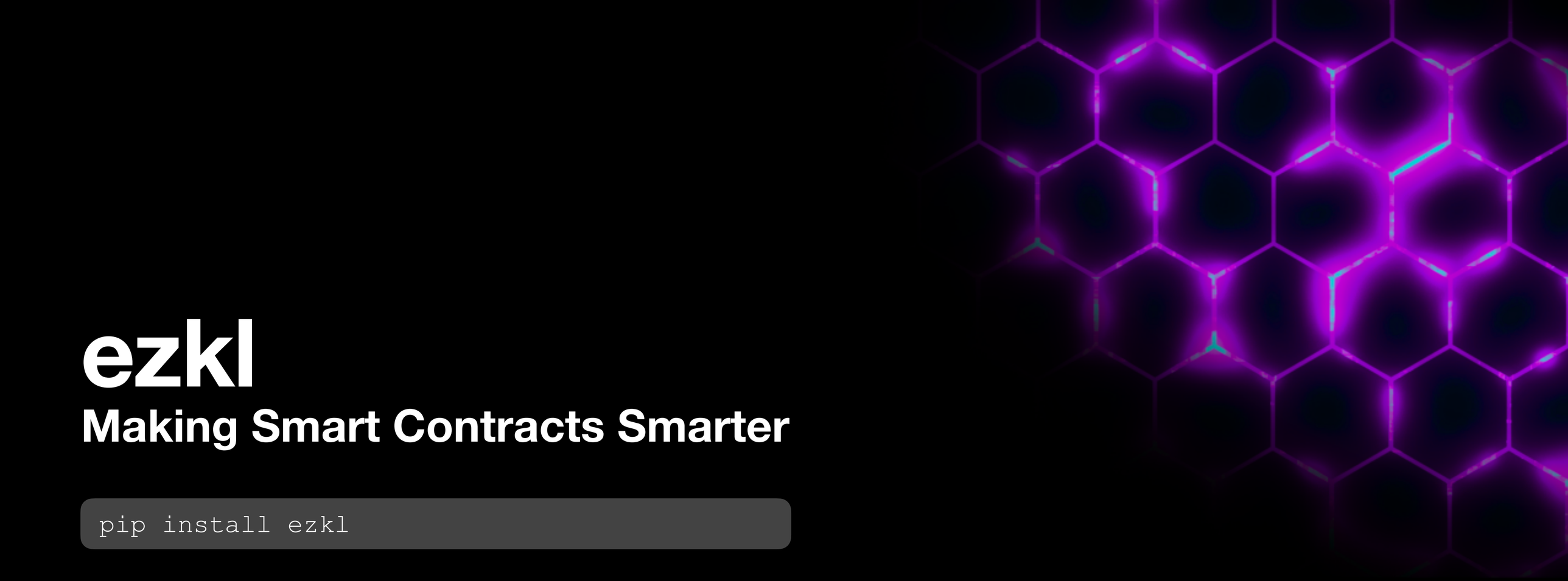


ezkl

Making Smart Contracts Smarter

```
pip install ezkl
```

Jason Morton & Joshia Seam | Zkonduit | ABCDE Aug 12

Problem

~~Smart~~ Contracts

Problem

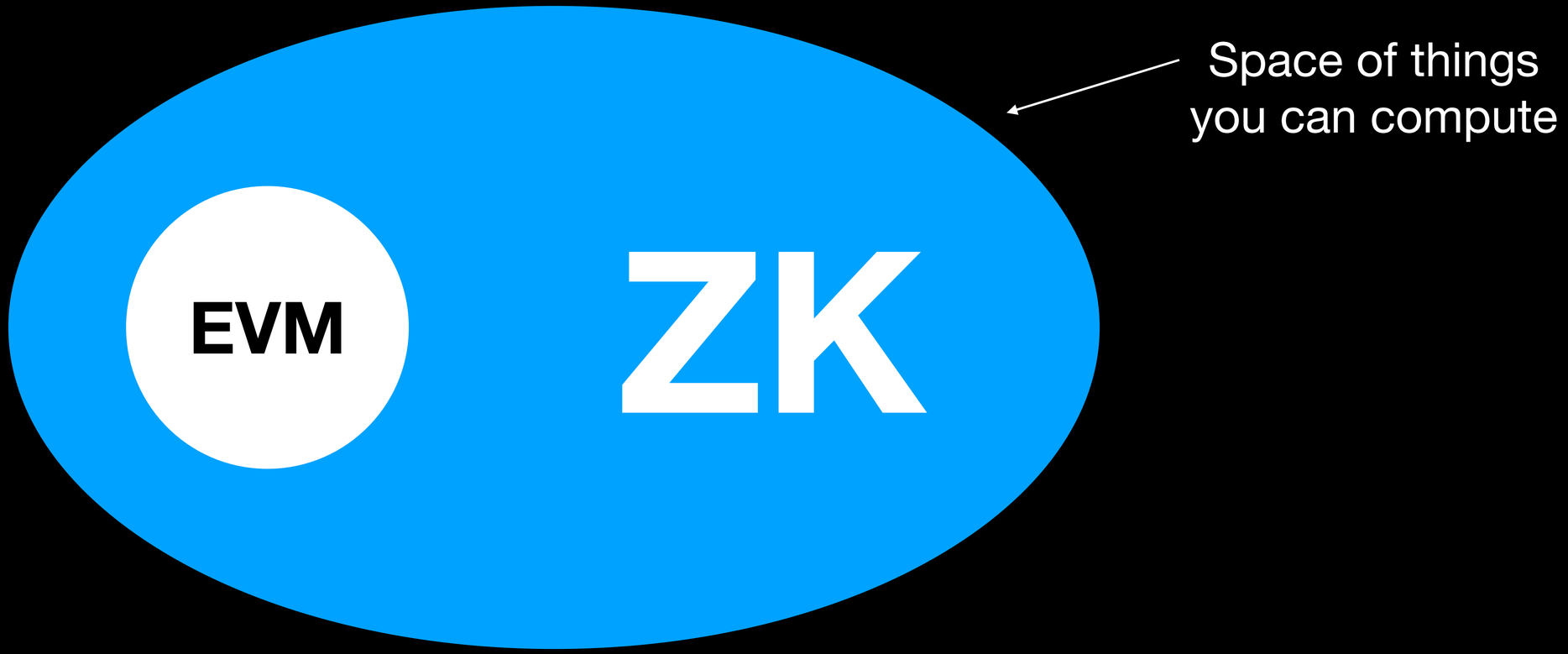
Limited to Elementary/Primary School Maths

