Lesson 13

Today's topics

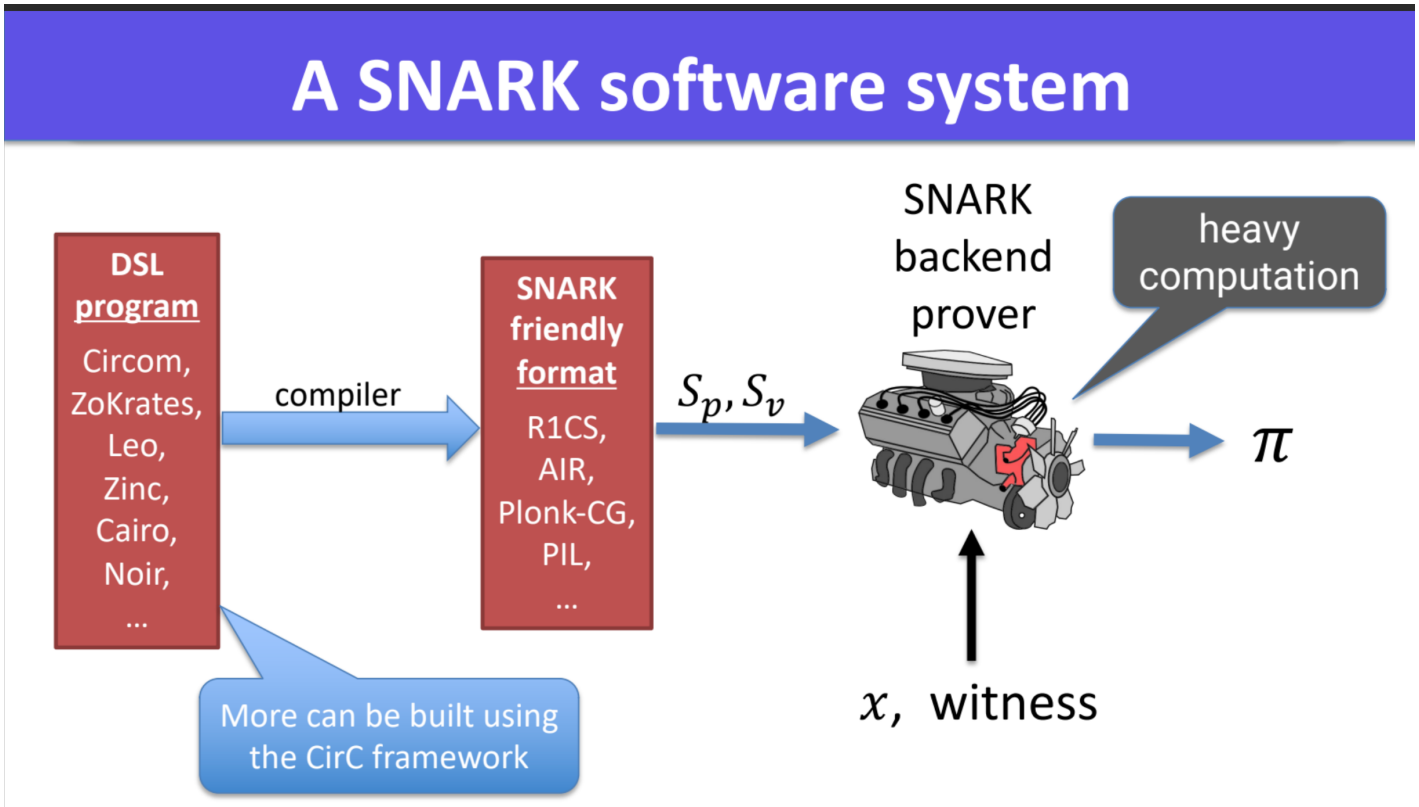
- zk modularity
- ORA.IO
- zkAudits
- Galadriel
- Hardware updates
- Research Areas
- Course Review
- Next Steps

News

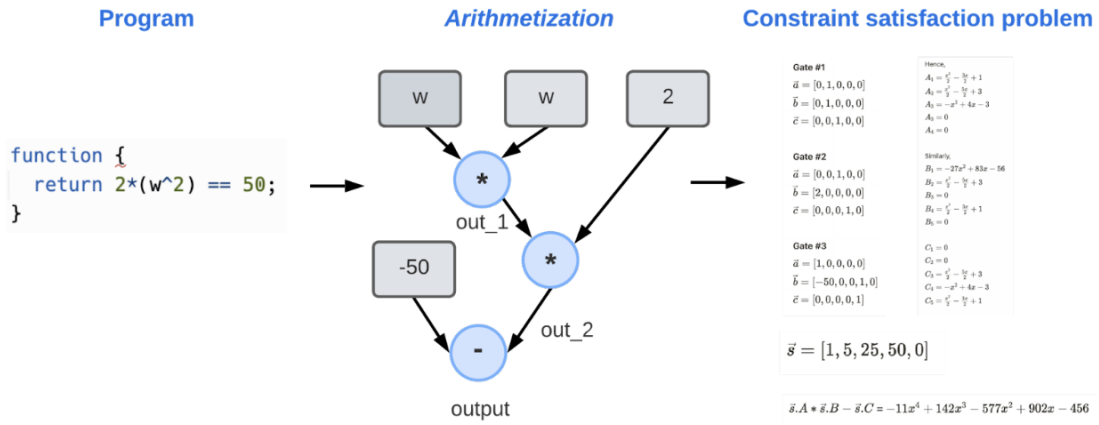
Partnership between Ingonyama and Aligned Layer announced

