Lesson 20

Week 5

Lesson 17 - STARK implementation

Lesson 18 - Plonk part 1 / Linea

Lesson 19 - Plonk part 2 / Boojum

Lesson 20 - ZKML

Today's topics

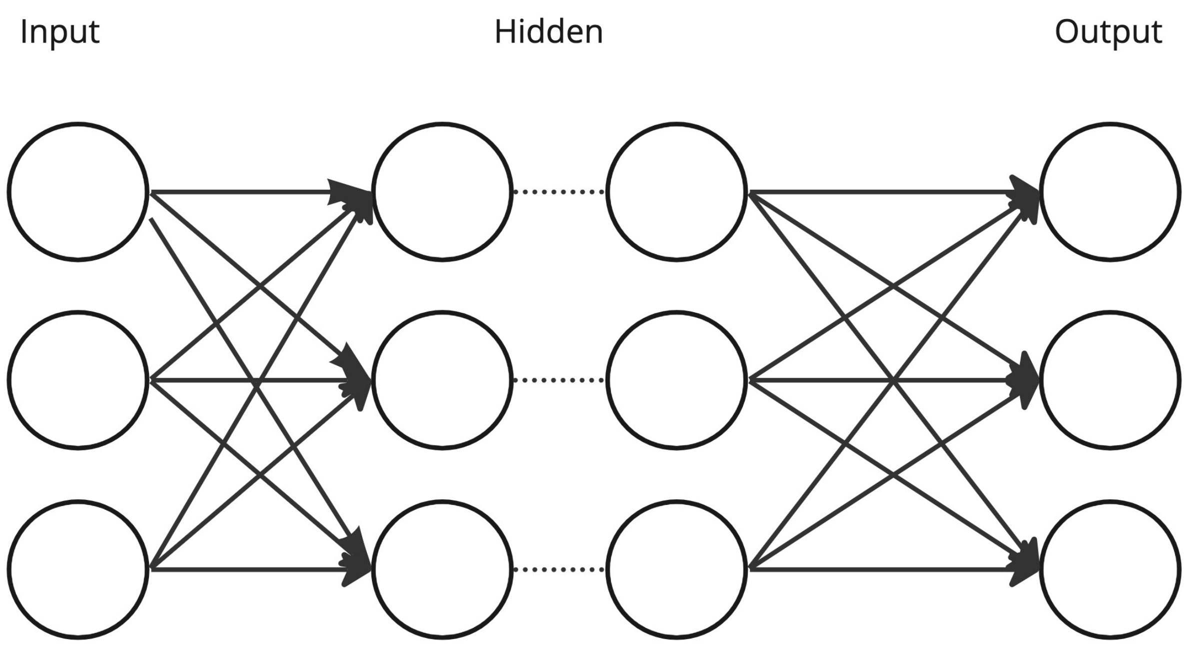
- Machine Learning background
- zkML Introduction
- Ecosystem
- Approaches to zkML
- Projects
  - EZKL
  - Giza
  - ZKaggle
- Hardware
  - Supranational
  - Ingonyama
- Resources

Machine Learning background

Machine learning, a branch of artificial intelligence, focuses on creating and utilizing algorithms that allow computers to independently learn and adapt from data. Through iterative processes, this leads to the optimization of performance. Large-scale language models such as GPT-4 and Bard epitomize the latest advances in natural language processing, using extensive training data to craft text that closely resembles human language.

In a neural network, each node functions as a linear regression model, accepting an input and possessing its own error term, or bias term, along with weights.
These weights are essential as they define the significance that a specific node assigns to a particular segment of the input.
The node's output is then determined, and an activation function applied to this output ascertains if the neural network will activate or "fire."
Activation occurs if the output value surpasses a set threshold.