EVM is restrictive

Out of Gas Errors
Contract Size Limit Reached
Stack Too Deep

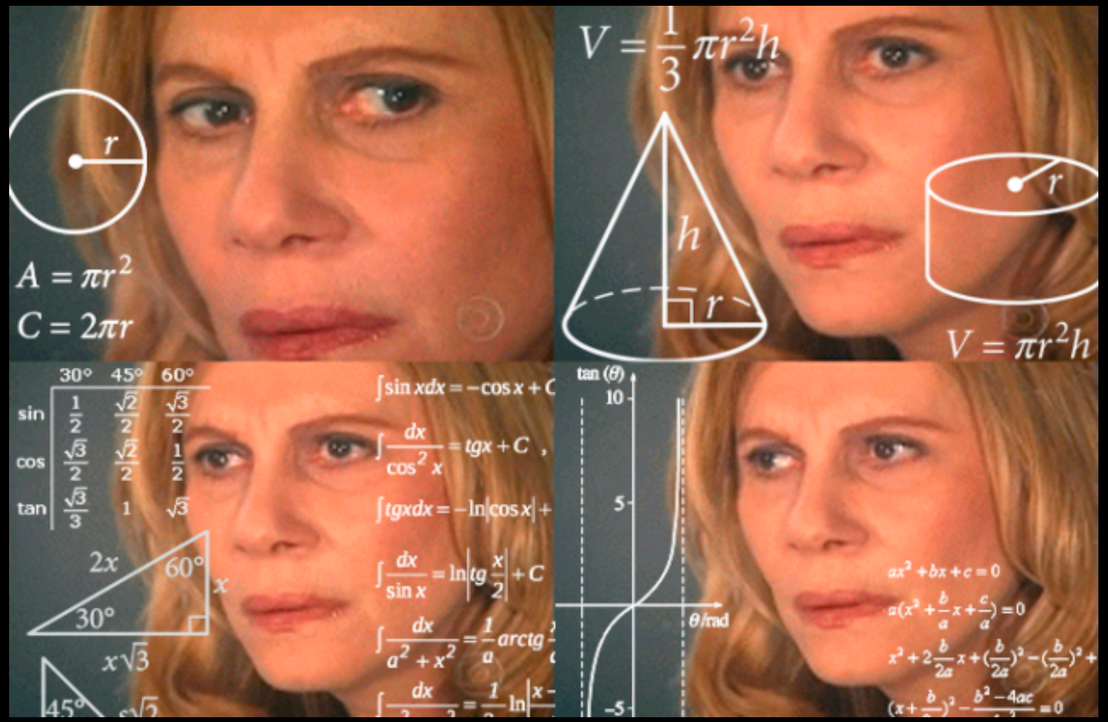
Solution

ZK + Smart Contracts

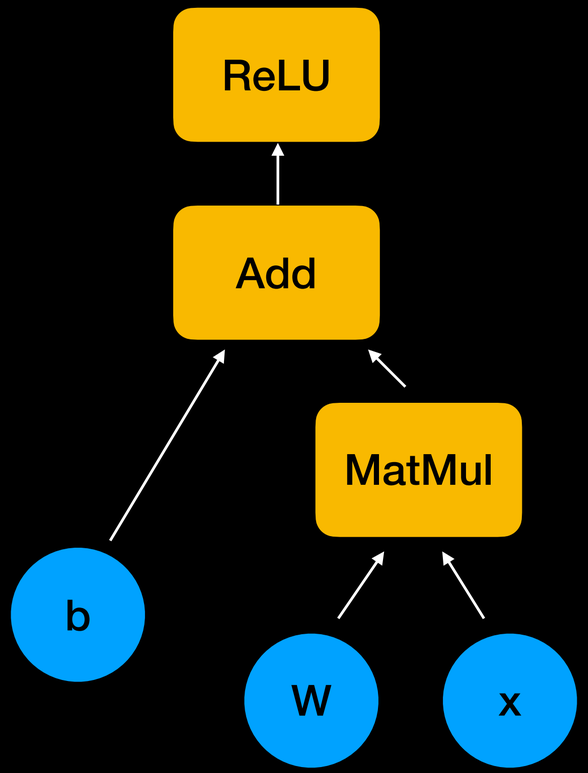


Catch:

