Unlocking On-Chain Privacy: An Intro to ZKP and ZKML

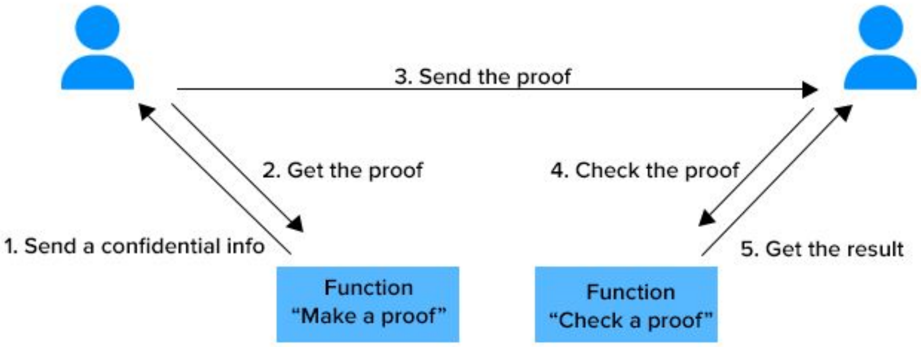
Dr. Cathie So
Privacy & Scaling Explorations Team, Ethereum Foundation

Twitter: @drCathieSo_eth

Quick self intro:

- Background in physics and cognitive science
- Joined blockchain/crypto/web3 in late 2021 (super lateeeeee)
- Self-taught ZKP and founded Zero-Knowledge “University”
- Believe that ZKML is what it takes to bring Web2 devs into Web3

What is (non-interactive) ZKP?



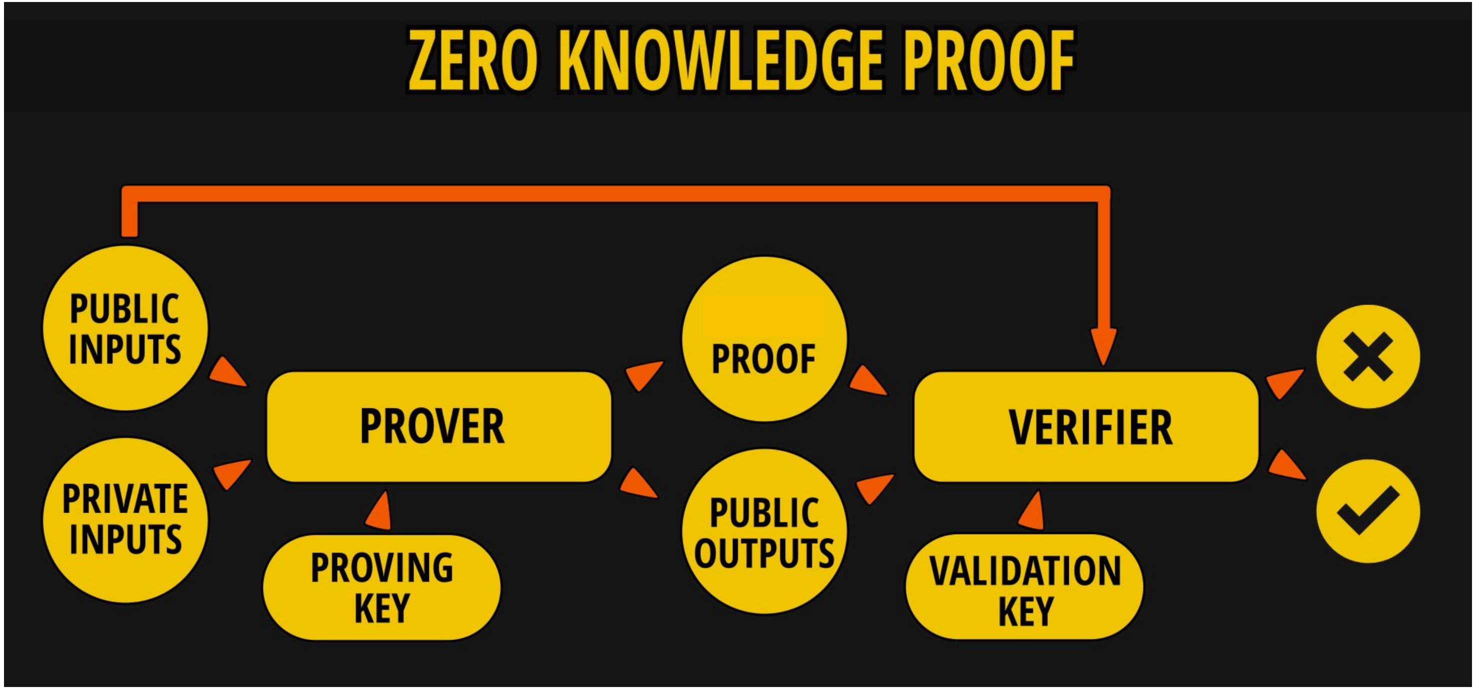
An interaction between two computer programs (or Turing machines) — respectively called a Prover and a Verifier — where the Prover works to convince the Verifier that some mathematical statement is true

Well, with the following properties…

3 (4?) Properties of ZKP

1. Completeness
2. Soundness
3. Zero-knowledge(ness)
4. *Succinctness*

1. If prover is honest, then they will eventually convince verifier.
2. Only if the statement is true can the prover convince the verifier.
3. No information is leaked to the verifier except for the fact that the statement is true.
4. The size of the proof is significantly smaller when compared to the underlying computation



A more complicated version

What does ZKP have to do with Blockchain?