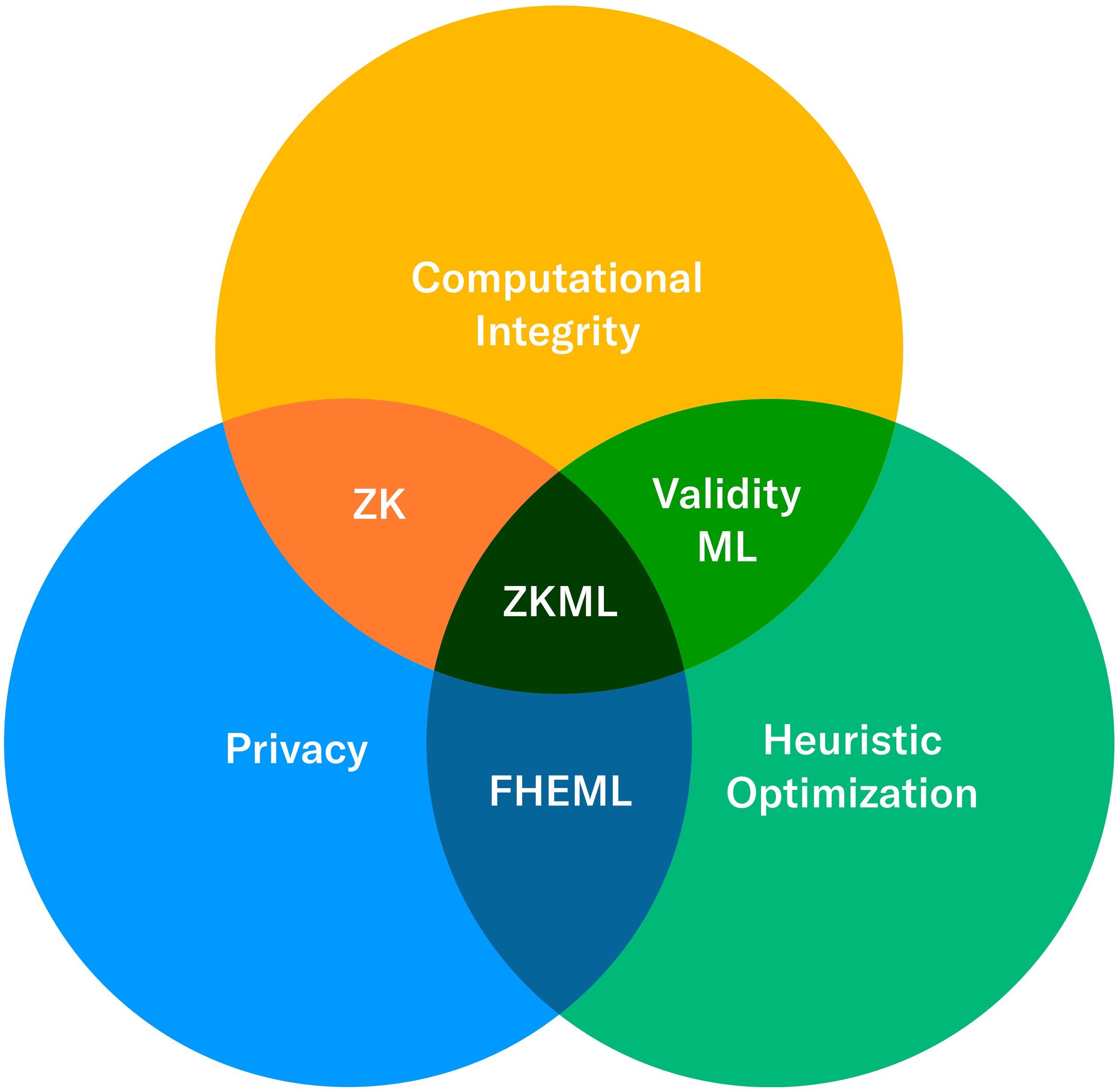
When a node activates or "fires," its output is then used as the input for another node in the network. This process continues with nodes from one layer transmitting data to the succeeding layer, culminating in a chain of data flow. This sequential data transfer from one layer to the next is what gives rise to the concept known as a "feedforward" network.



Training of the network involves 'back propagation' as is many times more expensive than 'feed forward' or inference.