Writing ZK circuits is difficult



Insight: neural nets and zk circuits are computational graphs



.onnx from neural network library

ezkl maps

Halo2 circuit

Solution

Train AI/ML models

Don’t write Halo2 circuits

Solution

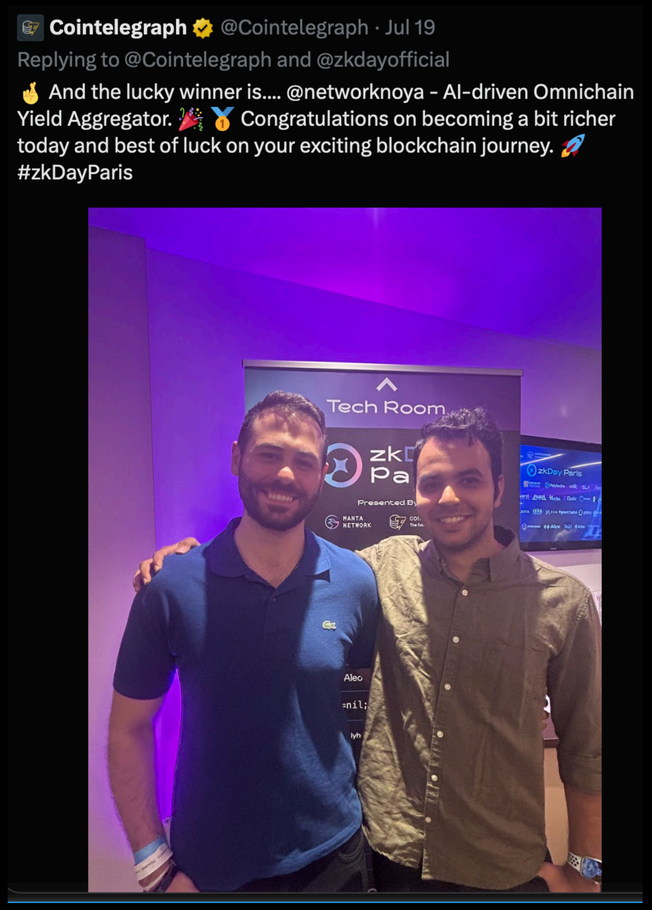
ezkl makes

ZK + Smart Contracts

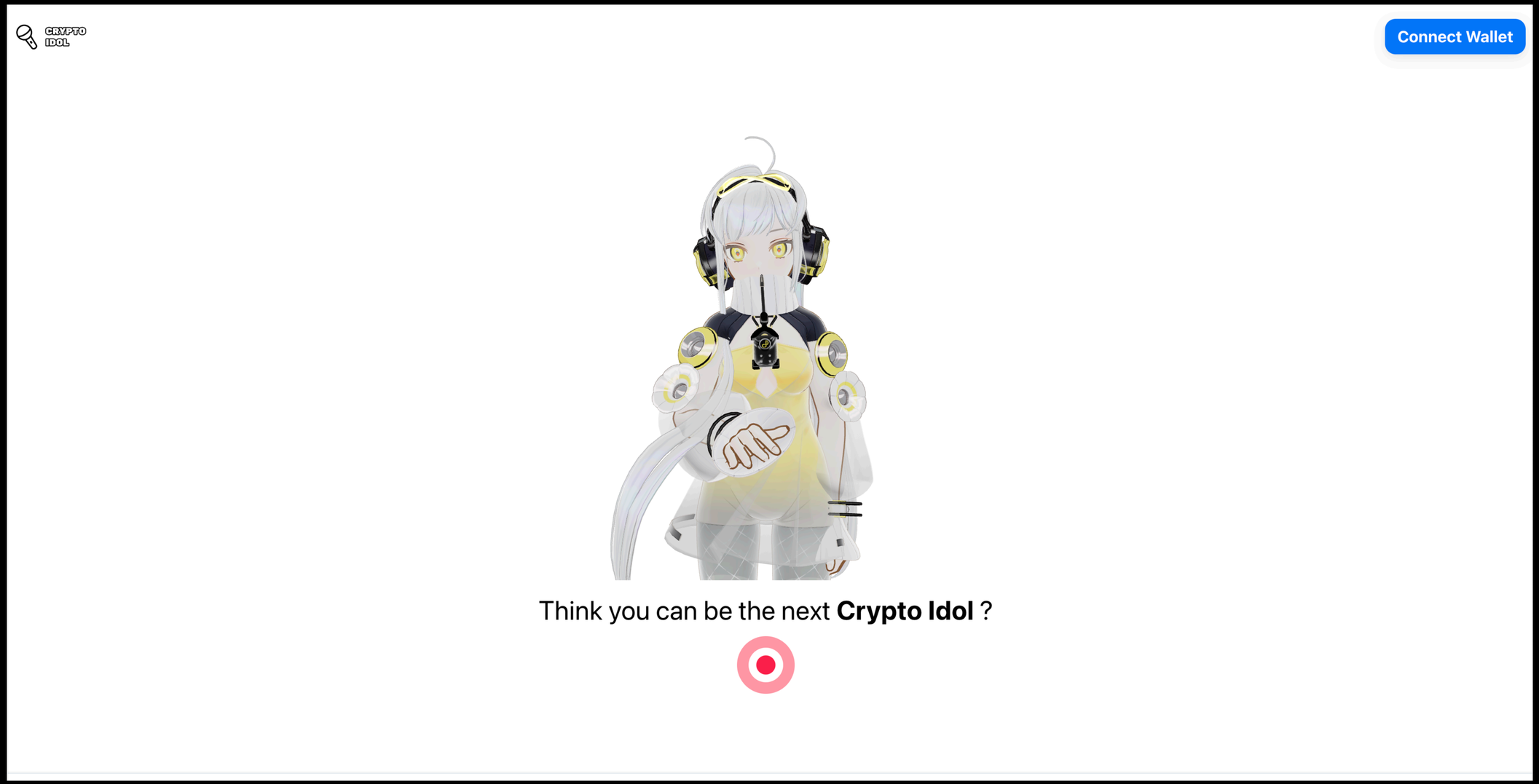
EZ

What can you build now? Some Hackathon Ideas

- Better DeFi vaults
  - Example: noya.ai
- On-chain credit scoring
- Generative NFTs
- Autonomous Worlds
- On-chain games
  - Example: cryptoidol.tech
- Identity/Account Abstraction (Build your own worldcoin)
- Apps that can see and use off-chain data



https://cryptoidol.tech demo



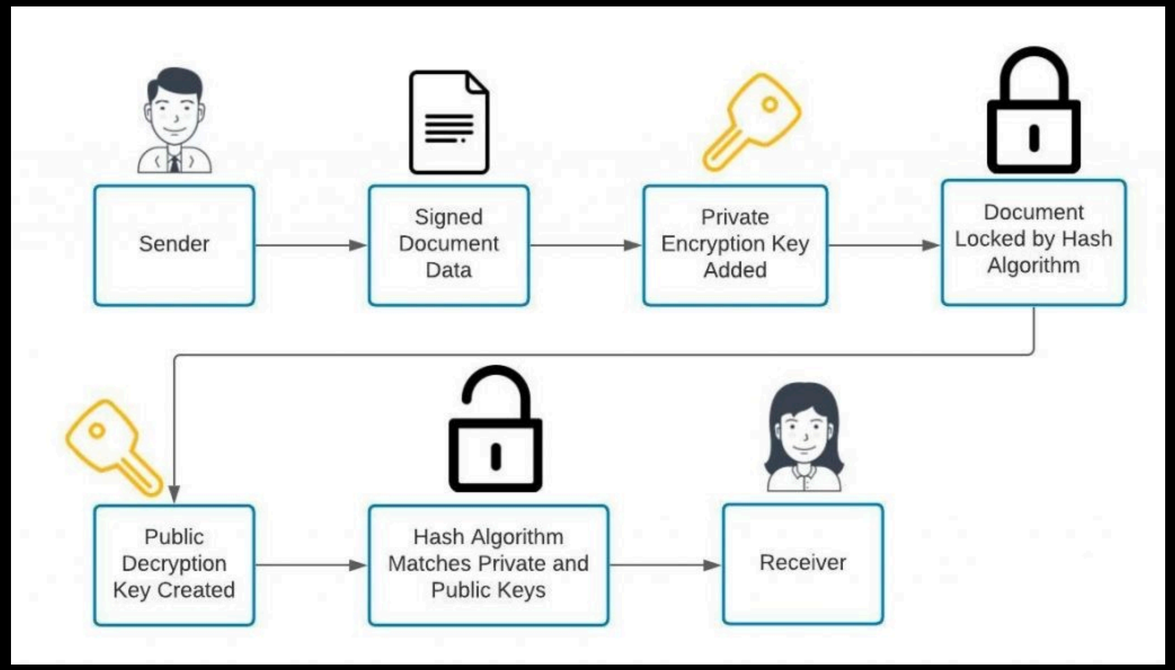
Gas costs are small on L2

Transaction Hash:	0xdf398daaf0f9756a8ddb631b925951351176c8a65b39841a3e8b5c5e2834a223
Status:	Success
Block:	45191493 984033 Block Confirmations
Timestamp:	24 days 21 hrs ago (Jul-17-2023 03:09:58 PM +UTC)
To:	Contract 0xc23c7cad2c36c689613a234892c158d645ef88cb
Value:	0 MATIC ($0.00)
Transaction Fee:	0.054294749305855905 MATIC ($0.04)
Gas Price:	0.000000099898526595 MATIC (99.898526595 Gwei)
MATIC Price:	$0.78 / MATIC

