

# Lesson 9

## This Week

Monday - Mina

Tuesday - zkEVM Solutions

Wednesday - Risc Zero

Thursday - Circom

### Mina



**SNARK**



**SNARK**



**SNARK**

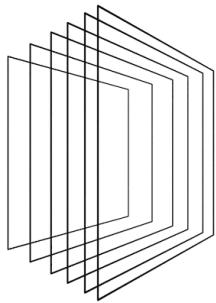


**3ED**

## Mina Blockchain

**22KB<sup>1</sup>**

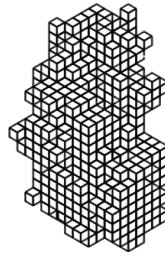
FIXED SIZE



## Other Blockchains

**300GB<sup>2</sup>**

INCREASING  
SIZE



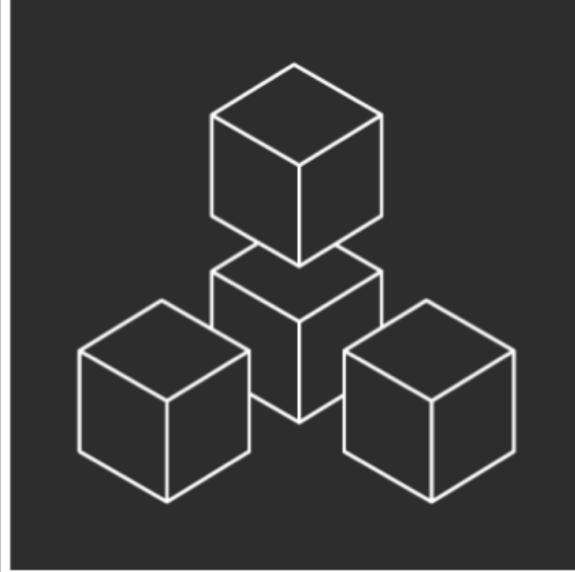
Mina is the first cryptocurrency protocol with a **succinct** blockchain. With Mina, no matter how much the usage of the network grows, the blockchain always stays the same size - **about 22kb**.

What data is needed for usable representation of the blockchain?

- A clear, usable representation of the basic data being queried of a state - accounts balances
- Data that a node needs in order to verify that this state is real in a trustless manner
- The ability to broadcast transactions on the network to make a transfer

## Roles in Mina

**Block producer**



#### AS A BLOCK PRODUCER

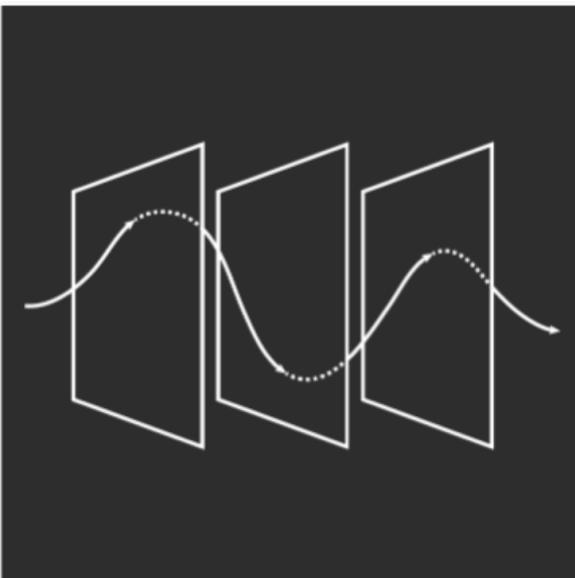
Block producers are akin to miners or stakers in other protocols. By staking Mina, they can be selected to produce a block and earn block rewards in the form of coinbase, transaction fees and network fees. Block producers can decide to also be SNARK producers.



A block in Mina is constituted of:

- Protocol state
  - Genesis state hash
  - Blockchain state
  - Consensus state
  - Consensus constants
- Protocol state proof
- Staged ledger diff
- Delta transition chain proof
- Current protocol version
- Proposed protocol version

#### [Snark worker](#)

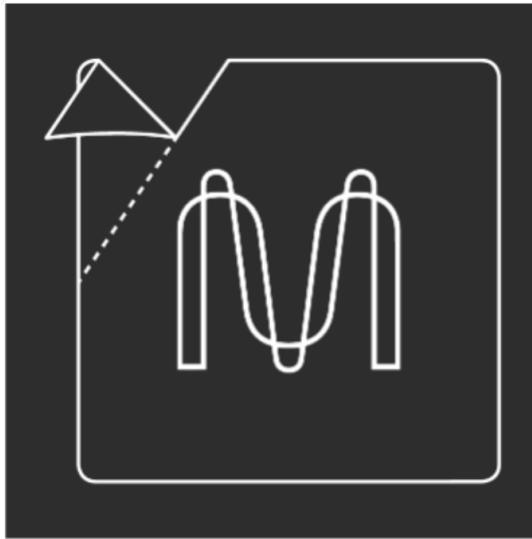


#### AS A SNARK PRODUCER

The second type of consensus node operator on Mina, snark producers help compress data in the network by generating SNARK proofs of transactions. They then sell those proofs to block producers in return for a portion of the block rewards.



#### [Professional block producer](#)



#### AS A PROFESSIONAL BLOCK PRODUCER

Because staking requires nodes to be online, some may choose to delegate their Mina to staking pools. These groups run staking services in exchange for a fee, which is automatically deducted when the delegator gets selected to be a block producer.



See [guide](#)

## Mina's consensus mechanism

Mina's consensus mechanism is an implementation of Ouroboros Proof-of-Stake. Due to Mina's unique compressed blockchain, certain aspects of the algorithm have diverged from the Ouroboros papers, and the version Mina uses is called **Ouroboros Samisika**. VRFs are used to decide whether to produce a block, the probability is proportional to the producer's stake.