zkML Introduction

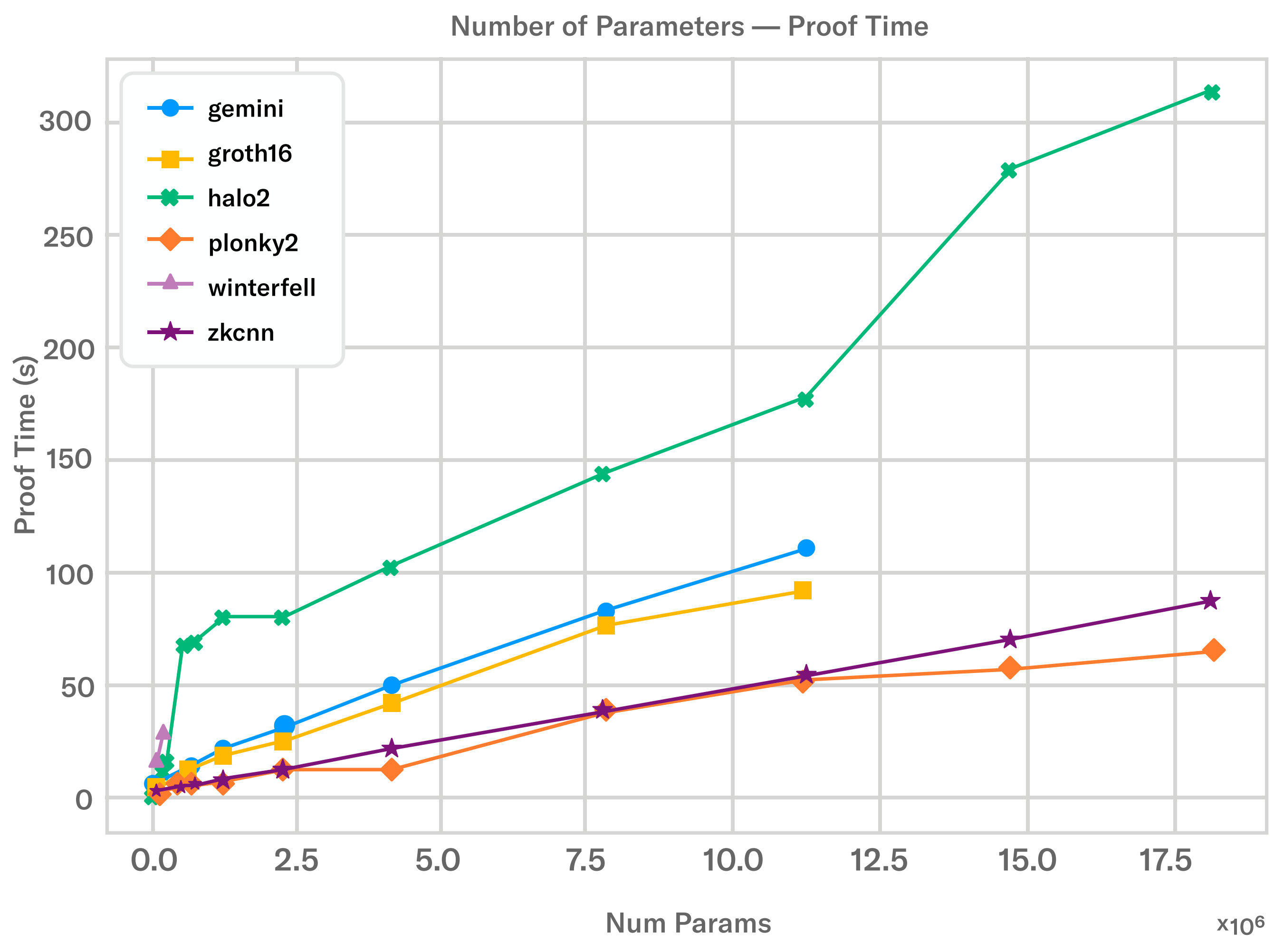
From introduction blog



The current state of zero-knowledge systems, even with high-performance hardware, is insufficient to handle large language models. But progress is being made with smaller models.

The Modulus Labs team recently released a paper titled “The Cost of Intelligence”, where they benchmark existing ZK proof systems against a wide range of models of different sizes. It is currently possible to create proofs for models of around 18M parameters in about 50 seconds running on a powerful AWS machine using a proving system like plonky2.

The following illustrates the scaling behaviour of different proving systems as the number of parameters of a neural network are increased:"