Diving Deeper into ezkl concepts

How we use Digital Signatures Today

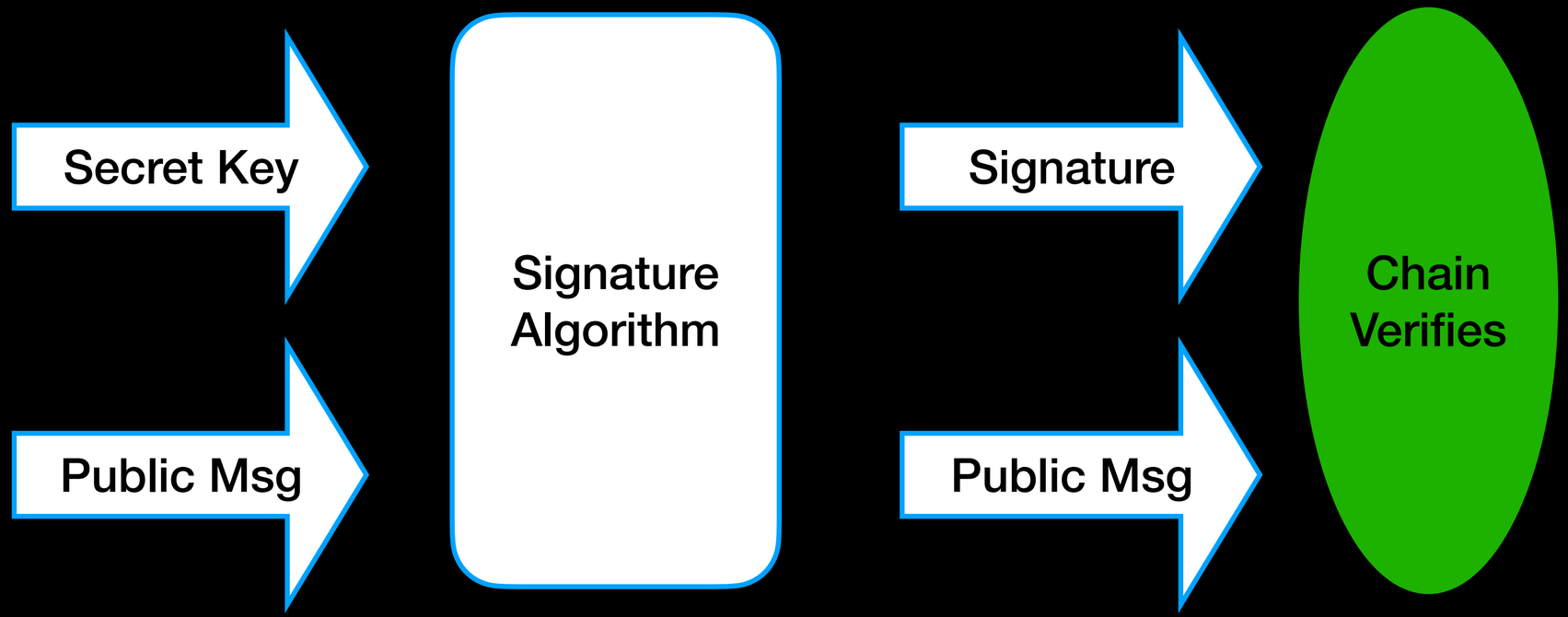
- Program **with** them
- Complex Constructions
- Chatty Protocols
- Limited Security Models and Privacy Options



What if we could program in Digital Signatures

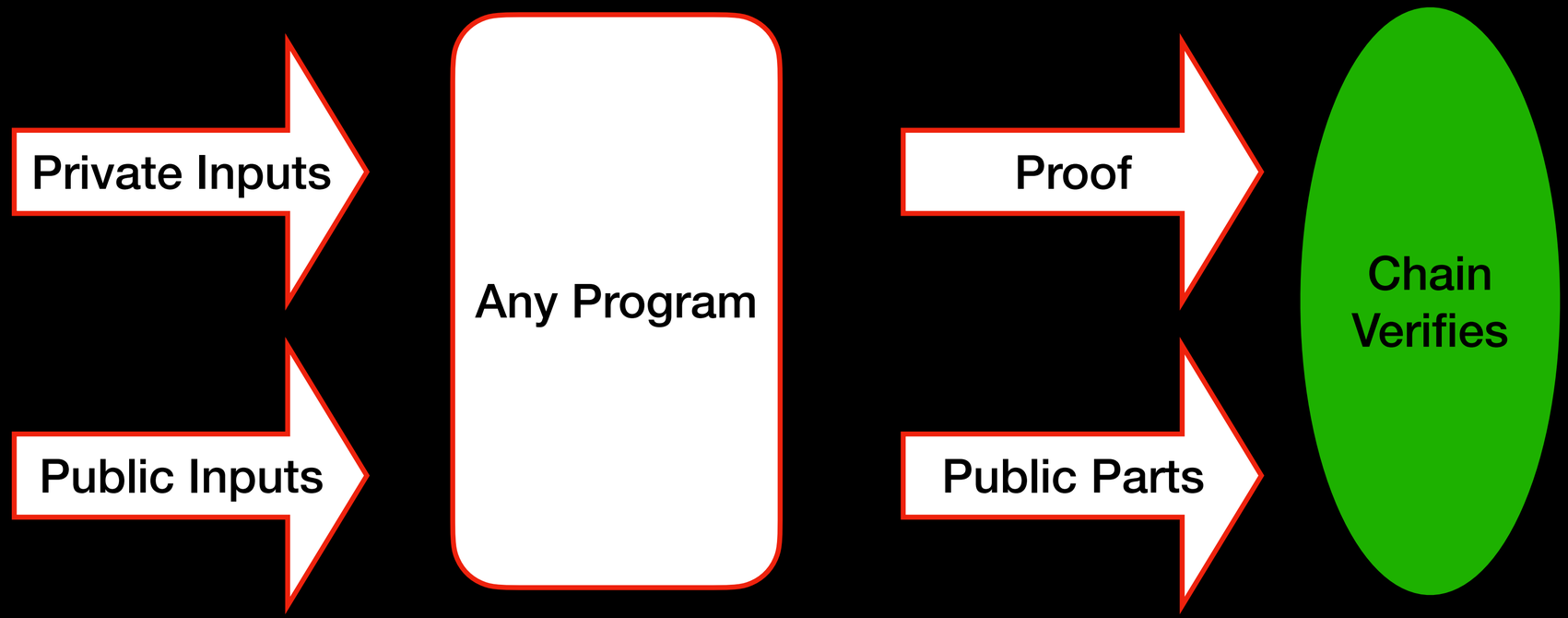
ZKPs are “programmable signatures”