## RECURSIVE COMPOSITION OF PROOF (SUCCINCTNESS)

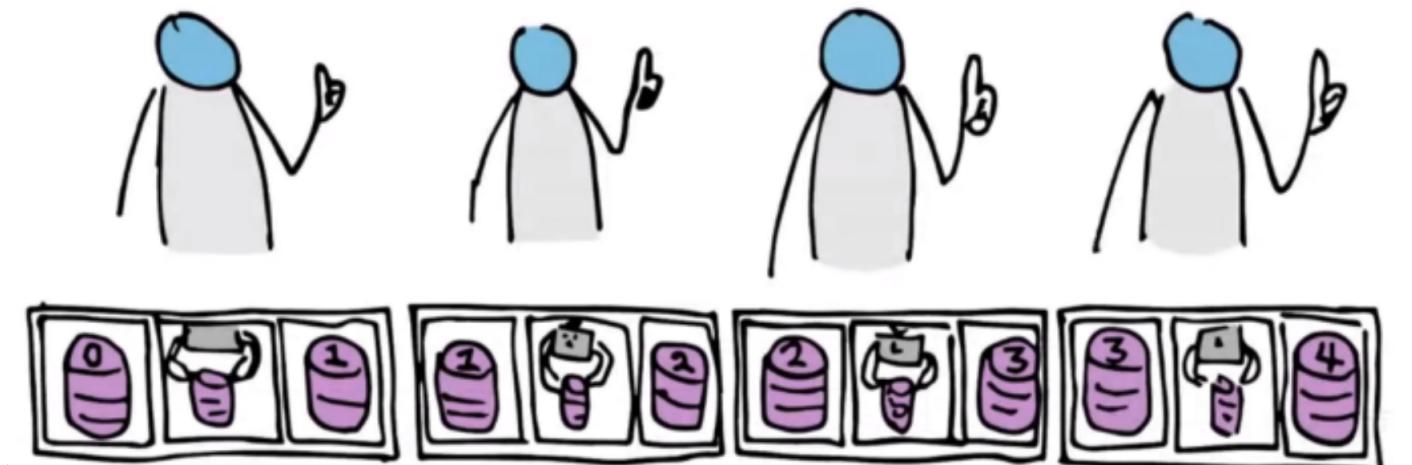
The blockchain is dynamic and new blocks keep getting added to it. However, we

would like to ensure succinctness at any given point in time. Therefore, as the blockchain “grows”, we compute a new SNARK proof that not only validates the new blocks, but also the existing SNARK proof itself. The notion of a SNARK proof

that attests to the verifiability of another SNARK proof is the notion of “incrementally-computable SNARK”

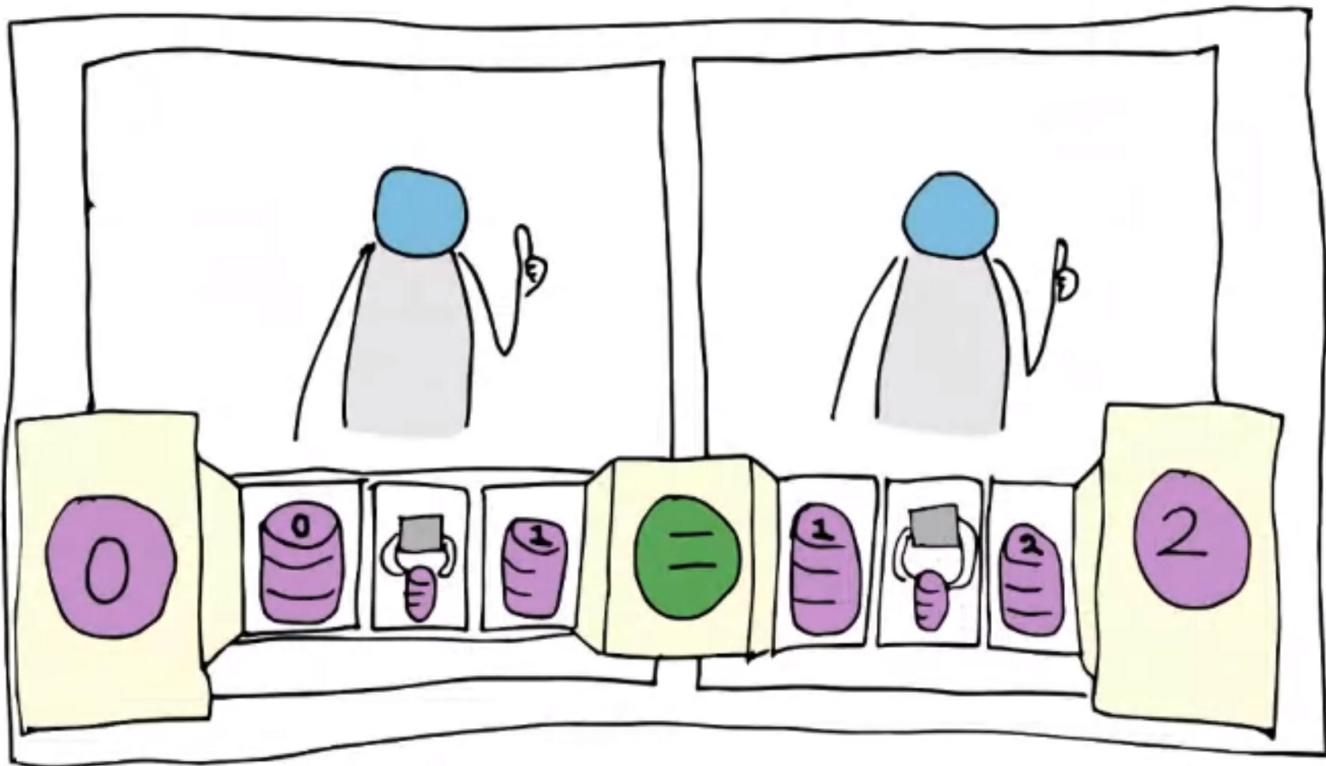
from Mina: Decentralized Cryptocurrency at Scale ([whitepaper](#))