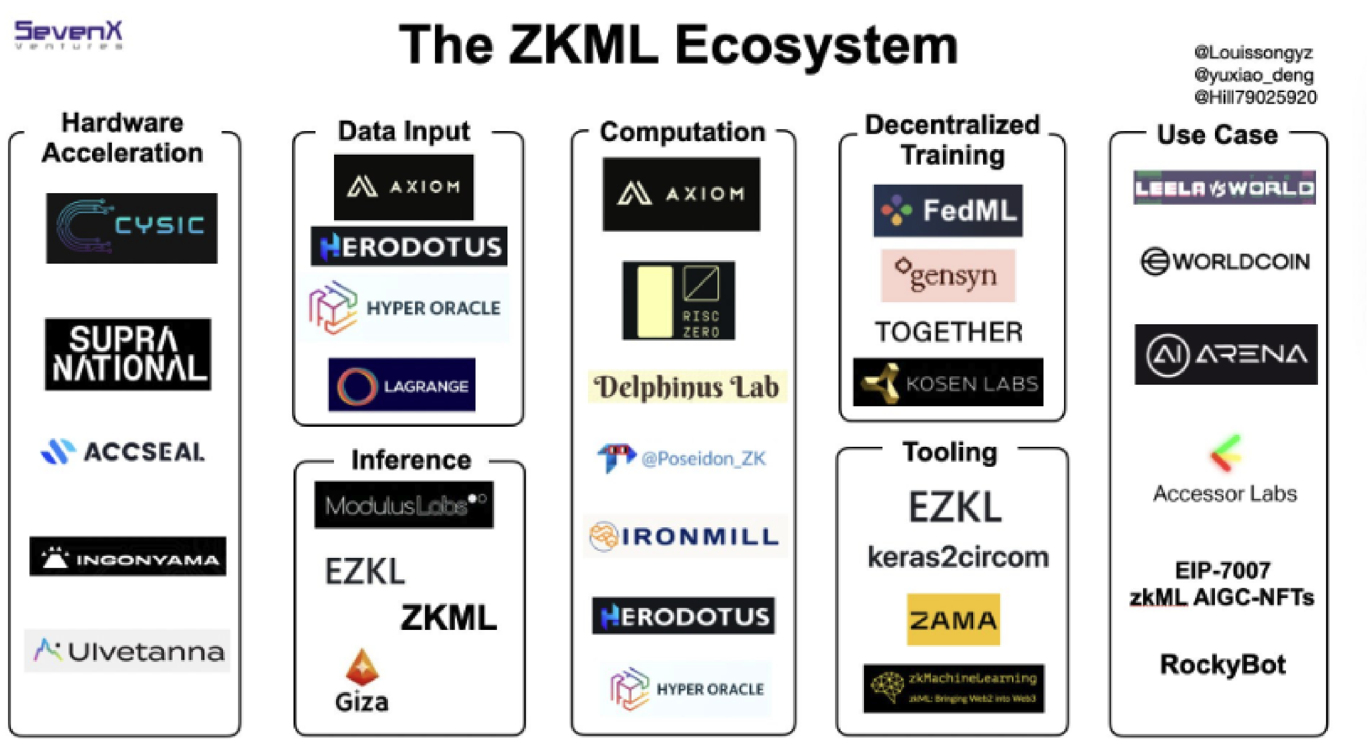
Modulus Labs

Modulus Labs are working on a number of use cases :

- On-chain verifiable ML trading bot - RockyBot
- Blockchains that self-improve vision

> - Enhancing the Lyra finance options protocol AMM with intelligent features
> - Creating a transparent AI-based reputation system for Astraly (ZK oracle)

Ecosystem



Approaches to modelling a neural network

From article by Pratyush Ranjan Tiwari

1. Naive implementation using addition/multiplication gates: The circuit would be very simple and any proof protocol could be utilised but the size of the circuit is very large $O(n^2 \cdot w^2)$ for image matrix being $n \times n$ and the weight matrix being $w \times w$. Making this method inefficient for any powerful CNN with more than a couple layers.

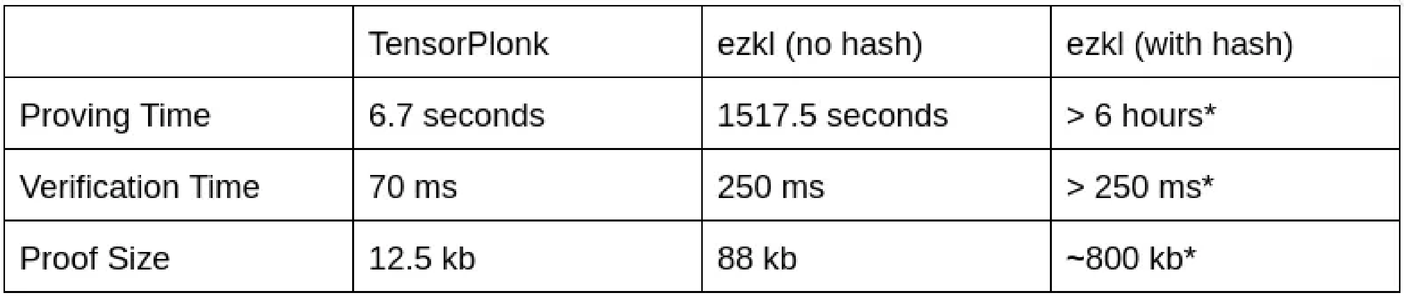
2. Compute convolutions using FFT: This was demonstrated by an S&P (Oakland) ‘20 paper ZXZS20 and results in a circuit of size $O(n^2 \log n)$ with $O(\log n)$ depth.
   Another variant of this approach was followed by the zkCNN paper by utilizing a sumcheck protocol for FFT, which is asymptotically optimal with prover time $O(n^2)$.
3. Convolution $\approx$ Polynomial multiplications: This approach was demonstrated by the vCNN paper. It uses the elegant approach of first computing the result of the convolution outside the circuit and then given the result as input, it checks equality of polynomial multiplication.
   The equality check is done by checking equality and random points, the security comes from the Schwartz-Zippel Lemma
   This is the most efficient approach as the zk proof circuit evaluating polynomials at random points is of size $O(n^2 + w^2)$.