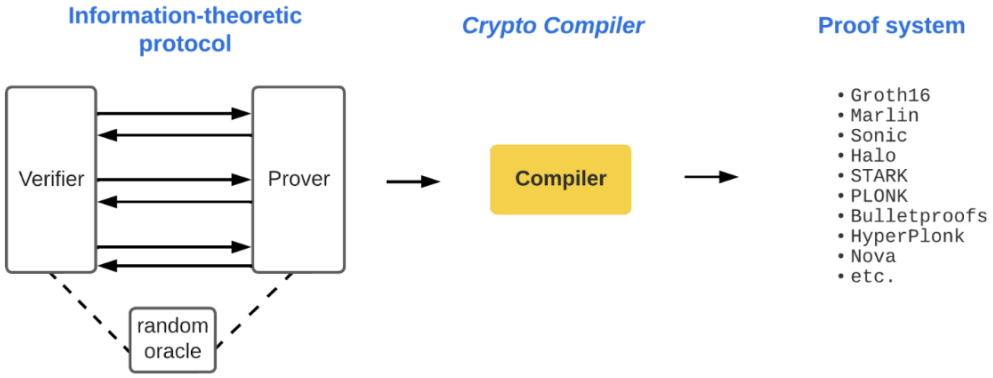
Modularity in ZKP Systems

Lifecycle of a Proof



From [zkCamp]