### Chaining certificates



Better to use recursive proofs :

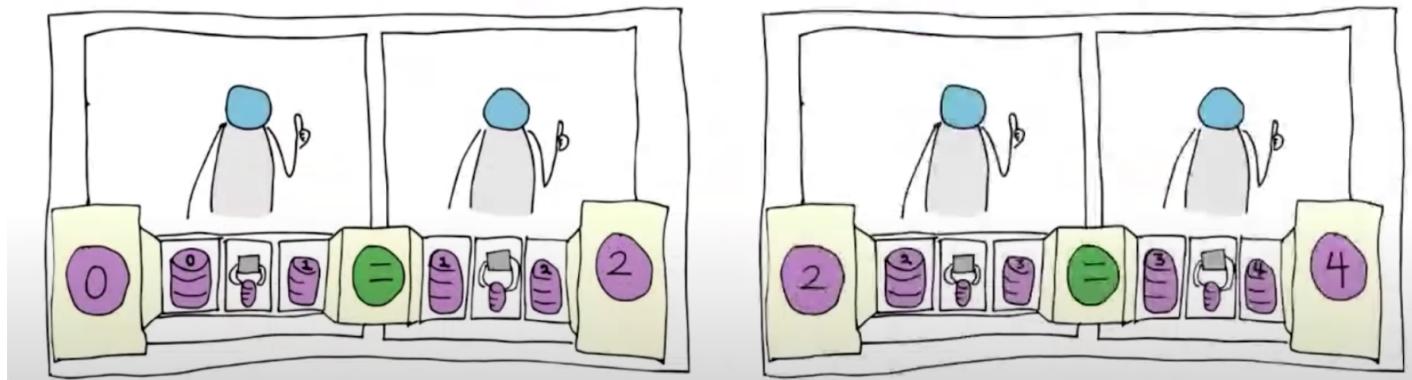
Thus, "You can get from **DB 0 to DB 2**"

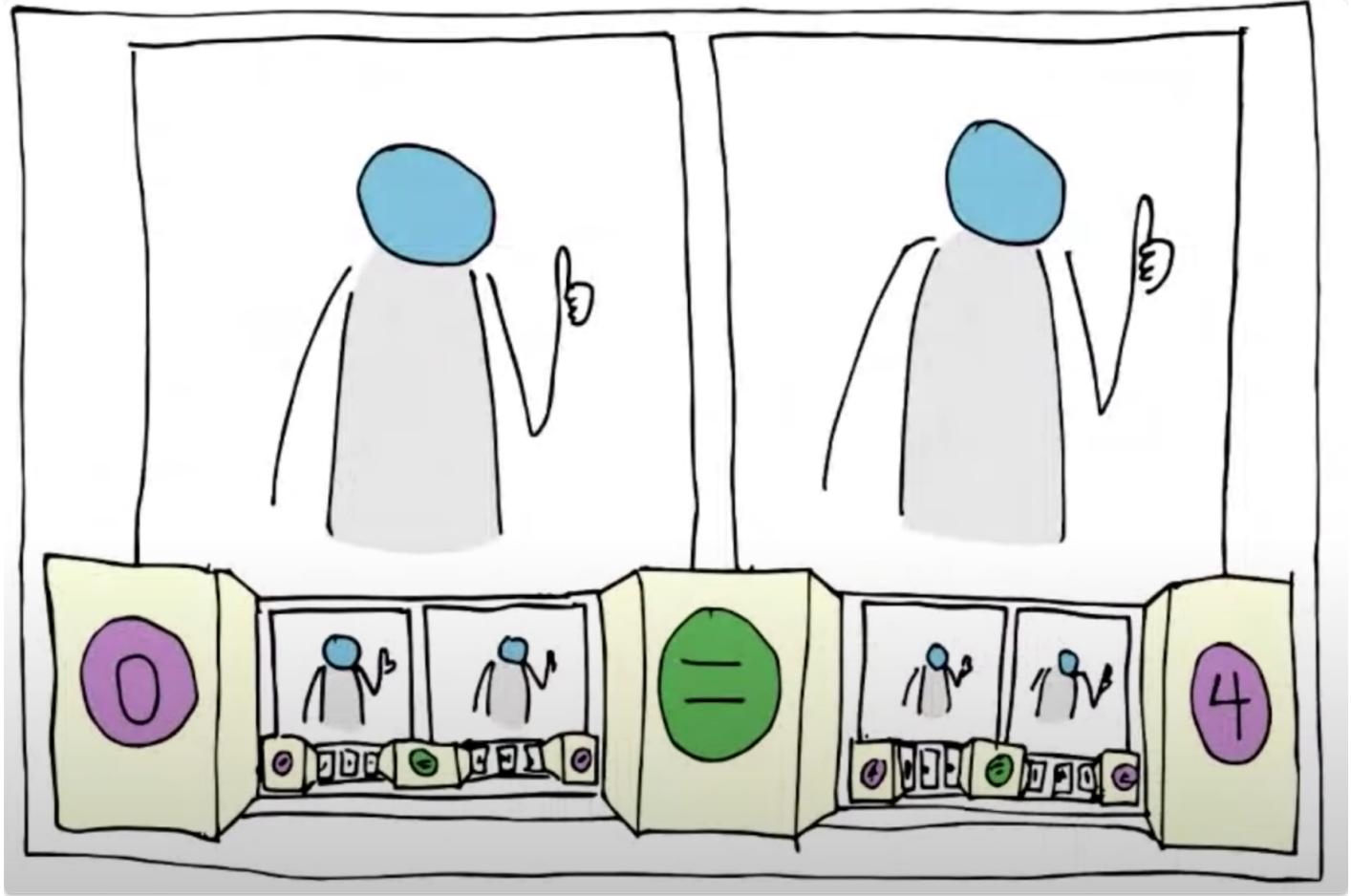


It's a SNARK, so still ~1 kB

"You can get from  
**DB 0 to DB 2**"

