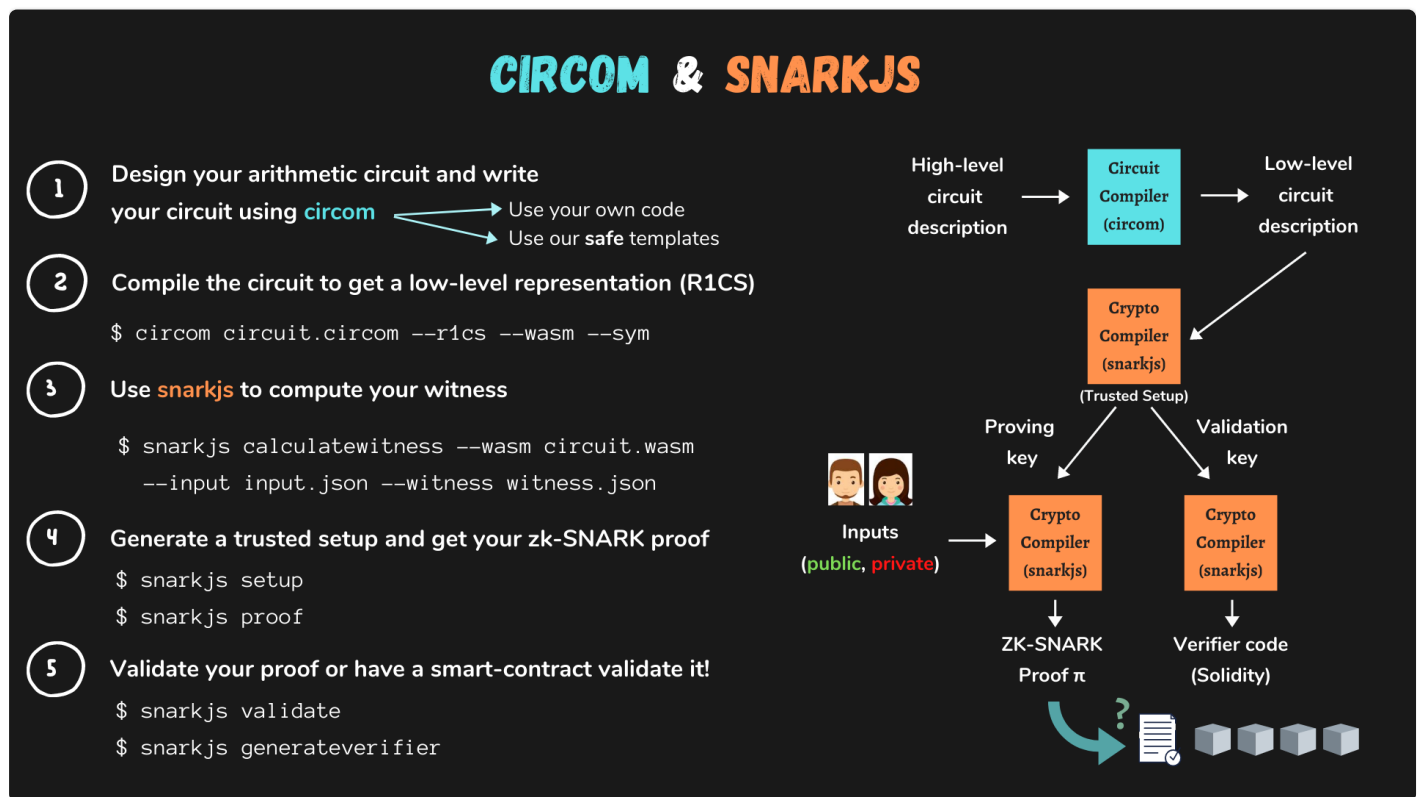


## Lesson 11



- Risc zero
- PLONK

## Lesson 12



- Circom
- SNARK theory

## Lesson 13

- STARK theory

## Lesson 14

- Alternative technologies
- Voting systems



## Lesson 15

- Identity Solutions
- Oracles

## Lesson 16

- Auditing / verification

# Developer Options

## Writing zkp programs

Many languages are rust like

- Cairo
- Noir
- Leo (Aleo)
- Risc Zero

Others :

- Circom
- Snarky.JS (Mina)
- Lurk (Filecoin)
- Solidity (for zkEVM)
- Solidity + WARP -> Cairo

In addition to the DSL -> SNARK / STARK approach, there are other cryptographic techniques that could be useful, see the alternative techniques lesson.

There are many areas to get involved in

- L2 solutions
- Privacy / Identity projects
- ZK-ML
- zk Games
- Security / Auditing

At the protocol level

- STARKS
- SNARKS - PLONK derivatives
- zk VMs
- Recursive proofs

## Hackathons and Builder Programs

Encode [Hackathons](#)

Mina [zkIgnite](#)

Aztec [grants](#)

Autonomous Worlds [hackathon](#)

## Communities / Groups

Mina UK [Meetup group](#)  
Starknet [Meetup groups](#)

## Resources

Zero Knowledge [Podcast](#)

Starknet [podcast](#)

ZK Hack and ZK Study Club [Videos](#)

ZK Whiteboard [sessions](#)

Cairo mummies [course](#)

Berkley ZKP [MOOC](#)

[Alex Pinto's Blog](#)

Extropy Foundation [collection](#)