1. Blockchain has limited block size
2. Blockchain transactions are fully transparent

What ZKP can offer as solutions:

1. Compress complicated computations into a succinct proof so that it can be proved on the blockchain
2. Prove ownerships of certain information/data while maintaining privacy

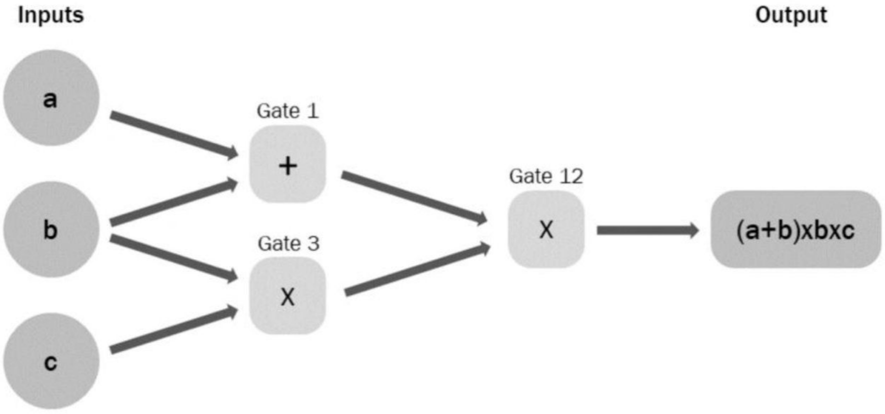
Two approaches to construct ZKP

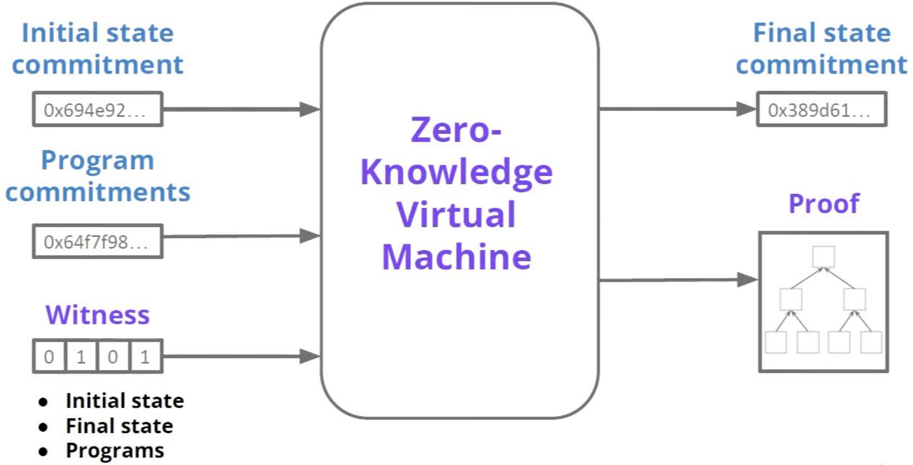
Algebraic circuits

- Analogy: ASIC
- Widely in use
- Disadvantage: you need a custom one for each kind of computation