"You can get from  
**DB 2 to DB 4**"





See [video](#)

## Running Mina in a browser See [Open Mina](<https://openmina.com/>) See [article](<https://medium.com/openmina/introducing-the-web-node-an-in-browser-mina-node-that-verifies-blocks-and-transfers-funds-ebc59a57e79a>)

The screenshot shows a dark-themed web page for 'OpenMina'. At the top, a large heading reads: 'With the Web Node, anyone can verify blocks and transfer funds directly through their browser'. Below it is a subtitle: 'An in-browser Mina node capable of validating blocks'. A prominent blue button labeled 'Launch Web Node' is centered. The page then displays three features in separate boxes:

- Security** (with a shield icon): Distributing control across multiple Web Nodes makes it difficult for malicious actors to compromise the network.
- Flexibility** (with a smartphone icon): Launch the Web Node from any device with an internet browser - you can even use your smartphone.
- Inclusivity** (with a person icon): Each new Web Node launched further democratizes the Mina blockchain and makes it even more transparent.

# Community

<https://awesome.mina.tools/>

<https://minacrypto.com/>



## Join Mina's zkIgnite, Cohort 2

26 projects received funding in Cohort 1 to start building zkApps on Mina. What will you build in Cohort 2?

Cohort 2 starts on May 24th, 2023 and will provide hands-on support and funding to those wanting to build on zero knowledge (ZK) – from ideation, technical mentorship, to funding your proof of concept and even VC support afterwards.

With more grants and rewards, there will be a variety of opportunities to get involved for different participants, from founders and builders to marketers and designers.

Full details for Cohort 2 will be shared in the coming weeks, sign up below to register your interest and receive next steps on getting involved.

↑  
TOP

## Details

### Staking on Mina

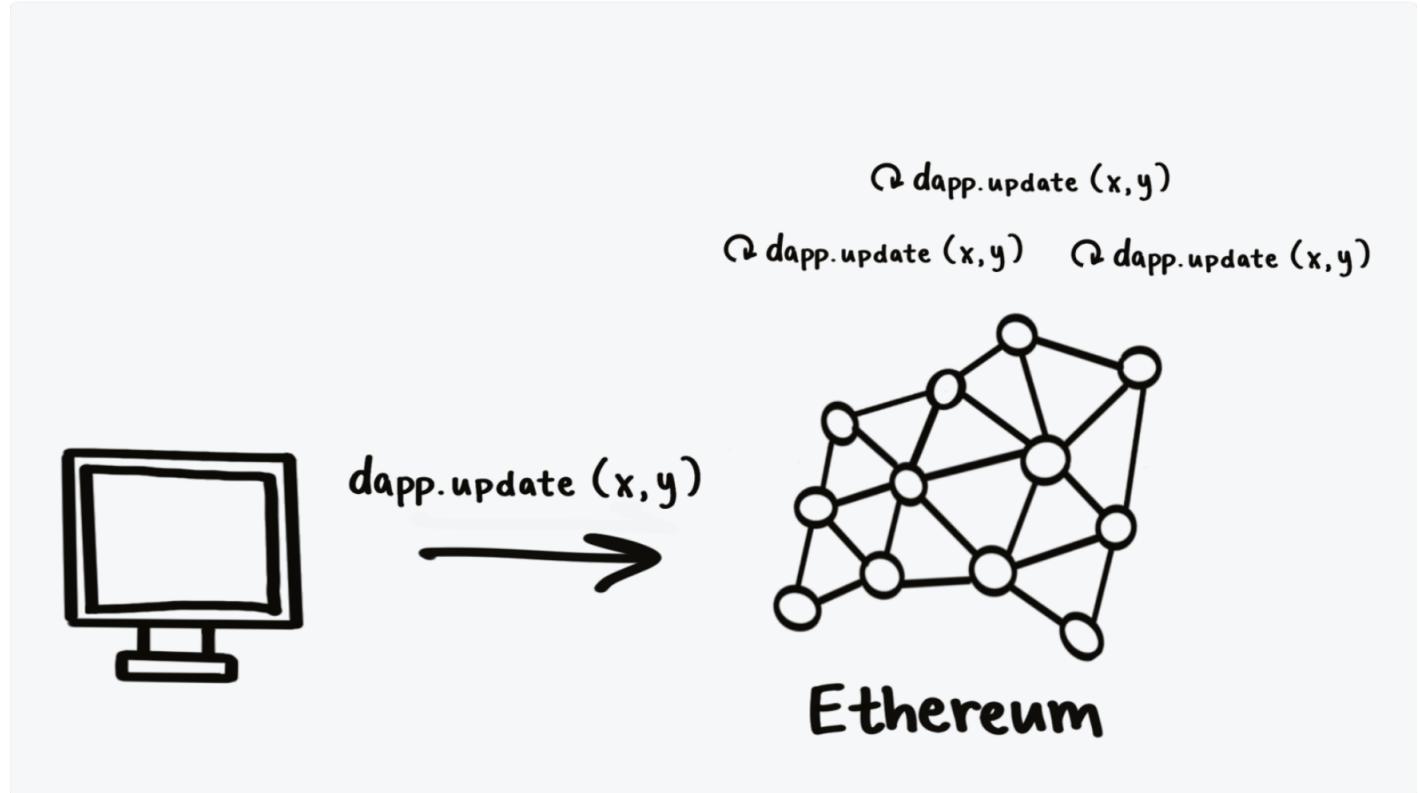
See [guide](#)

## zkApps

zkApps ("zero-knowledge apps") are Mina Protocol's smart contracts that use an off-chain execution and mostly off-chain state model.