Any program, any input you like with similar security and privacy



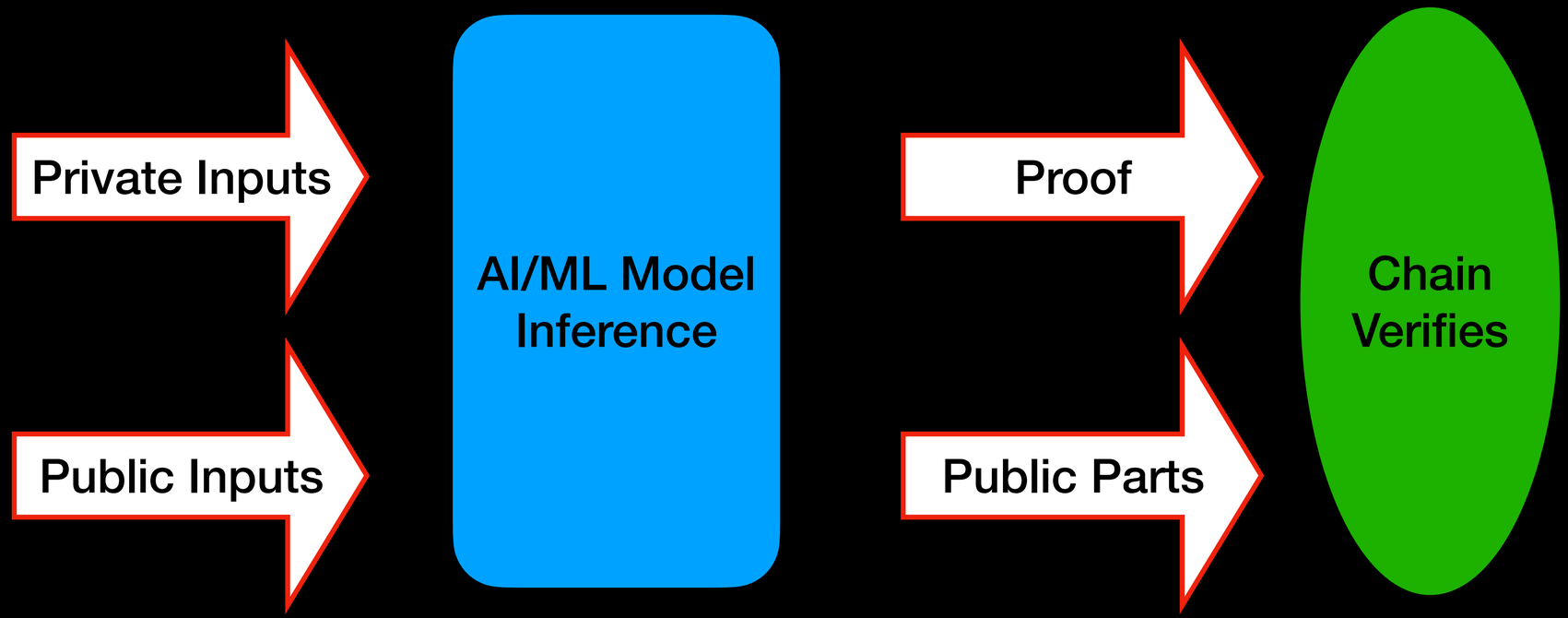
ZKPs are “programmable signatures”

Any program, any input you like with similar security and privacy



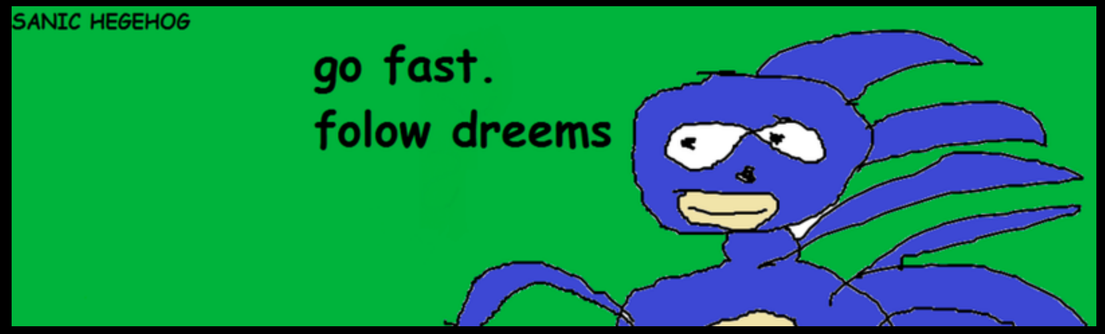
ZKML (Zero Knowledge Machine Learning)

Mostly Inference For Now

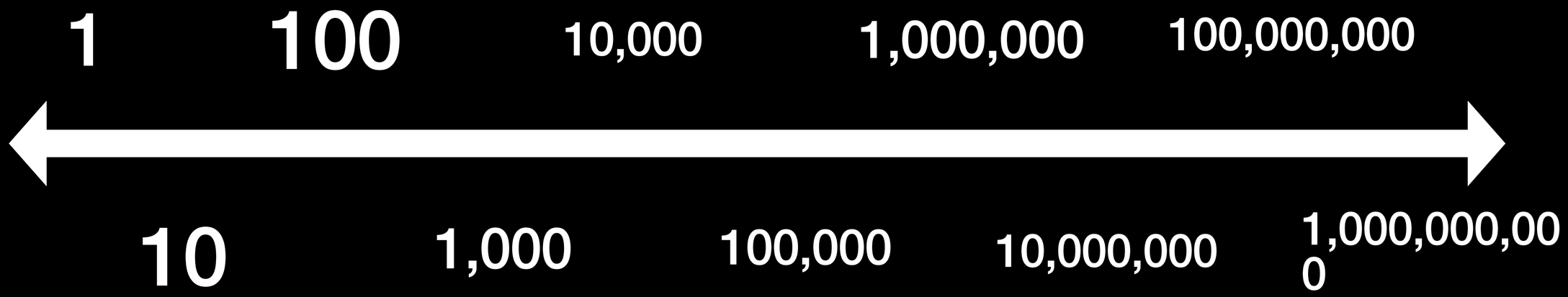


Why build w/ZKML now?