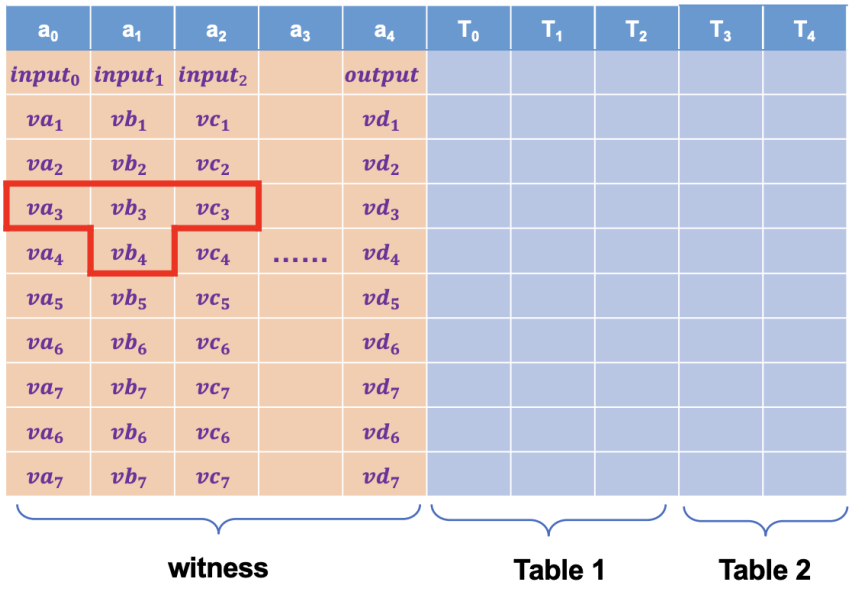


Arithmetisation

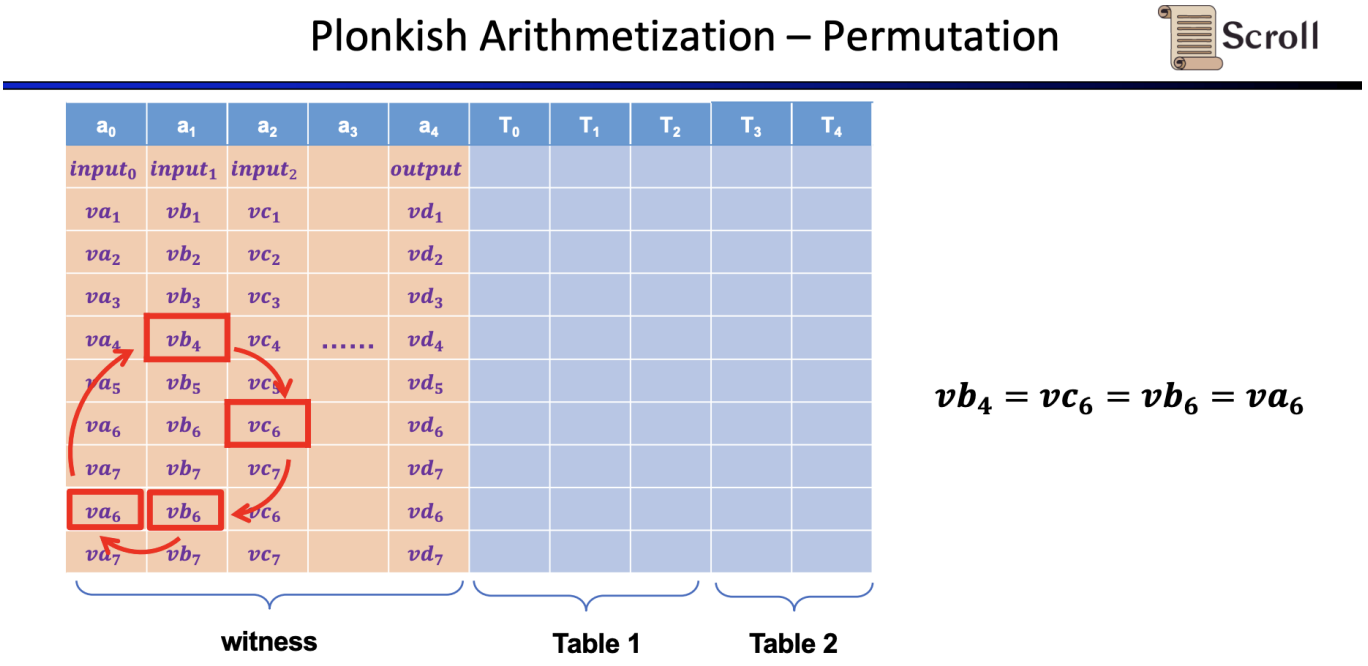
Example Arithmetisation in Scroll

Scroll custom gate

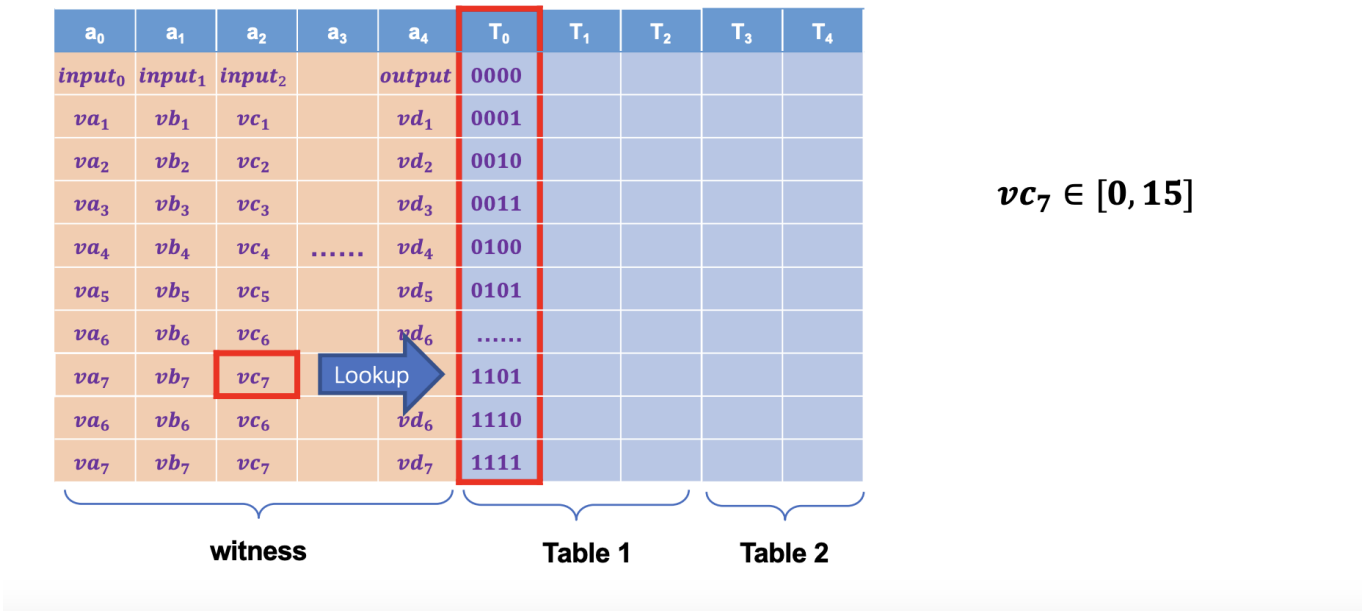
Taken from presentation



$$va_3 * vb_3 * vc_3 - vb_4 = 0$$



$$vb_4 = vc_6 = vb_6 = va_6$$



Interactive Oracle Proofs

In an IOP, the prover and the verifier interact with one another and have access to random oracles. IOPs make interactive proofs more efficient by using oracles to reduce communication and query complexity.