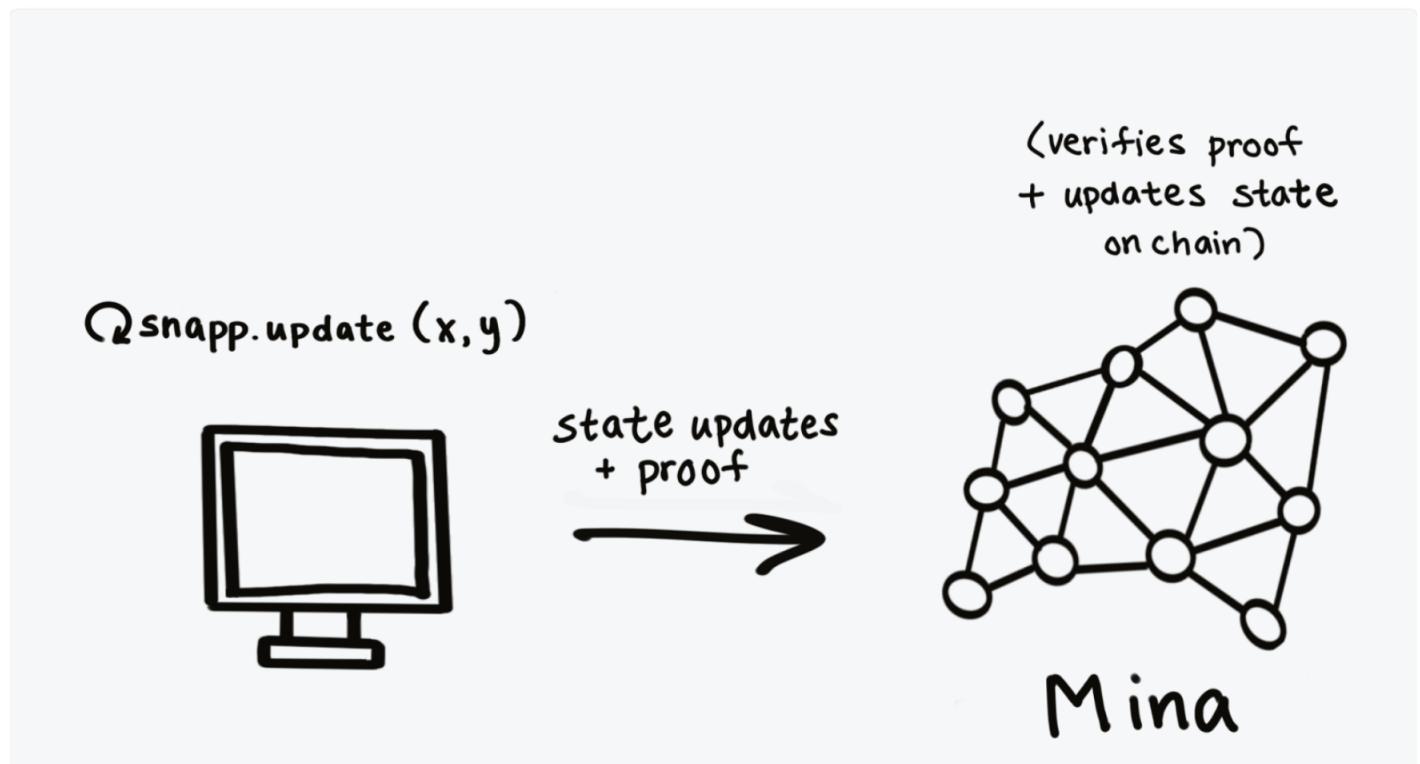
This allows for private computation and state that can be either private or public.