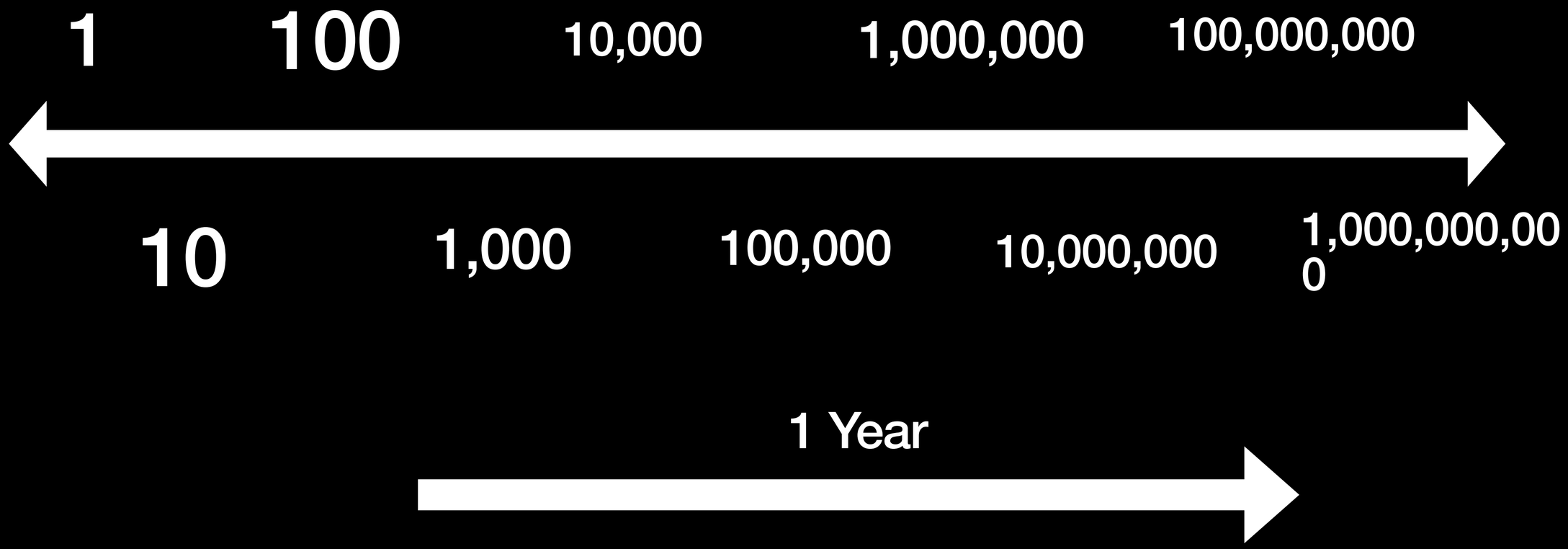
ZKPs are getting
easier, faster, and practical



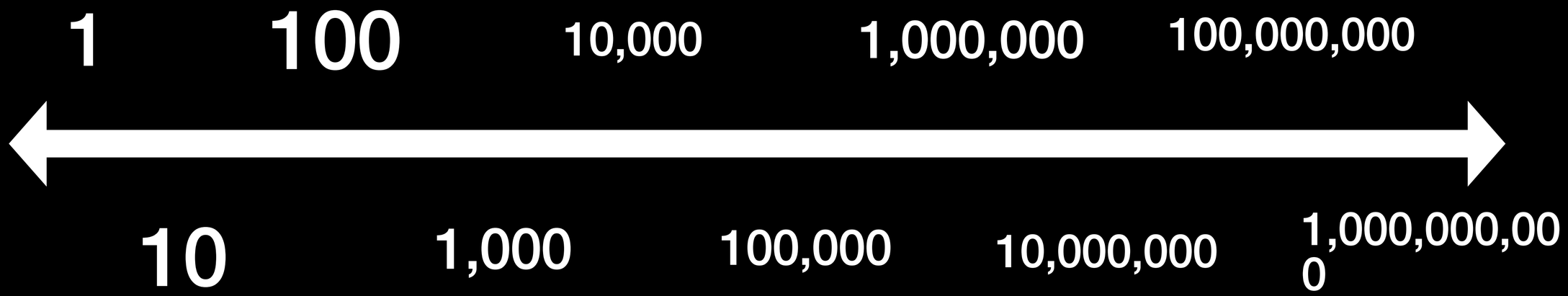
Floating-point operations per proof in ezkl