ZKVM

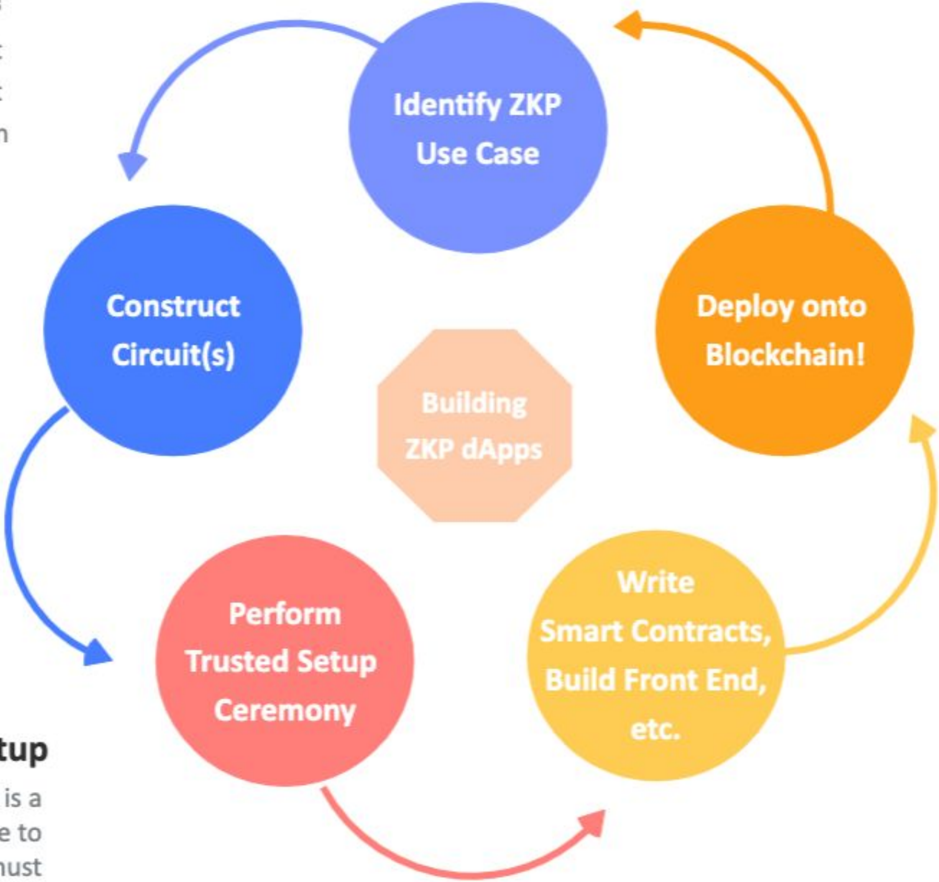
- Analogy: CPU
- Advantage: the computation is an input to the ZKVM, and possibly in a language that you already know





We will focus on circuits in ZK-SNARKs for the rest of the talk

Zero-Knowledge Succinct Non-Interactive Argument of Knowledge, as opposed to another proof architecture, ZK-STARKS



Building ZKP dApps

Identify ZKP Use Case

Construct Circuit(s)

Perform Trusted Setup Ceremony

Write Smart Contracts, Build Front End, etc.

Deploy onto Blockchain!

Do you need ZKP for that?

You will be surprised that a lot of ideas might turn out not needing ZKP as a solution

Learn a new language

Circuits are typically written in specific languages like Circom or LEO, but more recently you can also write them in Rust

Each circuit needs a setup

A trusted setup ceremony is a procedure that is done once to generate a piece of data that must then be used every time some cryptographic protocol is run.

It's a dApp after all!

Build all the other functionalities that are needed to allow users to use your app!

Congratulations!

Get your code audited, and deploy onto a blockchain to share with the whole world!

Lifecycle of launching ZK-SNARKs dApps

A little note on trusted setups and proving schemes...

New proving systems are developed every once in a while,
but here are some common ones you might come across if you are getting started:

- Groth16
  - requires a trusted setup for EVERY CIRCUIT
  - proof size is very small
- PLONK-ish schemes (PLONK, Turbo-PLONK, Halo2, etc.)
  - use specific style of circuit arithmetization
  - requires either a universal setup or no setup (for the case of Halo2) at all!