Information-theoretic proof system

A typical zkSNARK or zkSTARK will consist of two components

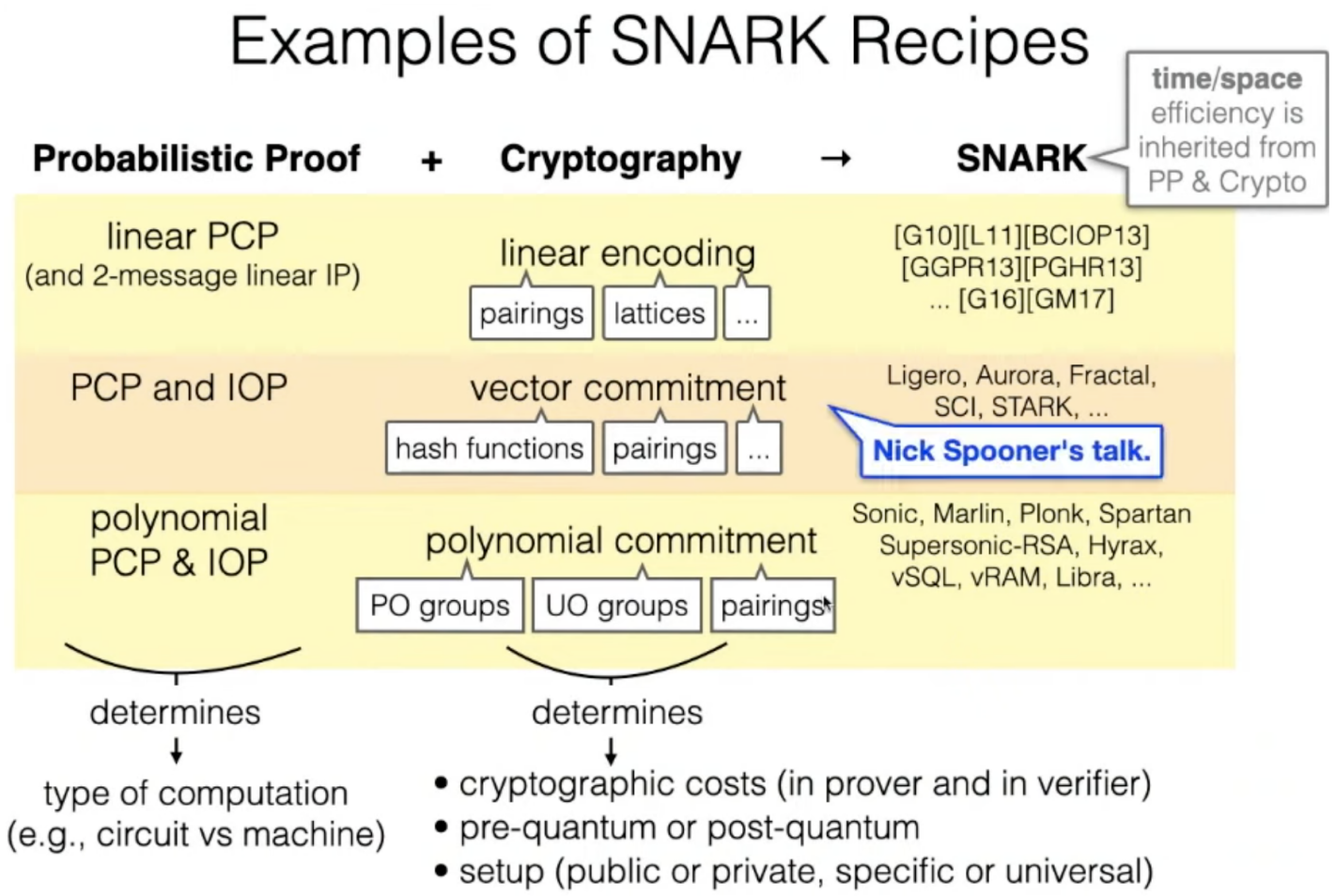
- Information-theoretic proof system
- Cryptographic compiler

The information-theoretic aspect deals with ensuring the soundness and completeness of the system, but doesn't specify the practicalities involved in implementing this.

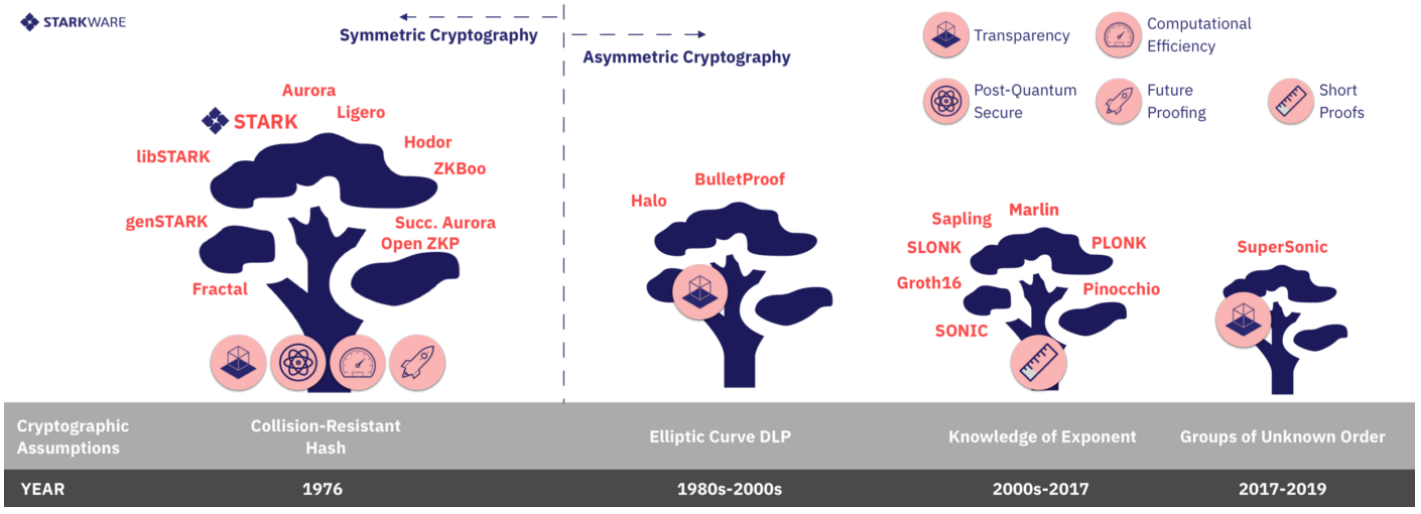
Cryptographic compiler

The cryptographic compiler is the practical aspect using cryptographic primitives such as random oracles, the resulting proof system is secure under a computationally bounder prover / verifier.