Flops (add, multiply) per proof in ezkl

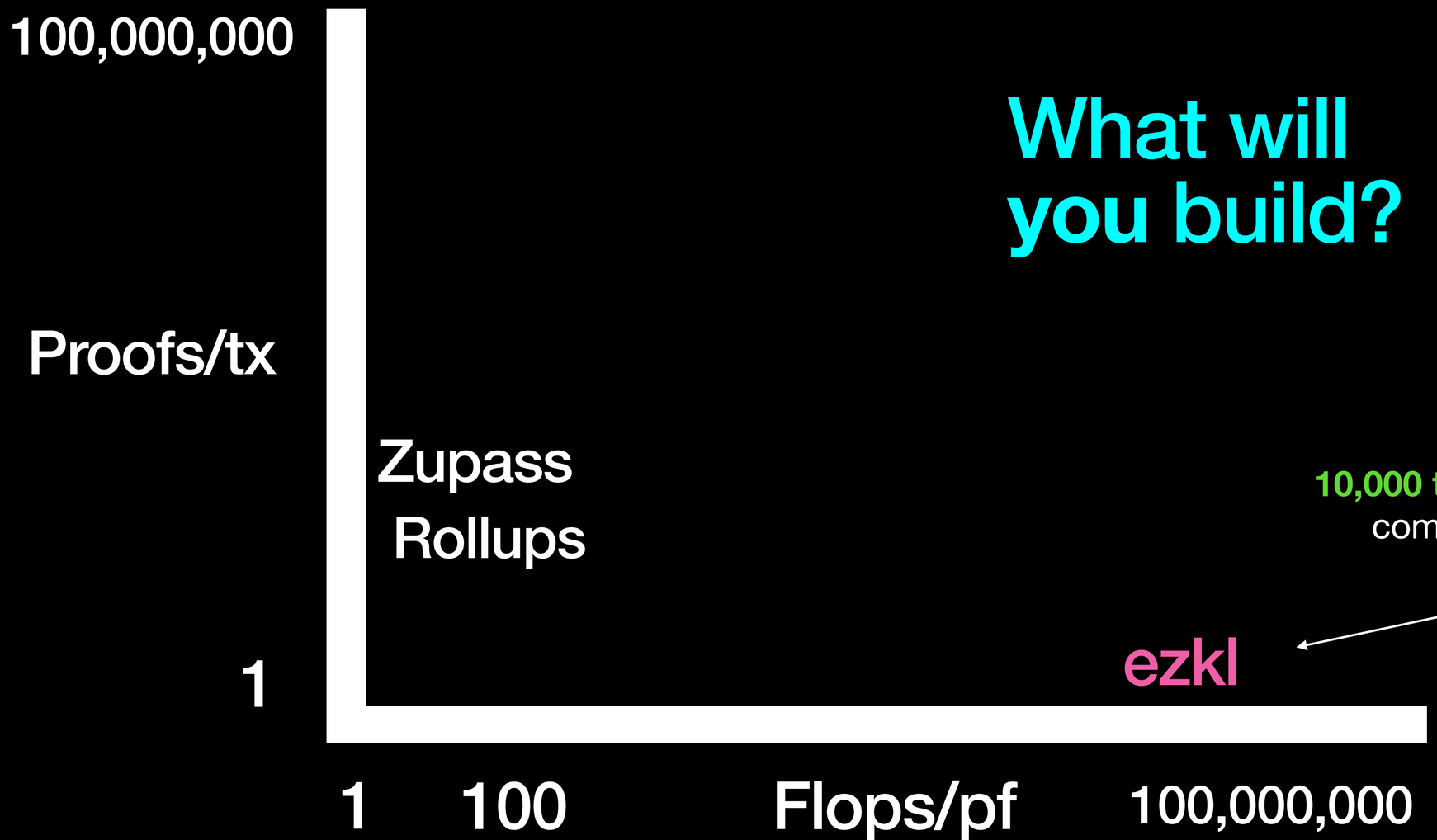


Proofs per chain transaction



Proofs per chain transaction

Compute per chain transaction



What will you build?

10,000 times faster today (12 Aug), compared to last September

Privacy Protocols

Rollups

Hermez

zkSync

Aztec

STARKWARE

ZK of Today

Email?

Oracles?

Games?

Autonomous Worlds?

AI DAOs?

Onchain Waifus?

Onchain physics simulators?

?????

ZK of Tomorrow

How can I build on ezkl?

EZKL engine

- Want to run some stats or an AI model on-chain but it doesn't fit (or you want the model or inputs to be secret)
  - Say a Python function result = forward(input)
- ezkl turns forward into
  - A prover that takes input and gives you (input, result, <hex>)
  - A smart contract that checks <hex> to determine if it is true that result = forward(input)
  - This lets you do arbitrary computation "on chain"

- As app developer, define your forward function in Python

```python
class MyModel(nn.Module):
    def __init__(self):
        super(MyModel, self).__init__()

        self.conv1 = nn.Conv2d(in_channels=1, out_channels=2, kernel_size=5, stride=2)
        self.conv2 = nn.Conv2d(in_channels=2, out_channels=3, kernel_size=5, stride=2)

        self.relu = nn.ReLU()

        self.d1 = nn.Linear(48, 48)
        self.d2 = nn.Linear(48, 10)

    def forward(self, x):
        # 32x1x28x28 => 32x32x26x26
        x = self.conv1(x)
        x = self.relu(x)
        x = self.conv2(x)
        x = self.relu(x)

        # flatten => 32 x (32*26*26)
        x = x.flatten(start_dim = 1)

        # 32 x (32*26*26) => 32x128
        x = self.d1(x)
        x = self.relu(x)

        # logits => 32x10
        logits = self.d2(x)

        return logits
```

Questions about result = forward(input)

- How much gas to verify?
  - about 400k
- How fast to prove?
  - Varies; stats and small ML takes seconds.
- Can parts be kept secret? Yes:
  - Prover's <hex> shows it knows input and/or forward
    - such that result = forward(input)
    - optionally without revealing input and/or forward to anyone

Export model to onnx (Boilerplate for Torch)

```
# Flips the neural net into inference mode
circuit.eval()

    # Export the model
torch.onnx.export(circuit,               # model being run
                  x,                     # model input (or a tuple for multiple inputs)
                  model_path,            # where to save the model (can be a file or file-like object)
                  export_params=True,        # store the trained parameter weights inside the model file
                  opset_version=10,          # the ONNX version to export the model to
                  do_constant_folding=True,  # whether to execute constant folding for optimization
                  input_names = ['input'],   # the model's input names
                  output_names = ['output'], # the model's output names
                  dynamic_axes={'input' : {0 : 'batch_size'},    # variable length axes
                                'output' : {0 : 'batch_size'}})

data_array = ((x).detach().numpy()).reshape([-1]).tolist()

data = dict(input_data = [data_array])

    # Serialize data into file:
json.dump( data, open(data_path, 'w' ))
```

Export model to onnx (Boilerplate for Keras)

```
spec = tf.TensorSpec([1, 28, 28, 1], tf.float32, name='input_0')


tf2onnx.convert.from_keras(model, input_signature=[spec], inputs_as_nchw=['input_0'], opset=12, output_path=model_path)

data_array = x.reshape([-1]).tolist()

data = dict(input_data = [data_array])

    # Serialize data into file:
json.dump( data, open(data_path, 'w' ))
```

Questions about result = forward(input)

- Can the secret parts be committed to, attested, or signed?
  - Yes, prover can prove it knows input and/or forward such that result = forward(input) and
  - that they hash to something it reveals and/or signs, or someone else signed
- Can you run the proof for me on a server somewhere?
  - Sure, happy to

Questions about result = forward(input)

- Can an input be:
  - User-uploaded? Yes
  - A database query? Yes
  - Current on-chain state? Yes
  - Historical on-chain state? Soon

Now make a setup

- Generate some artifacts that can be proved against, including
  - Ingredients the prover needs
  - Solidity verifier (maybe deploy it)
- Tell your client / provers where to find them
- Wire your smart contract into the verifier contract
  - check proof is true, then change state

Boilerplate 2: settings, compile, gen verifier

```
res = ezkl.gen_settings(model_path, settings_path, py_run_args=run_args)
assert res == True

res = await ezkl.calibrate_settings(val_data, model_path, settings_path, "resources")
assert res == True
print("verified")
```

```
res = ezkl.setup(
        model_path,
        vk_path,
        pk_path,
        srs_path,
        settings_path,
    )
```

```
res = ezkl.create_evm_verifier(
        vk_path,
        srs_path,
        settings_path,
        sol_code_path,
        abi_path,
    )
```