A major problem with these approaches is the need to convert floating point numbers to integers for the proofs. Similarly efficiently proving the correctness of matrix multiplication is difficult.

> Some possible workarounds are 'quantise' the weights so that they can be represented as elements in a finite field. This still has the problem of the modular nature of the field.

Tensor Plonk

See post by Daniel Kang
TensorPlonk is a described as a “GPU” for ZKML.
TensorPlonk can deliver up to 1,000x speedups for certain classes of models, such as the Twitter system above.
Using TensorPlonk, the proving cost for the Tweet example above would be ~$30 compared to ~$88,704.



	TensorPlonk	ezkl (no hash)	ezkl (with hash)
Proving Time	6.7 seconds	1517.5 seconds	> 6 hours*
Verification Time	70 ms	250 ms	> 250 ms*
Proof Size	12.5 kb	88 kb	~800 kb*

TensorPlonk optimizes matrix multiplications, non-linearities, and weight commitments with new optimizations and by extending recent work.

The second part of TensorPlonk accelerates the non-linearities.
To understand why we need this, existing work in ZKML that uses lookups uses a lookup proving argument plookup.
The proof is split into a “circuit” and “lookup tables.”
With plookup, the circuit must be the size of the table, so if we use a large table, we must use a large circuit.
With our optimizations, the lookup table can be as large

as 2^{24} elements but the circuit can be as small as 2^{16} elements! This means we would have an extra 250x overhead to use a large lookup table .

Instead, we can leverage cached quotients : cq, which allows for lookup tables of size independent of the circuit size.
Tensor Plonk further optimised cq to “batch” work across lookup tables, which can reduce the computations for the lookups by up to 3x.

The third optimisation optimises the weight commitments, which binds the model provider to use a fixed set of weights.
Instead of using a hash, which can be extremely expensive, we use a KZG commitment of the weight vectors which can reduce the proving time by up to 10x.

Weightless networks

See article
An approach from the zero gravity team winners of the ZK Lisbon hackathon '23 was use a neural net design that didn't use weights.
From their description

"We propose a different approach: let’s go back to a time before the NN paradigm was settled, to a time when a greater variety of neural nets roamed the earth, and let’s find a machine learning model more amenable to ZKP. One such model is the “Weightless Neural Network”.

Weightless means no weights, no floating point arithmetic, and no expensive linear algebra, let alone non-linearities

A Weightless Neural Network (WNN) is entirely combinatorial. Its input is a bitstring (e.g. encoding an image), and their output is one of several predefined classes, e.g. corresponding to the ten digits. It learns from a dataset of (input, output) pairs by remembering observed bitstring patterns in a bunch of devices called RAM cells, grouped into “discriminators” that correspond to each output class. RAM cells are so called since they are really just big lookup tables, indexed by bitstring patterns, and storing a 1 when that pattern has been observed in an input string that is labeled with the class of this discriminator.

Zero Gravity is a system for proving an inference run (i.e. a classification) for a pre-trained, public WNN and a private input. In Zero Gravity, the prover claims to know an input bitstring x such that the public model classifies it as class y. The input x can be treated as a private input, in which case the system is zero-knowledge: although inference does reveal something about x to the verifier (namely its corresponding output class y), this information is already contained in the statement being proved.

Applications

Daniel Kang is working on verifying image authenticity.