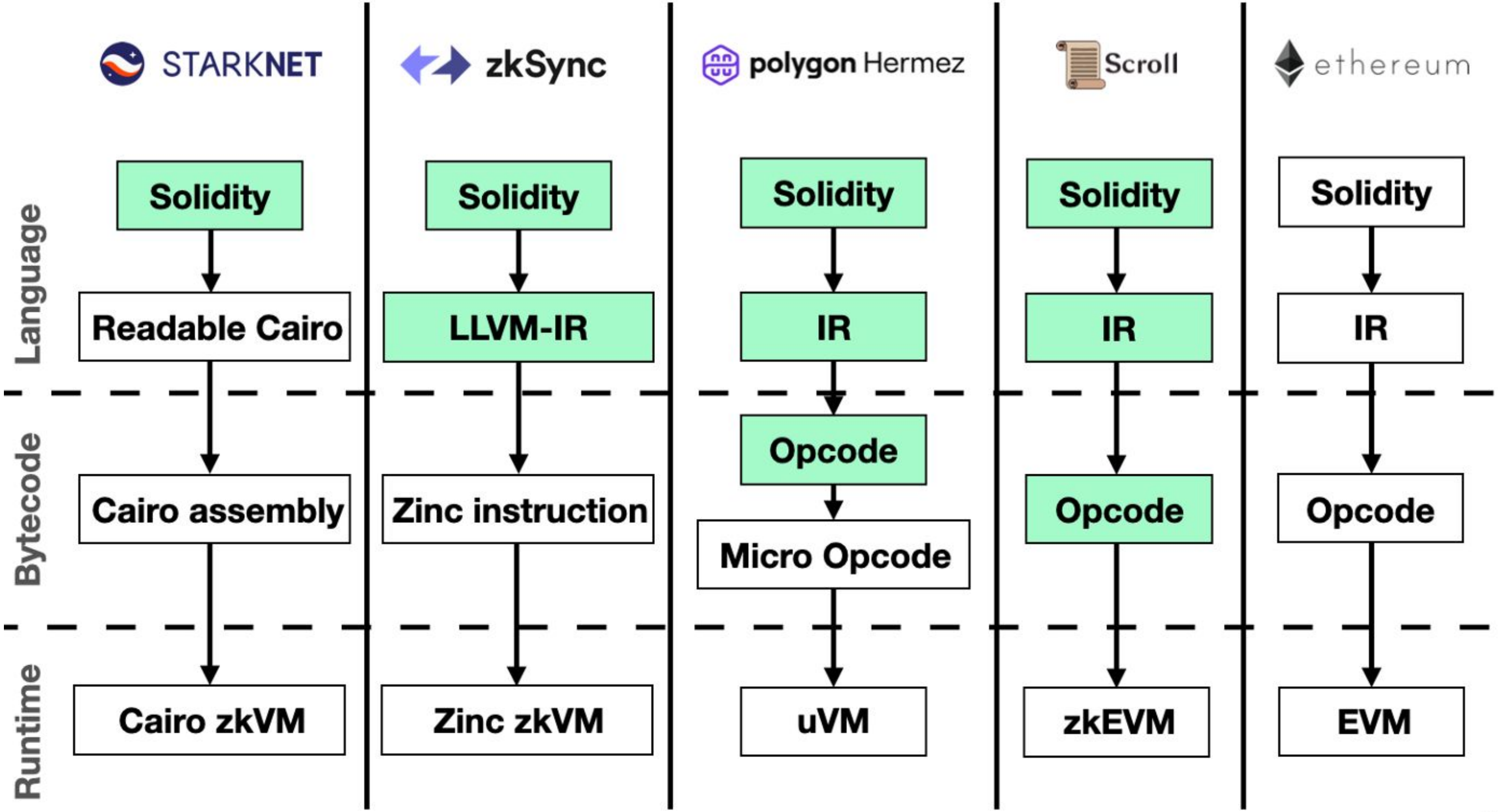
Major area of application #1

Scaling-related, making use of the succinctness property

- Signature Aggregation
  Enable huge multisigs, compressing the on-chain computation size

- ZK-Rollups/zkEVMs
  Layer 2 systems (of ETH) proving that their block generations are valid

Typical components of a zkEVM



Language

Bytecode

Runtime

STARKNET

Solidity

Readable Cairo

Cairo assembly

Cairo zkVM

zkSync

Solidity

LLVM-IR

Zinc instruction

Zinc zkVM

polygon Hermez

Solidity

IR

Opcode

Micro Opcode

uVM

Scroll

Solidity

IR

Opcode

zkEVM

ethereum

Solidity

IR

Opcode

EVM

Major area of application #2

Privacy-related, making use of the zero-knowledge property

- Mixers
  Increase blockchain level of privacy through an anonymity set, where a user hides among a set of k other users.

- DiDs (Decentralized Identifiers)
  Self-sovereign digital identities allowing users to prove their identity without the need of exposing their private information

Examples of Mixer Applications

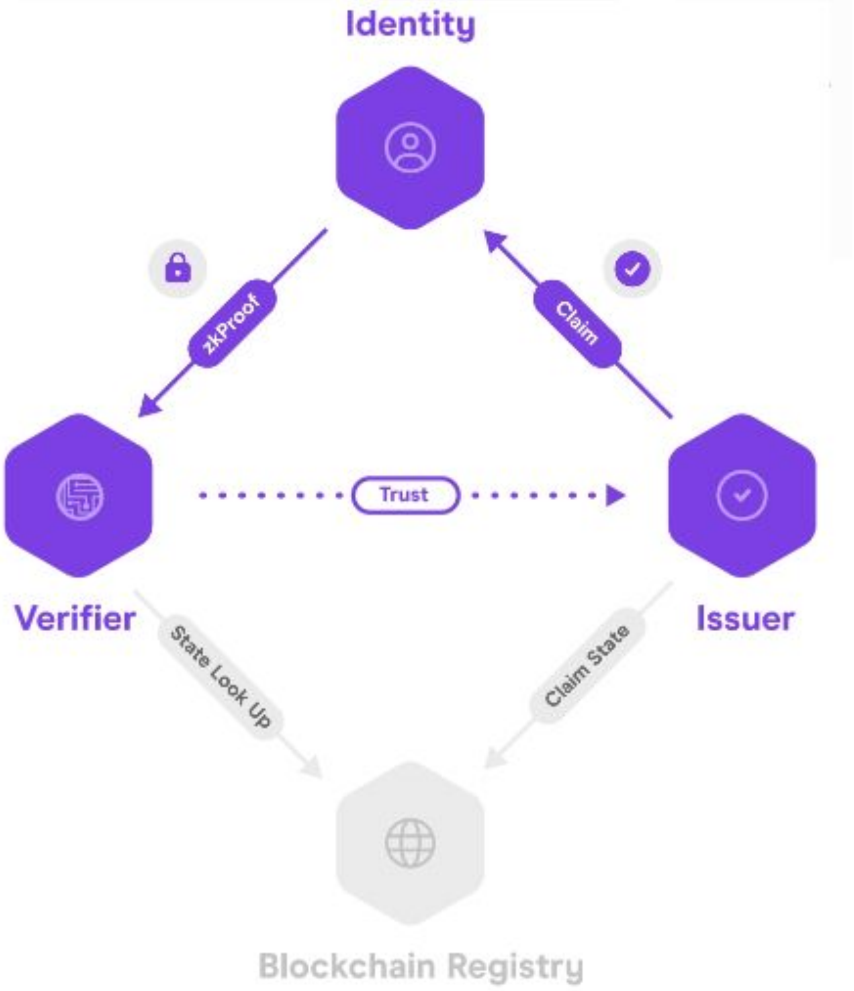
Tornado Cash
‘Mixes potentially identifiable or "tainted" cryptocurrency funds with others, so as to obscure the trail back to the fund's original source’