From zkCamp

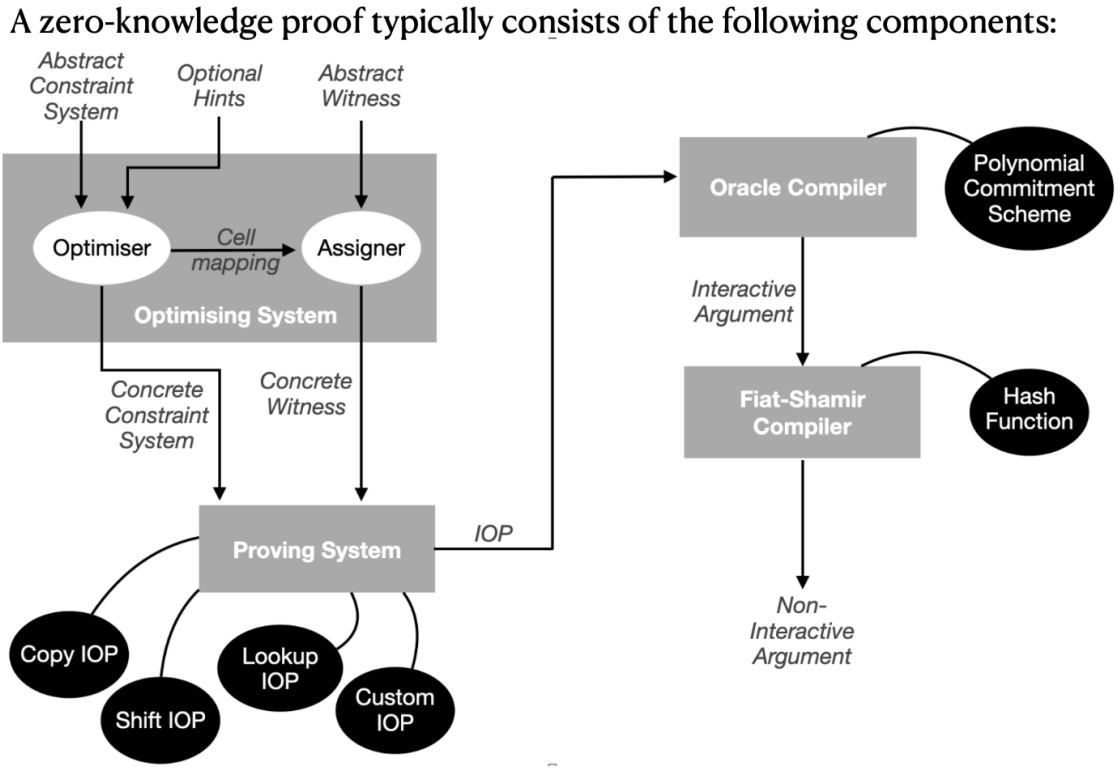


Families of Proving Systems



This (older) representation is useful when getting an initial mental model of the ecosystem.
The real ecosystem however is much more complex, as proving systems borrow technologies and approaches from each other.

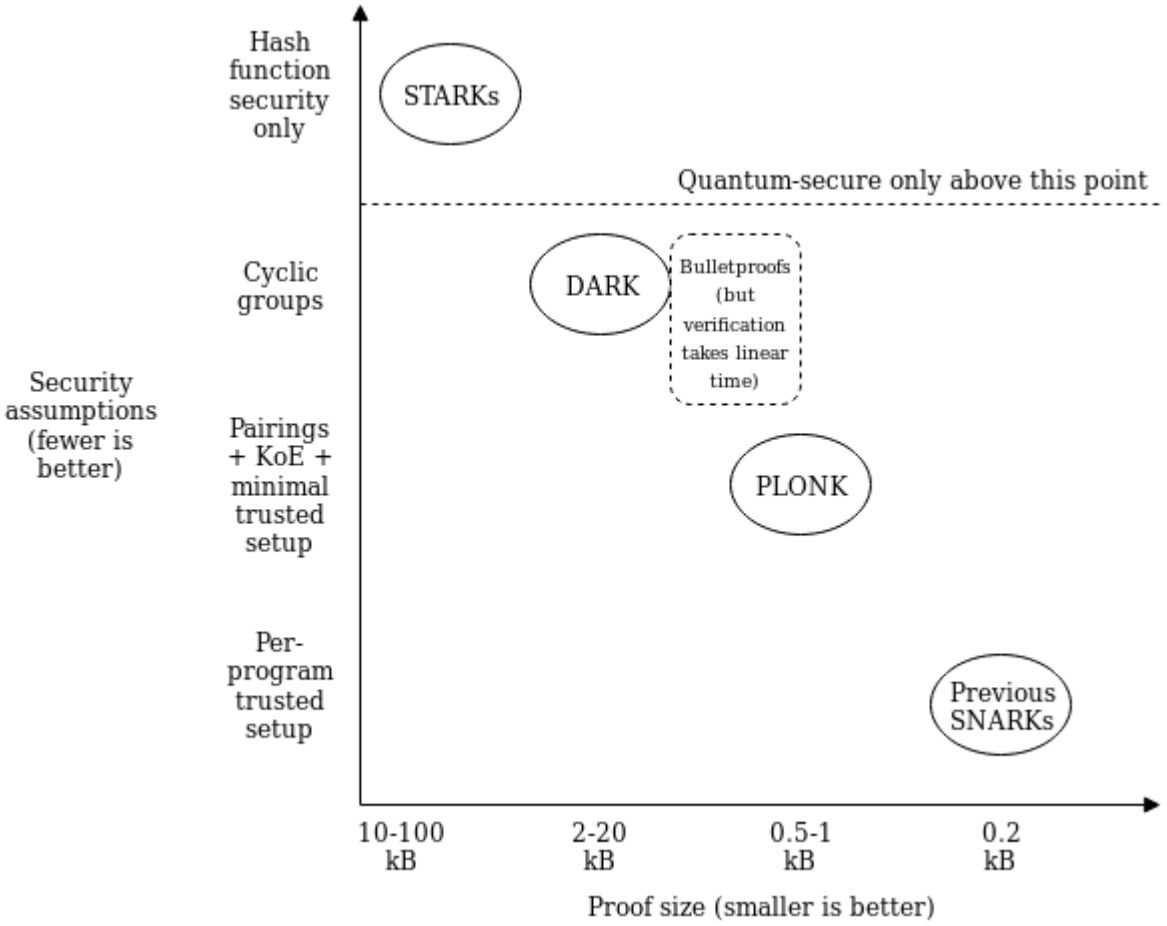
The Plonk 'standard'