### Ethereum dapps

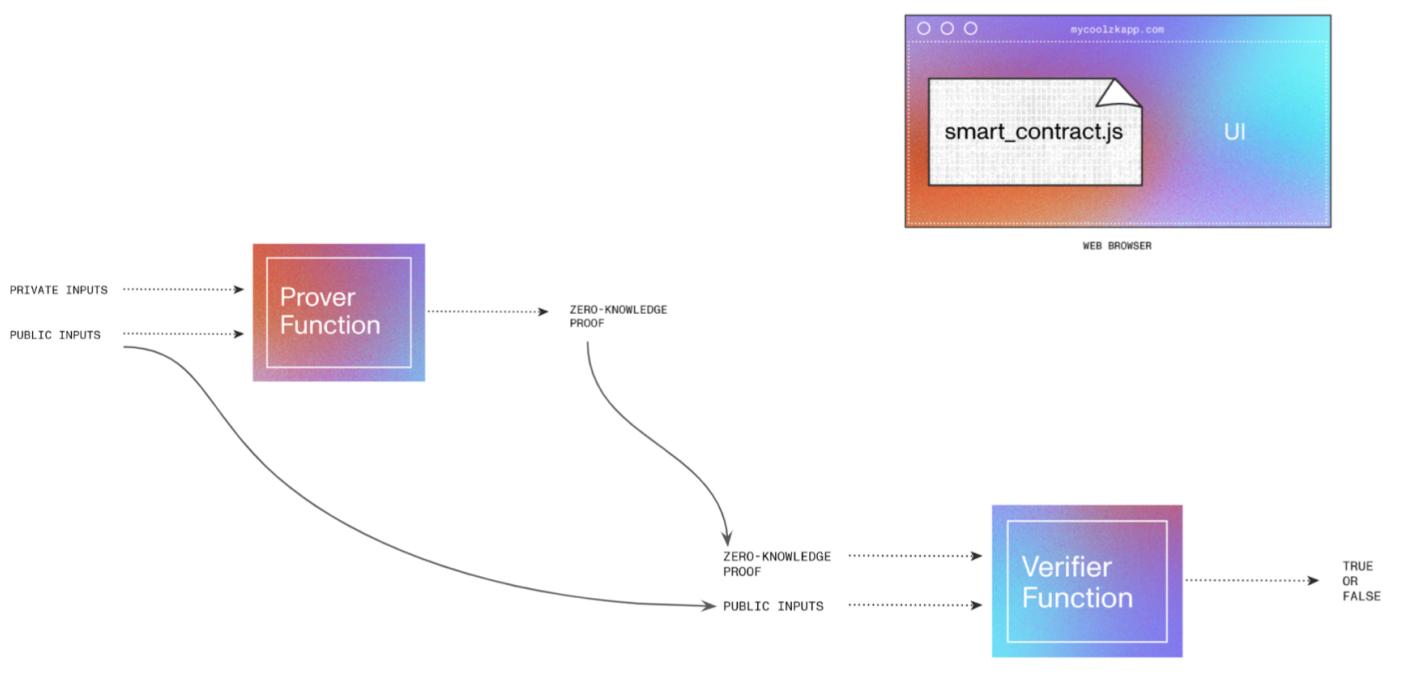
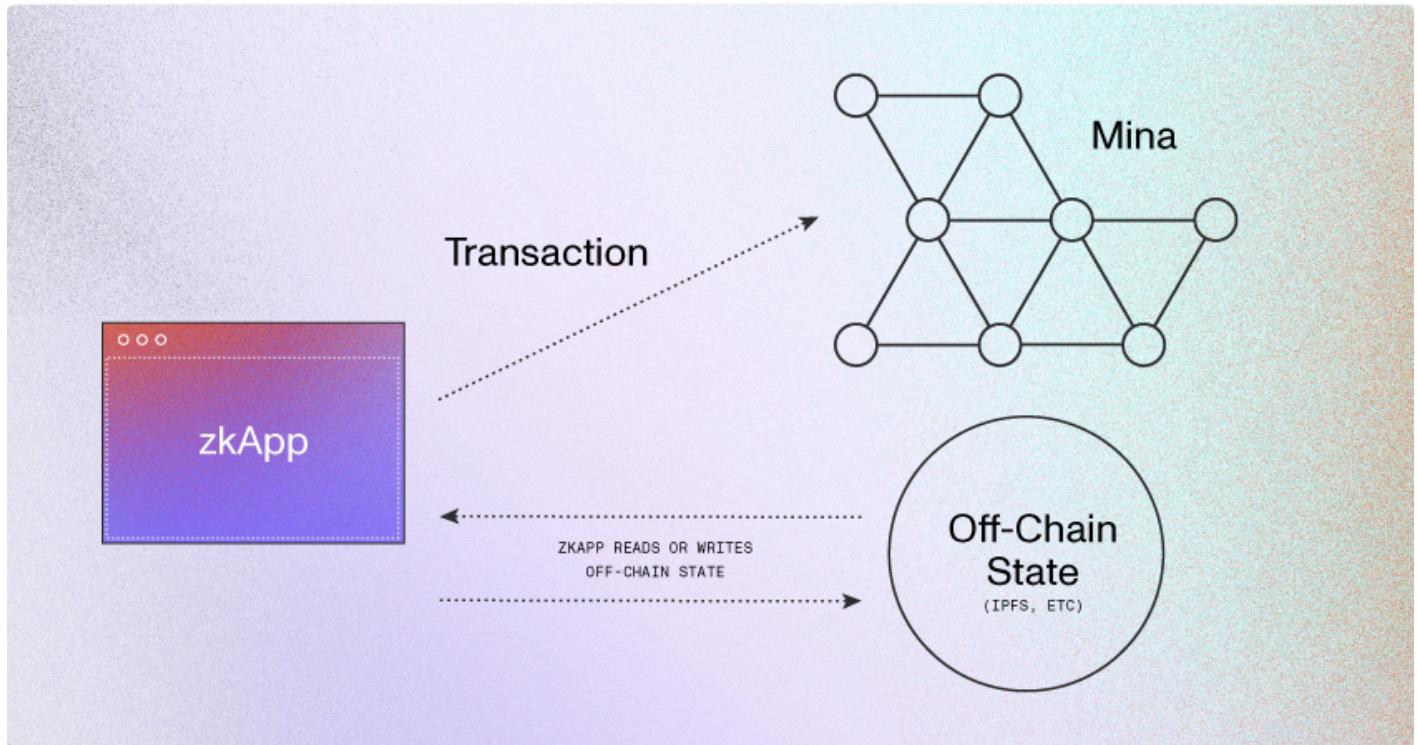


Ethereum uses **on-chain computation**.

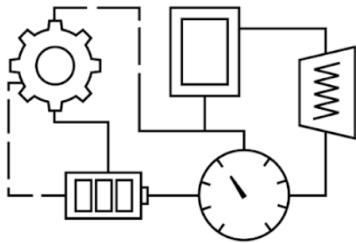
### Mina zkApps



Mina uses off-chain computation & on-chain verification.

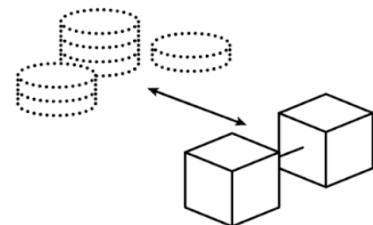


## zkApps use cases



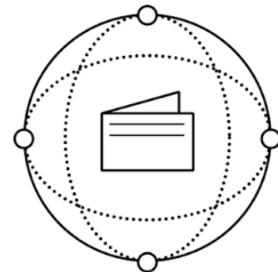
### BUILD ZKAPPS<sup>3</sup> PRIVACY-ENABLED APPS

Develop dapps that use zero knowledge to ensure data-level privacy, verifying requirements without exposing the underlying user information.