Semaphore (and its derived applications)
‘Allows you to cast a signal (for example, a vote or endorsement) as a provable group member without revealing your identity’

- Interep
- Zkitter
- Voting

Note: to make the best use out of these applications, the key is to have a large **anonymity set**

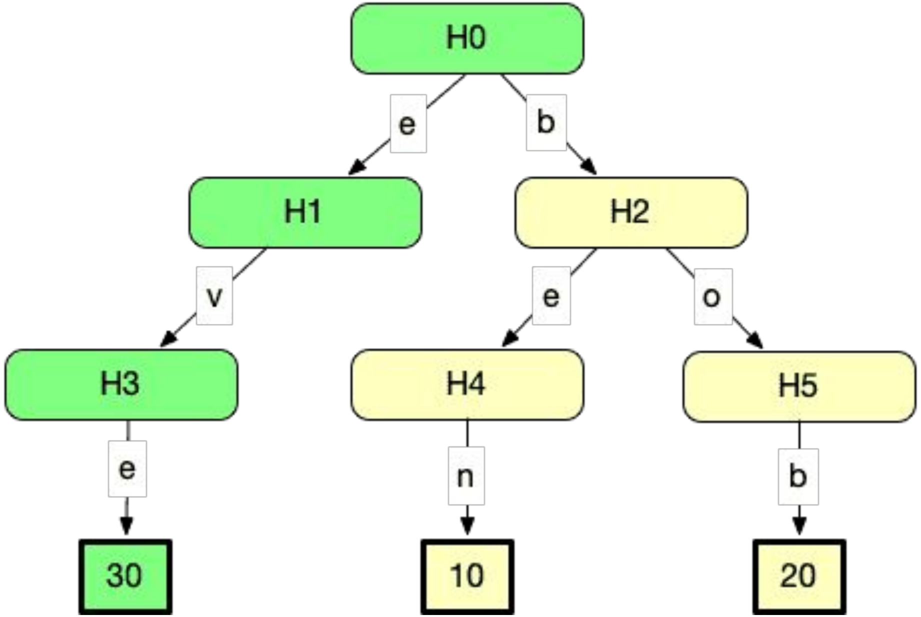
Polygon's zkID



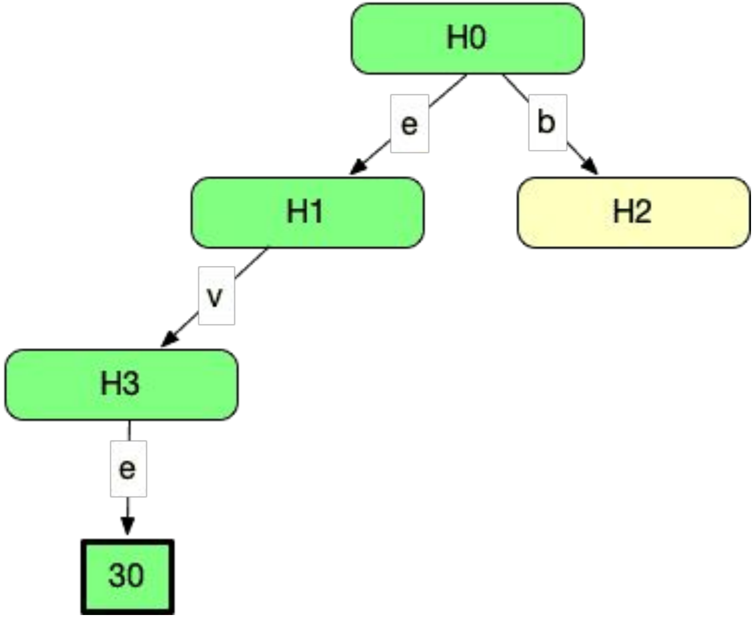
These privacy-related ZKP apps all use the same (type of) circuit: a Merkle proof!