Polynomial Commitments in PLONK

PLONK uses Kate commitments based on trusted setup and elliptic curve pairings, but these can be swapped out with other schemes, such as FRI (which would turn PLONK into a kind of STARK

Polynomial Commitment Schemes



This means the arithmetisation - the process for converting a program into a set of polynomial equations can be the same in a number of schemes.
If this kind of scheme becomes widely adopted, we can thus expect rapid progress in improving shared arithmetisation techniques.

Halo2

See tutorial
See documentation
See Halo2 Book

Halo 2 is a proving system that combines the Halo recursion technique with an arithmetisation based on PLONK, and a polynomial commitment scheme based around the Inner Product Argument

History

Chips

Using our API, we define chips that "know" how to use particular sets of custom gates. This creates an abstraction layer that isolates the implementation of a high-level circuit from the complexity of using custom gates directly.

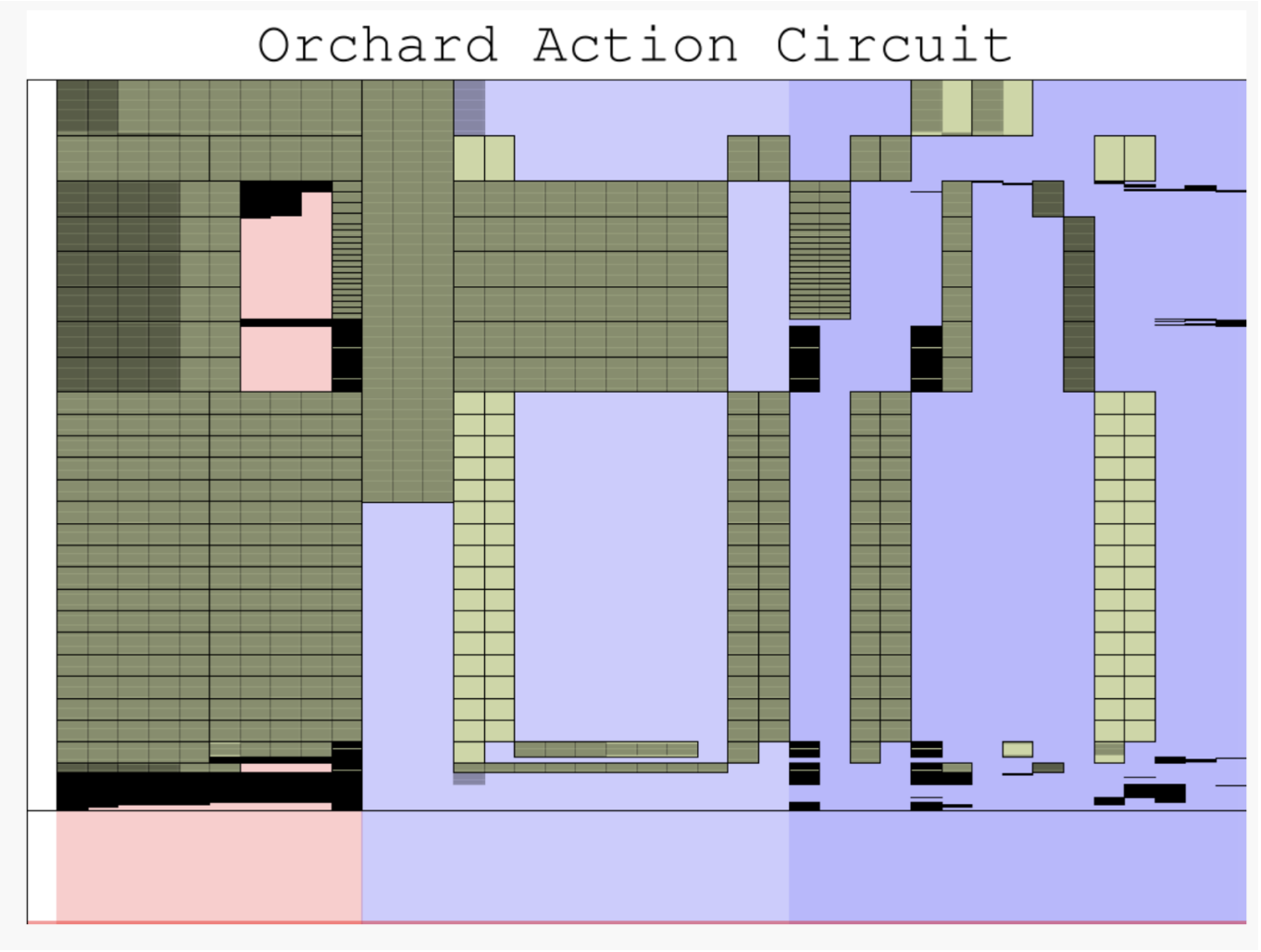
Example Simple Circuit

```
trait NumericInstructions<F: Field>: Chip<F> {
    /// Variable representing a number.
    type Num;

    /// Loads a number into the circuit as a private input.
    fn load_private(&self, layouter: impl Layouter<F>, a: Value<F>) ->
Result<Self::Num, Error>;

    /// Loads a number into the circuit as a fixed constant.
    fn load_constant(&self, layouter: impl Layouter<F>, constant: F) ->
Result<Self::Num, Error>;

    /// Returns `c = a * b`.
    fn mul(
        &self,
        layouter: impl Layouter<F>,
        a: Self::Num,
        b: Self::Num,
    ) -> Result<Self::Num, Error>;

    /// Exposes a number as a public input to the circuit.
    fn expose_public(
        &self,
        layouter: impl Layouter<F>,
        num: Self::Num,
        row: usize,
```

```
    ) -> Result<(), Error>;
}
```