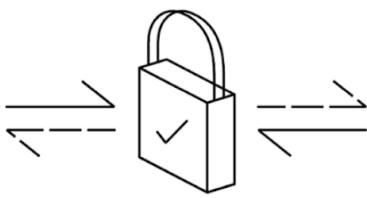
### POWER ENTERPRISE INTEROPERABILITY

Use Mina to combine the cost-efficiency and privacy of a private chain with the interoperability of a public chain.



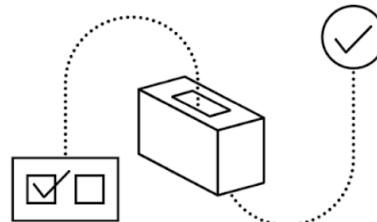
### MINIMIZE TRANSACTION FEES

Power trustless e-commerce and global peer-to-peer transactions without using centralized intermediaries, or paying costly transaction fees.



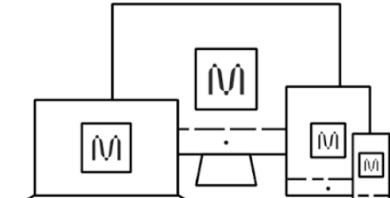
### POWER SECURE & FAIR FINANCIAL SERVICES

Ensure lenders only use fair criteria to make decisions and securely verify relevant information without accessing private user data.



### ENABLE PRIVATE & AUDITABLE ELECTIONS

Guarantee fully verifiable and auditable elections, while keeping the process private and protecting individuals' voting information.



### ACCESS MONEY FROM ANYWHERE IN THE WORLD

With a 22kb<sup>1</sup> Mina chain, access peer-to-peer stablecoins and tokens via smartphone and bring hard-earned money anywhere you go.

## zkBridge ETH & WMina example

The bridge is currently only one way- i.e MINA state can be read on Ethereum , but not the other way around.

How does the bridge works ?

- Retrieve MINA state proof and the verification key.
- Generate an Auxiliary proof for the state received.
- Post the Auxiliary proof to Ethereum blockchain , where a smart contract will update only if valid.

[ETH] A Wrapped MINA contract on ETH will be created which will be the minter. This contract has a MINA address where a user will deposit funds into.

[MINA] User deposits funds into address above on MINA blockchain.

[MINA] Deposit transaction gets committed.

[ETH] Auxiliary Proof of MINA state is submitted to ETH.

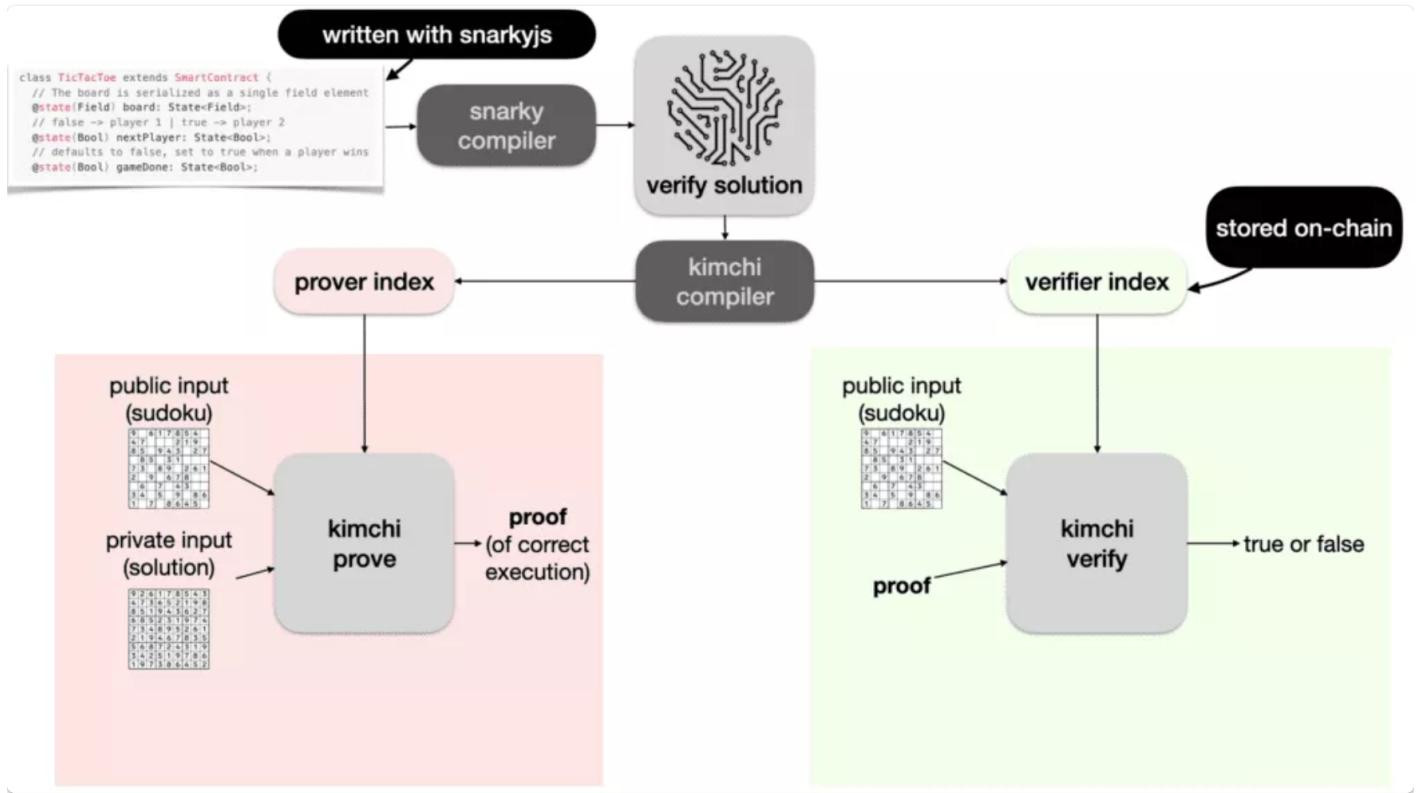
[ETH] MINA state proof is validated and confirmed.