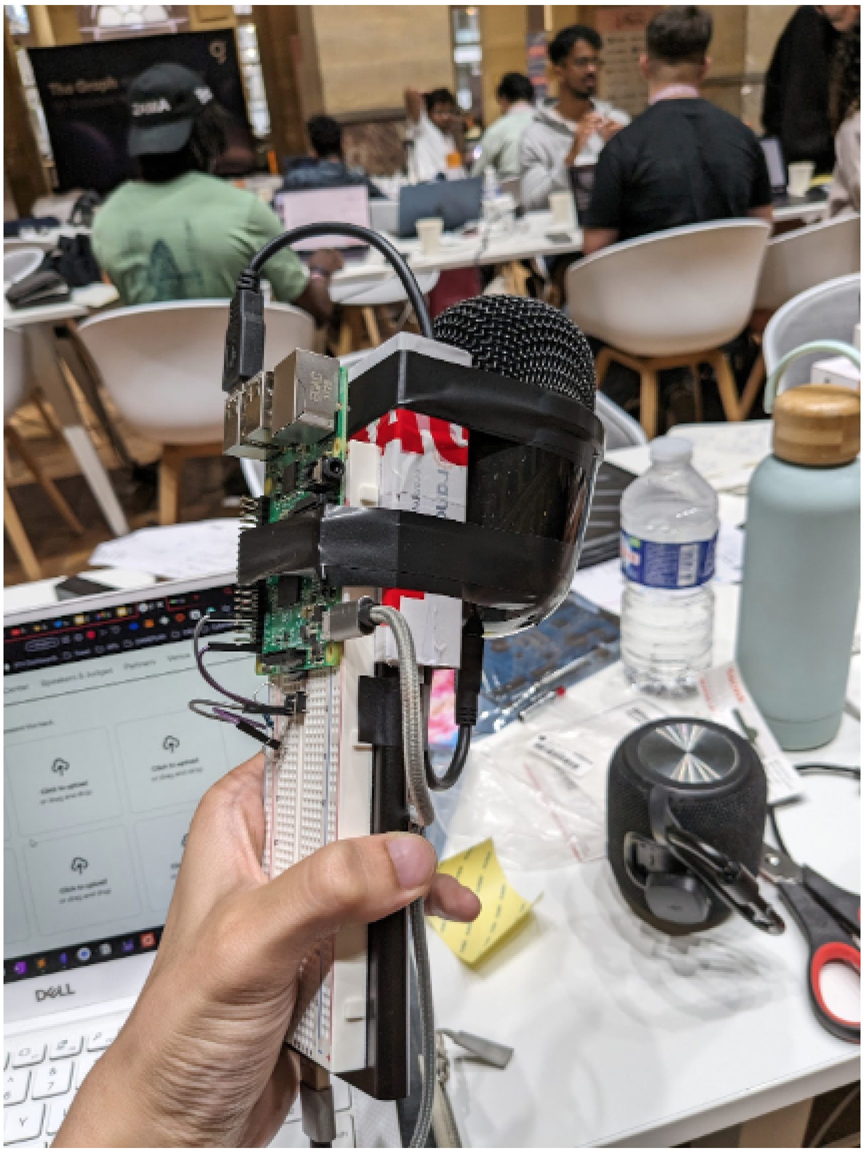
A similar project is the ZK Microphone.

See recent project from ETH Global

ZK Microphone

🎙🔒 ZK Microphone: Trusted audio in the age of deepfakes 🔒🎙 Generative AI is a threat to society. It enables disinformation, manipulation, and political subversion. We've built the world's first attested microphone and used ZK-SNARKs to protect authenticity and privacy.

Source Code



Also see this talk

EZKL

See Repo
See Introduction from zkSummit 10

Ezkl is a library and command-line tool for doing inference for deep learning models and other computational graphs in a zk-snark.
It enables the following workflow:

1. Define a computational graph, for instance a neural network (but really any arbitrary set of operations), as you would normally in pytorch or tensorflow.
2. Export the final graph of operations as an .onnx file and some sample inputs to a `.json` file.
3. Point `ezkl` to the `.onnx` and `.json` files to generate a ZK-SNARK circuit with which you can prove statements such as:
   Halo2 is used as a backend.

ddkang/zkml

This is a project for constructing proofs of ML model execution in ZK-SNARKs.
See paper and Blog post Trustless verification

Giza

Applications

- **Smart Contracts Enhanced by AI**: StarkNet smart contracts can now incorporate machine learning (ML) elements into their operational logic.
- **Gaming with AI Agents**: Integrate AI-driven characters that exist and interact entirely on-chain within games developed on StarkNet.
- **Machine Learning Inference on Ethereum's L1 Layer**: Execute inferences on StarkNet, sending the outcomes to L1 for decision-making within L1 contracts using results from the implemented models.
- **Web2 Inference Delivery via Oracle**: Deploy a robust, fully accessible, censorship-resistant model that is both traceable and efficient for inference services in web2 solutions, facilitated by an oracle.