What is a Merkle proof?

Merkle Tree

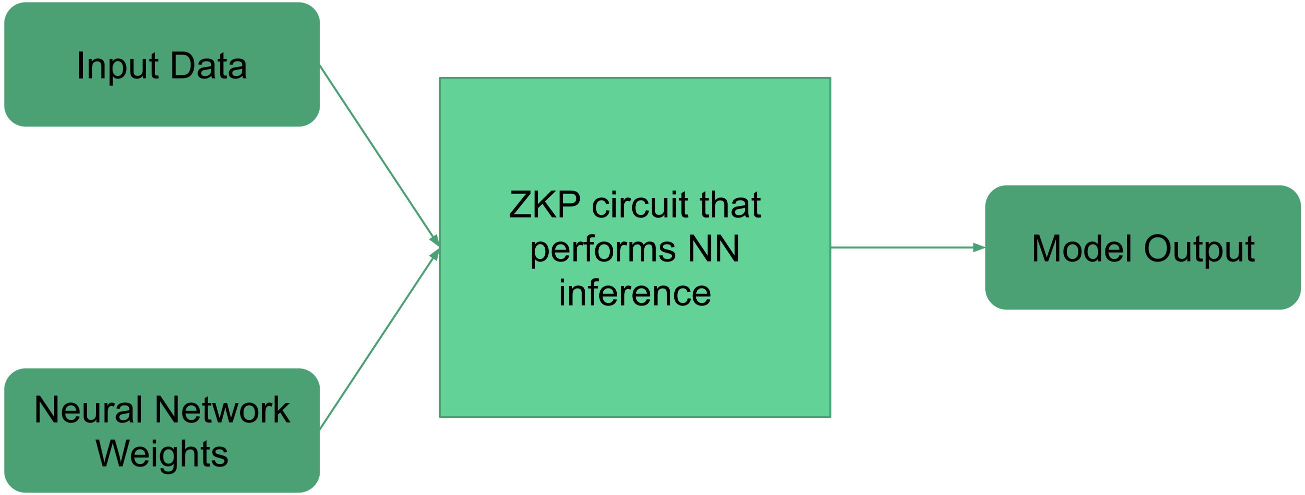


Merkle Proof



Creating custom circuits could open up applications in other space, like ZKML!

Say we have some circuit that performs NN inference...