Halo 2 circuits are two-dimensional: they use a grid of "cells" identified by columns and rows, into which values are assigned.
Constraints on those cells are grouped into "gates", which apply to every row simultaneously, and can refer to cells at relative rows.
To enable both low-level relative cell references in gates, and high-level layout optimisations, circuit developers can define "regions" in which assigned cells will preserve their relative offsets.

Example from ZCash



In the example circuit layout pictured, the columns are indicated with different backgrounds.
The instance column in white;
advice columns in red;
fixed columns in light blue; and
selector columns in dark blue.
Regions are shown in light green, and assigned cells in dark green or black.

Column Types

- Instance columns contain per-proof public values, that the prover gives to the verifier.
- Advice columns are where the prover assigns private (witness) values, that the verifier learns zero knowledge about.
- Fixed columns contain constants used by every proof that are baked into the circuit.
- Selector columns are special cases of fixed columns that can be used to selectively enable gates.

Powdr

See Docs

powdr is a modular compiler stack to build zkVMs. It is ideal for implementing existing VMs and experimenting with new designs with minimal boilerplate.

- Domain specific languages are used to specify the VM and its underlying constraints, not low level Rust code
- Automated witness generation
- Support for multiple provers as well as aggregation schemes
- Support for hand-optimized co-processors when performance is critical
- Built in Rust 🦀

See video from Christian Reitwiessner

Installation

Rust is a prerequisite
Instructions are here

Front Ends

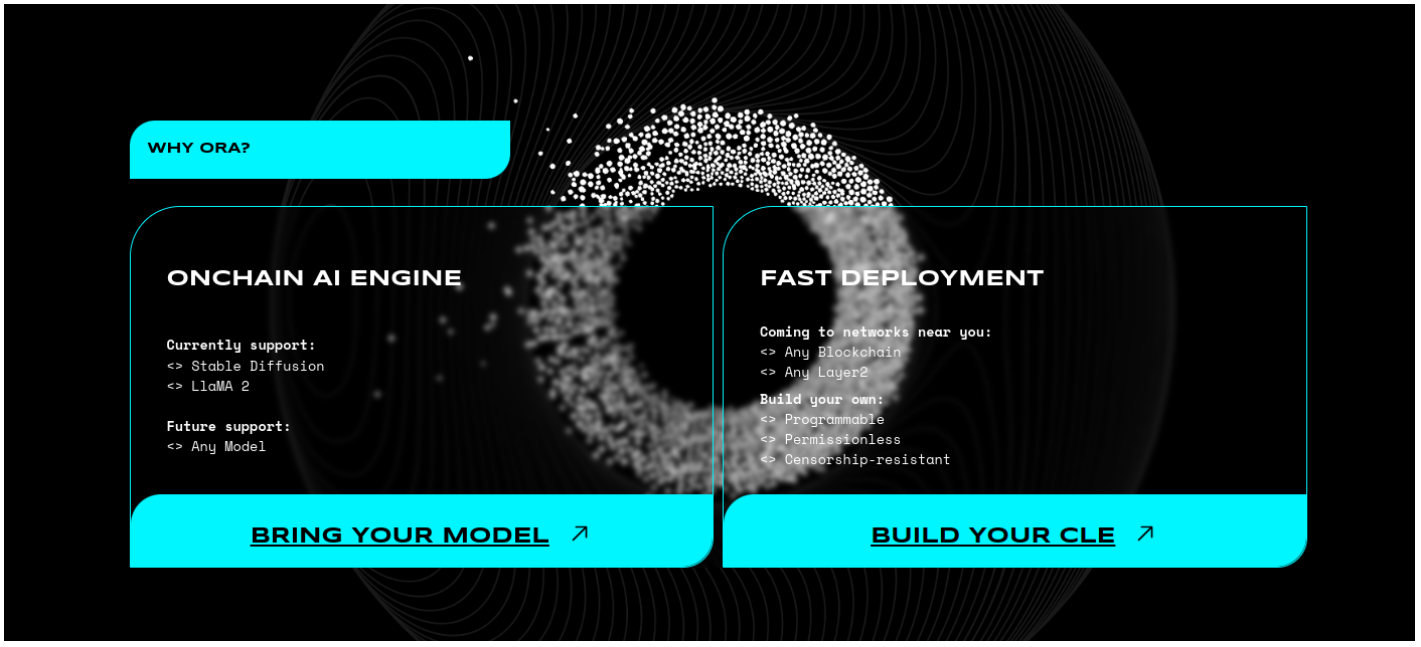
A Risc V frontend is available and others are under development.