Targeted Initiatives

Orion: ONNX Runtime

ONNX Runtime serves as a multi-platform accelerator for machine learning models, designed with a versatile interface to allow for integration with hardware-specific libraries. It's compatible with various frameworks including PyTorch, Tensorflow/Keras, TFLite, scikit-learn, and more.

With Orion, a new Cairo 1.0-based ONNX runtime is introduced, aiming to supply a verifiable execution environment for ML model inferences utilizing STARKs.

Collaboration with Kleros

Orion's ability to translate machine learning languages into a certifiable smart contract language enables Giza to bring simple ML models on-chain, thereby significantly enhancing the functionality of smart contracts.

Structured as a Schelling-point coordination game, Kleros Court calls upon random token holders to serve as jurors. Their votes on the given case are guided by specific rules and evidence. Inconsistent and wrong decisions are minimized through a sturdy appeals process and crypto-economic incentives, leading to the majority vote forming the case's final ruling.

Applications of Kleros have expanded into areas such as DeFi insurance claims, eCommerce dispute resolution, DAO governance, identity verification, data curation, and social recovery of assets.

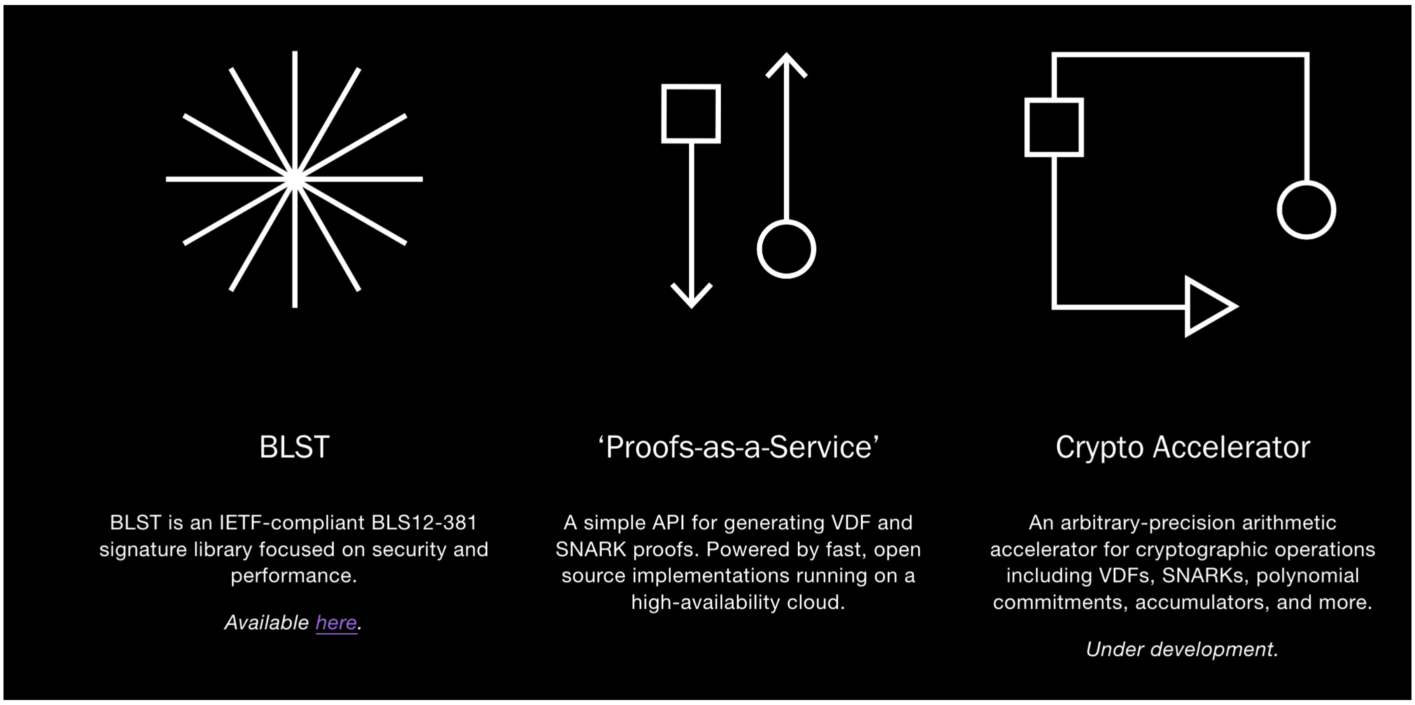
Humanity Verification Process

> Kleros has innovated a decentralized method for authenticating user-submitted videos, in which jurors wager on the authenticity of submissions—whether they are fraudulent or genuine.
>
> The utilization of on-chain ML can markedly boost efficiency and lower the expenses related to human judgment, thus facilitating the large-scale implementation of social authentication layers. By integrating supplementary data sources like wallet transaction history and image recognition, ML models are able to generate scores that guide human decision-making where applicable.