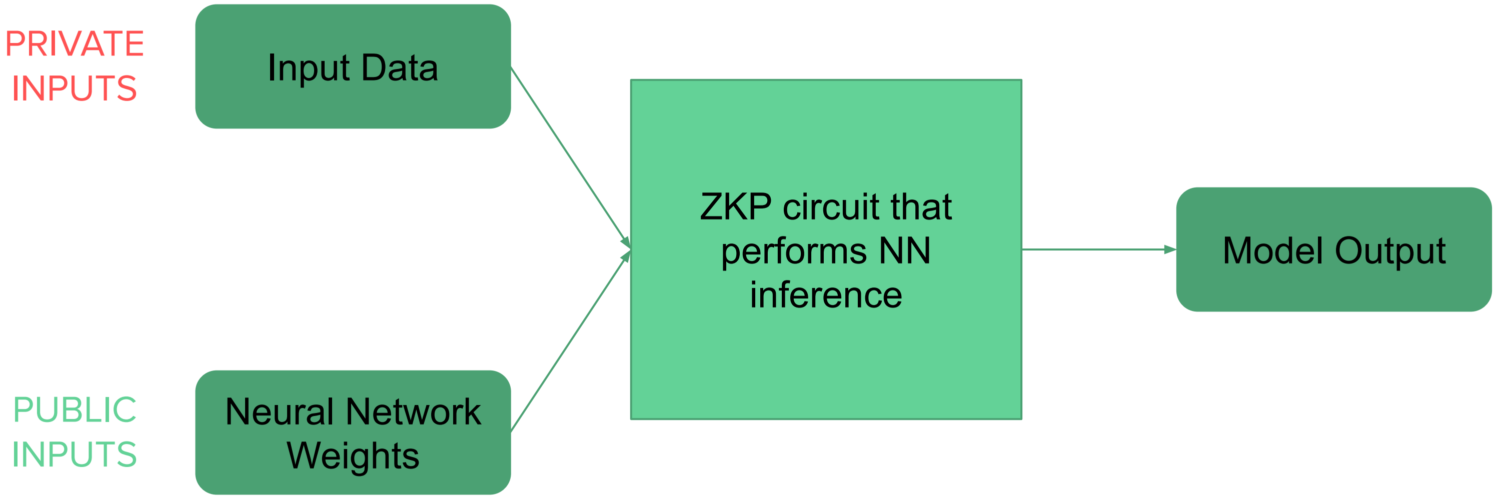
Input Data

Neural Network Weights

ZKP circuit that performs NN inference

Model Output

Use case #1: Private Data, Public Model



PRIVATE INPUTS

Input Data

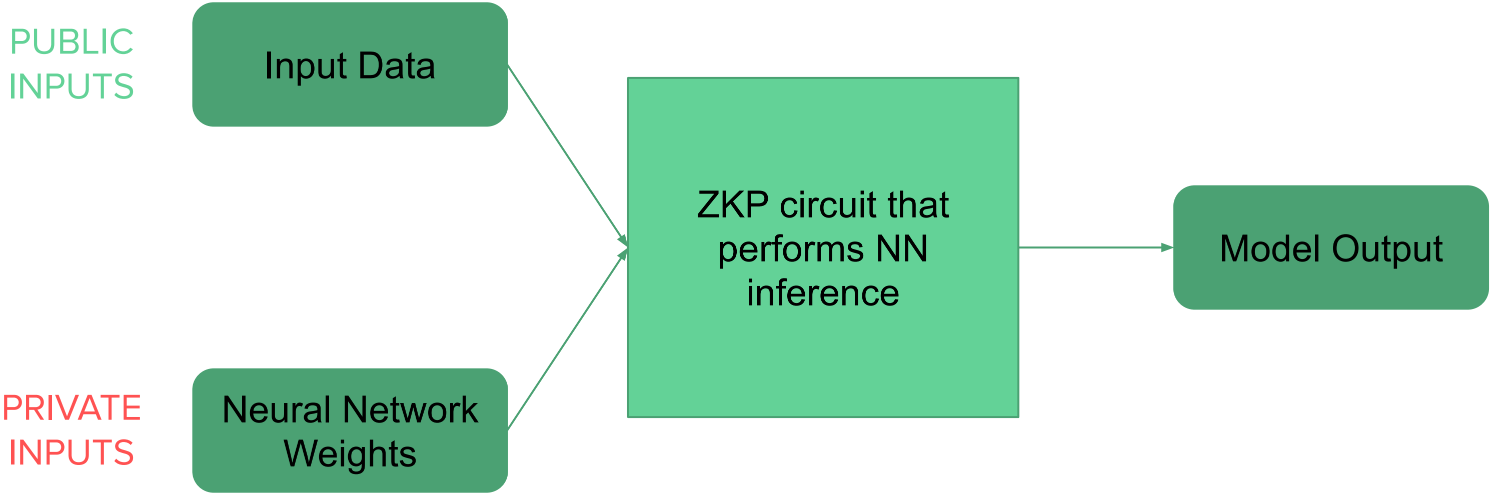
PUBLIC INPUTS

Neural Network Weights

ZKP circuit that performs NN inference

Model Output

Use Case #2: Public Data, Private Model



Why Develop ZKML?

On-Chain:

- On-chain Trading Model
- Biometric Authentication for Smart Wallet
- Verifiable AI Protocol Assistants

Off-Chain:

- Outsourcing Inference Computation
- Verifiable Model Benchmarking
- Proving Model Training Correctness

Transpiling NNs into ZKP circuit

Challenges:

- Fixed-point arithmetic
- Model size

2 years ago
zk-ml/linear-regression-demo by Peiyuan Liao

One year ago
0xZKML/zk-mnist from 0xPARC
Final dense layers as a ZKP circuit

10 months ago
socathie/zkML by me
Full MNIST model as a ZKP circuit

4 months ago
zk-ml/uchikoma - transpiler for non-fp RT
AI art generation minted as NFTs

!! Two weeks ago !!
zkonduit/ezkl update
100M params!!