Boilerplate 2: settings, compile, gen verifier

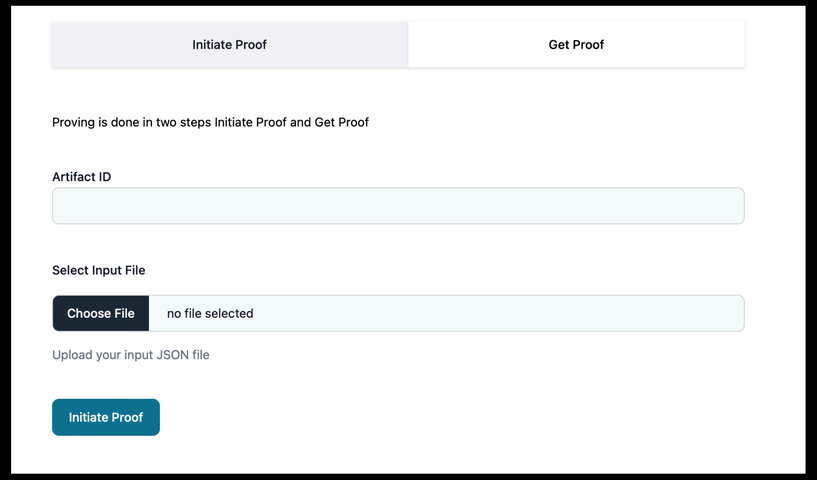
```
res = ezkl.gen_settings(model_path, settings_path, py_run_args=run_args)
assert res == True

res = await ezkl.calibrate_settings(val_data, model_path, settings_path, "resources")
assert res == True
print("verified")
```

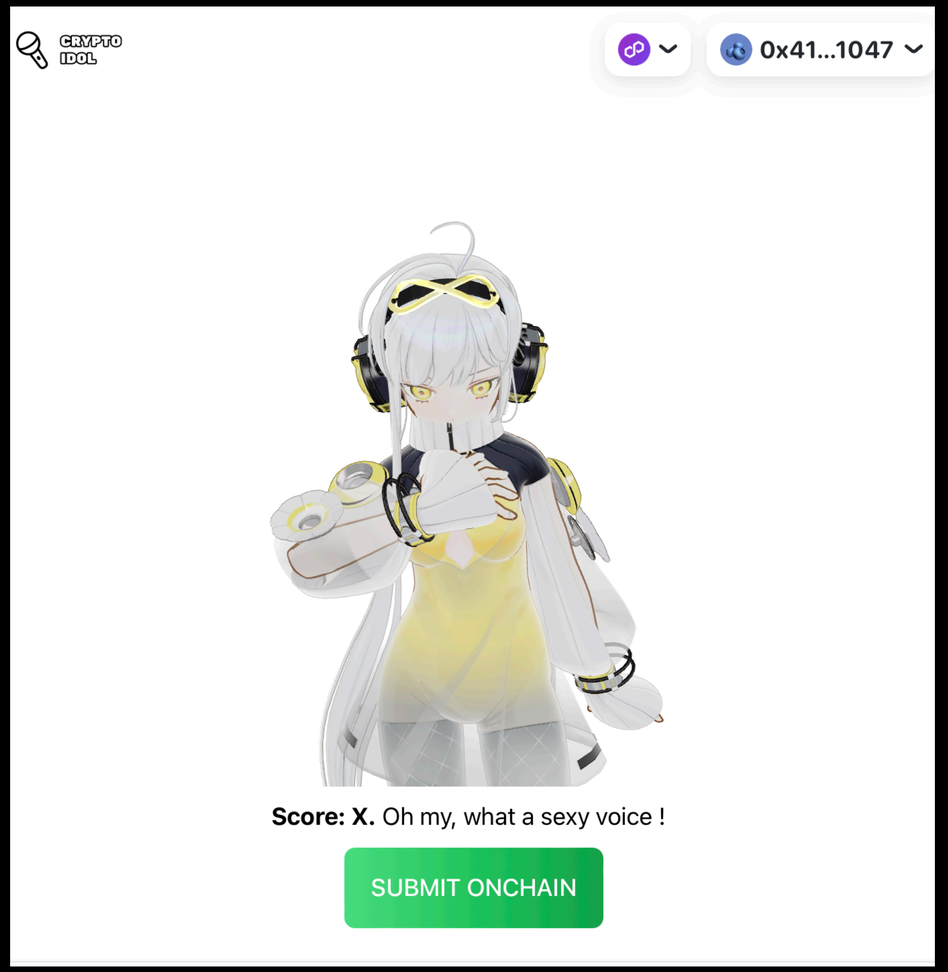
```
res = ezkl.setup(
        model_path,
        vk_path,
        pk_path,
        srs_path,
        settings_path,
    )
```

Deploy this, call from your contract

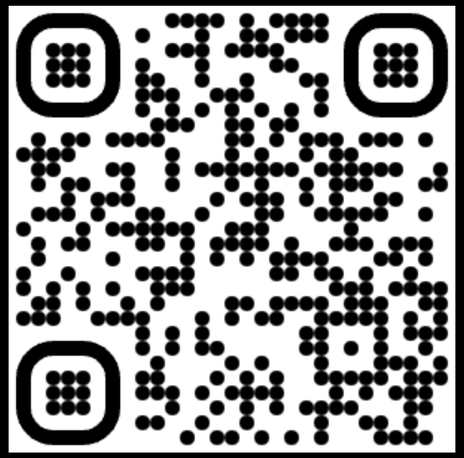
```
res = ezkl.create_evm_verifier(
        vk_path,
        srs_path,
        settings_path,
        sol_code_path,
        abi_path,
    )
```

Using ezkljs, prove from your app





For More Examples Checkout ezkl/examples on Github



GitHub

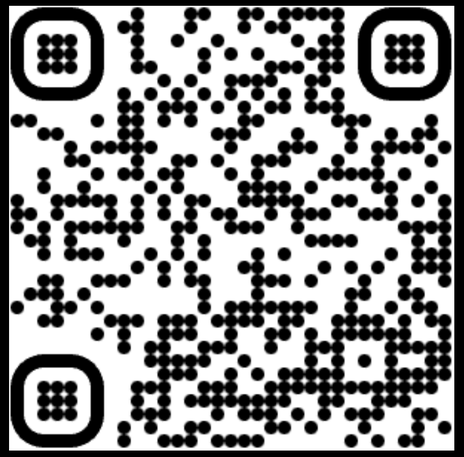
https://github.com/zkonduit/ezkl

To ask questions join our Discord or Telegram

Discord
https://discord.gg/HrgSTAy2AS

Telegram
https://t.me/+QRzaRvTPIthlYWMx

ezkl hub waitlist



Typeform

https://mmycoj5vy74.typeform.com/to/Z2aikKUt