> ## ZKaggle
>
> See repo
> "Our browser-based frontend allows bounty providers to upload their data to Filecoin and set up a computing task with bounty rewards. Bounty hunters can browse all open bounties, download the data onto their local machine, and compute.
> When they are ready to submit, they construct a ZK proof to submit the hashed computed results on-chain."

Supranational

See site



BLST

BLST is an IETF-compliant BLS12-381 signature library focused on security and performance.

Available here.

‘Proofs-as-a-Service’

A simple API for generating VDF and SNARK proofs. Powered by fast, open source implementations running on a high-availability cloud.

Crypto Accelerator

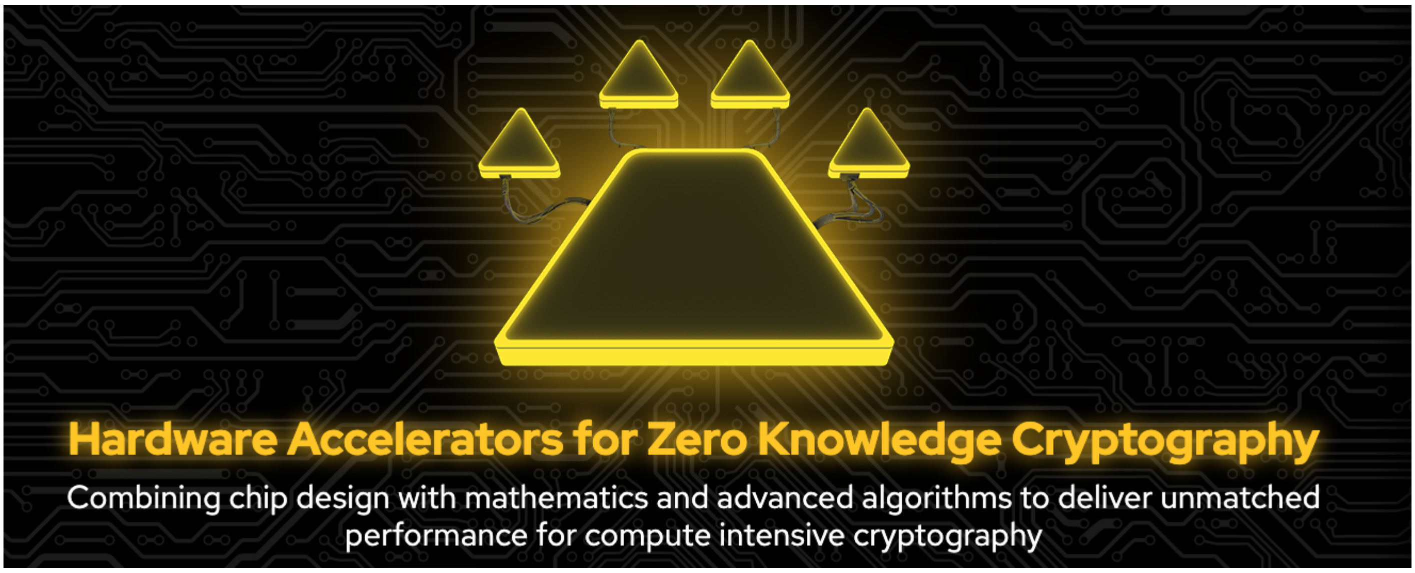
An arbitrary-precision arithmetic accelerator for cryptographic operations including VDFs, SNARKs, polynomial commitments, accumulators, and more.

Under development.

Also see VDF day videos

Ingonyama - Hardware for zkML

See Site


Hardware Accelerators for Zero Knowledge Cryptography
Combining chip design with mathematics and advanced algorithms to deliver unmatched performance for compute intensive cryptography

See this talk about the 'ZPU'

zkML Resources

dcbuilder has many resources

Introduction to zkML from Worldcoin

Worldcoin awesome repo

Zero Knowledge Podcast

Image redacting using zkSNARKS paper

ZK ML community [calls](- ZKML community call #0) on telegram

Research from PSE team at EF