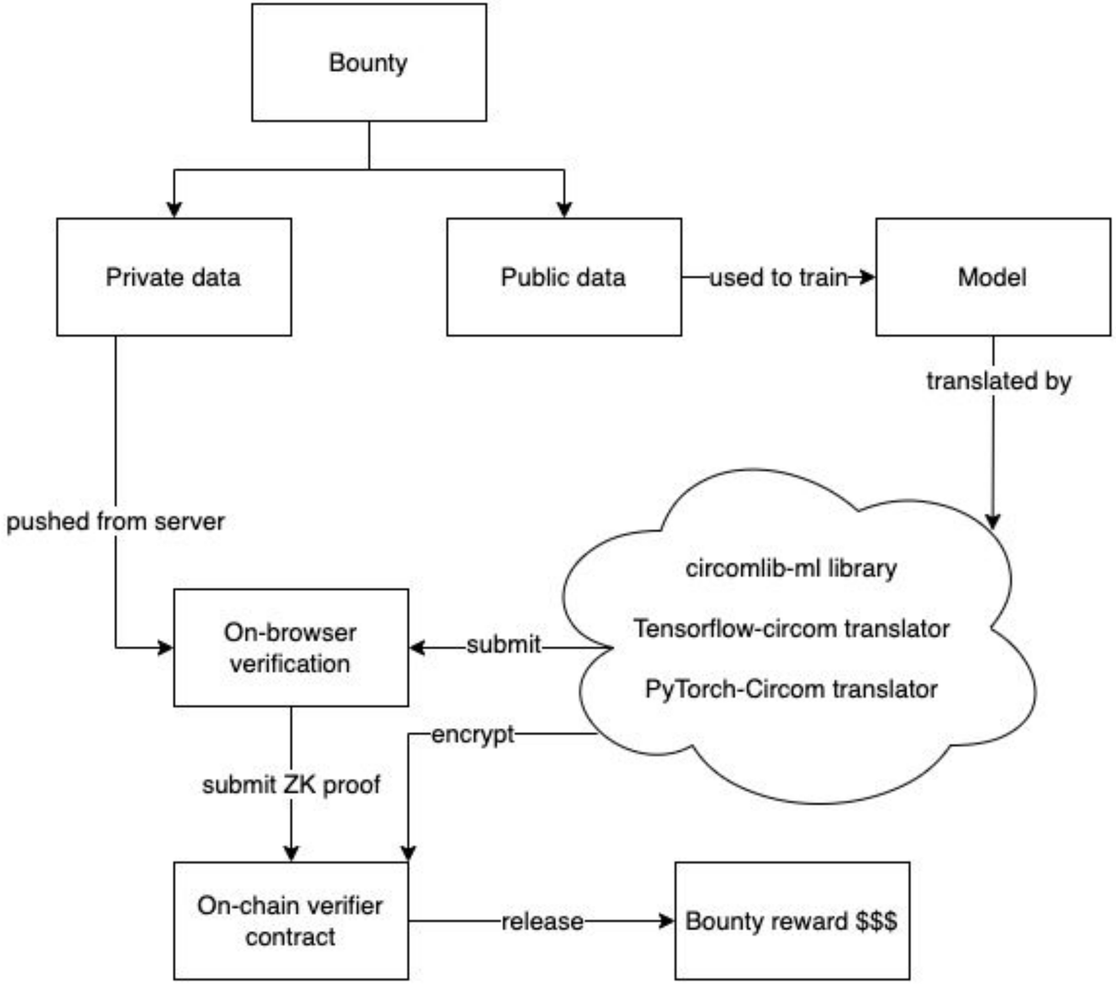
!!! Last week !!!
ddkang/zkml by Daniel Kang
GPT2, Bert, and Diffusion models!!!

!!! Last week !!!
zkp-gravity/0g by me and my team
Won ZK HACK Lisbon with weightless NNs

A ZKML POC

Three major components that form the project:

1. circomlib-ml
   A comprehensive Circom library containing circuits that compute common layers in TensorFlow Keras.
2. keras2circom
   A user-friendly translator that converts ML models in Python into Circom circuits.
3. ZKaggle
   A decentralized bounty platform for hosting, verifying, and paying out bounties, similar to Kaggle, but with the added benefit of privacy preservation.



More detailed writeups:

1. https://hackmd.io/@cathie/zkml
2. https://hackmd.io/@cathie/zkml-research

Tech Stack for ZKPs

Circom

- special-purpose language to write circuits
- options to prove with Groth16 or PLONK and export as a verifier smart contract

Cairo

- general-purpose Turing-complete language used by StarkNet

Rust (Halo2, Plonky2, etc.)

- latest proving schemes
- need to be familiar with PLONKish arithmetization
- where most exciting things happen!!!

How to get started?

A lot of free online resources out there, but just to name a few...

0xPARC
https://0xparc.org

Zero-Knowledge “University”
https://course.zku.one

ZK HACK Whiteboard Sessions
https://zkhack.dev/whiteboard/

Projects mentioned so far:

- zkEVMs:
  - StarkNet and Cairo
    https://starknet.io/docs
  - zkSync
    https://zksync.io
  - Polygon's Hermez
    @0xPolygonHermez
  - Scroll
    https://scroll.io
- Mixers:
  - Tornado Cash
    https://github.com/tornadocash (read only)
  - PSE's Semaphore (and its derived applications)
    https://github.com/privacy-scaling-explorations
- ZKML:
  - WorldCoin
    https://worldcoin.org
  - Linear A *aka* zk-ml *aka* @zkp_ml
    - zk-ml/linear-regression-demo
    - zk-ml/uchikoma
  - 0xZKML/zk-mnist
  - socathie/zkML and socathie/circomlib-ml
  - zkonduit/ezkl
  - ddkang/zkml
  - zkp-gravity/0g
- Others:
  - Polygon's zkID
    @0xPolygonID

POAP