[ETH] Wrapped MINA Contract is called which uses the proof and validates the balance and mints Wrapped MINA on Ethereum.

---

# How Mina creates proofs.

## Pickles and Kimchi



## Pickles

Pickles has two components:

- core zk-SNARK,
- developer toolkit (containing a wide array of library functionality and the Pickles Inductive Proof System)

## Kimchi

The heart of the Mina proving system is Kimchi, this is the protocol that creates the proofs.

Kimchi is based on the Plonk family of zkSNARK proof systems.

One notable improvement over Plonk is that Kimchi doesn't require a trusted setup.

This is achieved by adding a polynomial commitment to the setup (similar to Bulletproofs)

## Other Improvements in Kimchi

- The gates used in the circuit have changed from being generic gates (which could represent any functionality) to specific gates to target known functionality such as Poseidon Hashes.
- More than 2 inputs are allowed for gates.
- The use of lookup tables for public data or simple boolean operations.

## How does Pickles compare to other proof systems ?

System	Setup	Programmability	Proof Size	Prover Speed	Recursion Ready	Open Source
<b>Pickles</b>	Trustless	Full	8kB	Fast	Arbitrary	Yes
<b>AIR STARKs</b>	Trustless	Partial, complex constraint system model limiting to succinct circuits	100kB	Very fast	No	No
<b>Halo</b>	Trustless	Full	3.5kB	Fast	Linear	Yes
<b>BN128 PLONK</b>	Trusted, universal	Full	0.5kB	Fast	No	Yes
<b>MNT Groth16</b>	Trusted, per-circuit	Full	1.5kB	Medium	Arbitrary	Yes

## Curves used

Mina uses the curves Pallas and Vesta (collectively Pasta).

### Pallas

- curve equation:  $y^2 = x^3 + 5$
- base  
field: 2894802230932904885589274625217197696336305648194156  
0715954676764349967630337
- scalar  
field: 2894802230932904885589274625217197696336305648194164  
7379679742748393362948097
- mina  
generator: (1, 12418654782883325593414442427049395787963493412  
651469444558597405572177144507)
- arkworks generator: (-1, 2)

### Vesta

- curve equation:  $y^2 = x^3 + 5$
- base  
field: 2894802230932904885589274625217197696336305648194164  
7379679742748393362948097
- scalar  
field: 2894802230932904885589274625217197696336305648194156  
0715954676764349967630337
- mina  
generator: (1, 11426906929455361843568202299992114520848200991  
084027513389447476559454104162)
- arkworks generator: (-1, 2)

## Resources

### Extropy Medium Article

A great introduction to ZKPs and Mina is the Mina [Book](#)

### Zero Knowledge Podcast ZKP Intro

Zero Knowledge Podcast [Mina Episode](#)



# SnarkyJS

zkApps are written in TypeScript using SnarkyJS.

SnarkyJS is a TypeScript library for writing smart contracts based on zero-knowledge proofs for the Mina Protocol. It is included automatically when creating a new project using the Mina zkApp CLI.

## Primitive data types

Field elements are the basic unit of data in zero-knowledge proof programming. Each field element can store a number up to almost 256 bits in size.

```
new Bool(x); // accepts true or false

new Field(x); // accepts an integer, or a numeric string

new UInt64(x); // accepts a Field – useful for constraining
numbers to 64 bits

new UInt32(x); // accepts a Field – useful for constraining
numbers to 32 bits

PrivateKey, PublicKey, Signature; // useful for accounts and
signing

new Group(x, y); // a point on our elliptic curve, accepts two
Fields/numbers/strings

Scalar; // the corresponding scalar field (different than Field)
```

In typical programming, you might use:

```
const sum = 1 + 3;
```

In SnarkyJS, you would write this as:

```
const sum = new Field(1).add(new Field(3));
```

This can be simplified as:

```
const sum = new Field(1).add(3);
```

## Conditionals (Important !)

Traditional conditional statements are not supported by

SnarkyJS:

```
// this will NOT work
if (foo) {
x.assertEquals(y);
}
```

Instead, use SnarkyJS' built-in `Circuit.if()` method, which is a ternary operator:

```
const x = Circuit.if(y.equals(Field.zero)), a, b); // behaves
like `foo ? a : b`
```

## Common methods

```
let x = new Field(4); // x = 4

x = x.add(3); // x = 7

let hash = Poseidon.hash([x]); // takes array of Fields, returns
Field

x = x.sub(1); // x = 6

let privKey = PrivateKey.random(); // create a private key

x = x.mul(3); // x = 18

let pubKey = PublicKey.fromPrivateKey(privKey); // derive public
key

x = x.div(2); // x = 9

let msg = [hash];

x = x.square(); // x = 81

let sig = Signature.create(privKey, msg); // sign a message

x = x.sqrt(); // x = 9

let b = x.equals(8); // b = Bool(false)

b = x.greaterThan(8); // b = Bool(true)

b = b.not().or(b).and(b); // b = Bool(true)

b.toBoolean(); // true
```

```
sig.verify(pubKey, msg); // Bool(true)
```

## Setup

### Dependencies:

NodeJS 16+ (or 14 using node --experimental-wasm-threads)

NPM 6+

Git 2+

IDE (VS Code recommended)

### [zkapp-cli](#)

```
npm install -g zkapp-cli
```

```
zk project my-proj-name
```

```
npm run build && node ./build/src/index.js
```

## Useful links for snarkyJS

[Crowdcast](#)

[Resource Kit](#)