Backends

Halo 2 and eSTARK are supported

ORA

See https://www.ora.io/
See paper
See Repo



Two main products of ORA:

- **AI Oracle (OAO)**: Brings AI onchain.
- **zkOracle**: Brings complex compute and historical data onchain.

ORA is a decentralised protocol and network:

Developers can define custom computations, register those computations to ORA protocol, and utilise ORA CLE standards to add automation or AI to their smart contracts.

Node operators can run AI / ZK Oracle nodes to execute and secure computations with verifiable proofs.

Community

Discord : https://discord.gg/MgyYbW9dQj

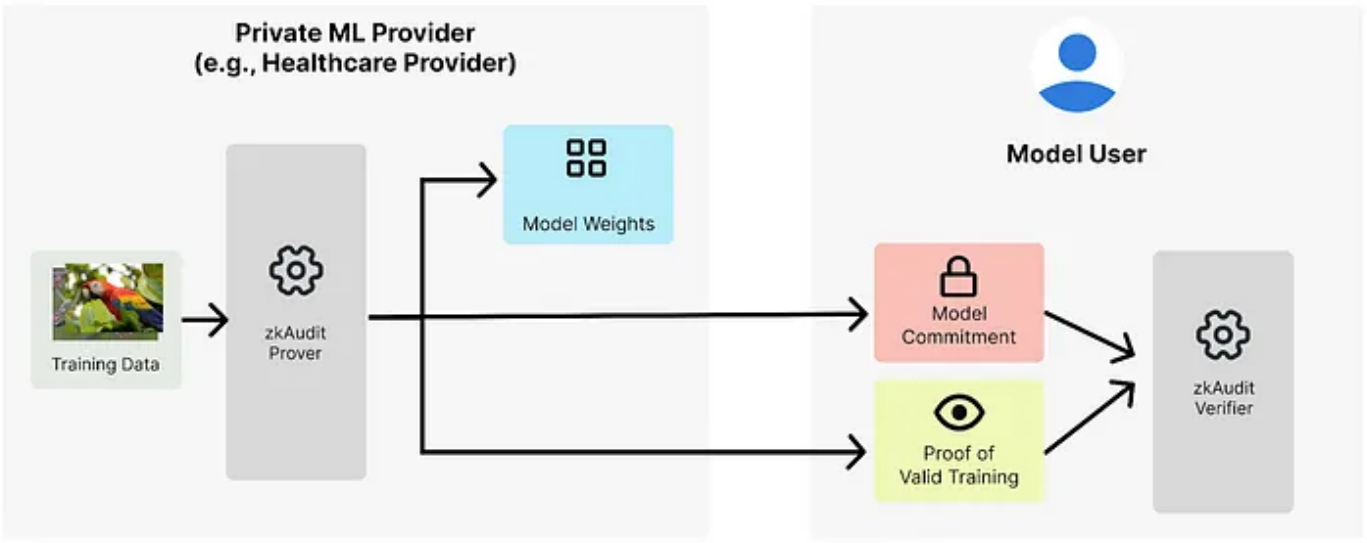
A Scholar program is available

ZK Audit

Trustless audits of machine learning

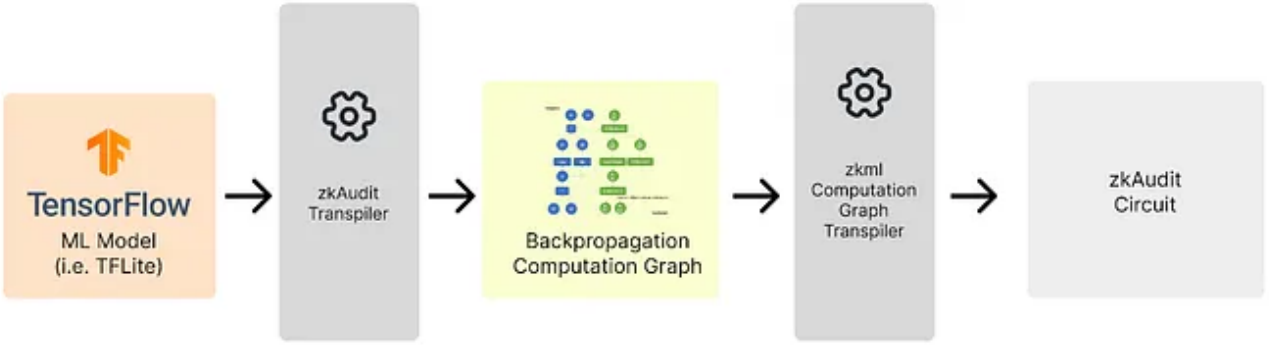
From [article]

"With ZKAudit, a provider of an ML model can prove that a private model was trained from private data and subsequently prove properties about the trained model, such as whether or not it contains specific copyrighted material. To accomplish this, ZKAudit uses zero-knowledge proofs to ensure the validity of the ML model provider."



Proof of training

To enable proofs of training, we created a transpiler within the zkml framework that creates backpropagation computation graphs which is compiled into zero-knowledge proof circuits. This allows ZKAudit to utilize zkml's existing optimized linear algebra and non-linearity operations.



Galadriel AI

See Docs

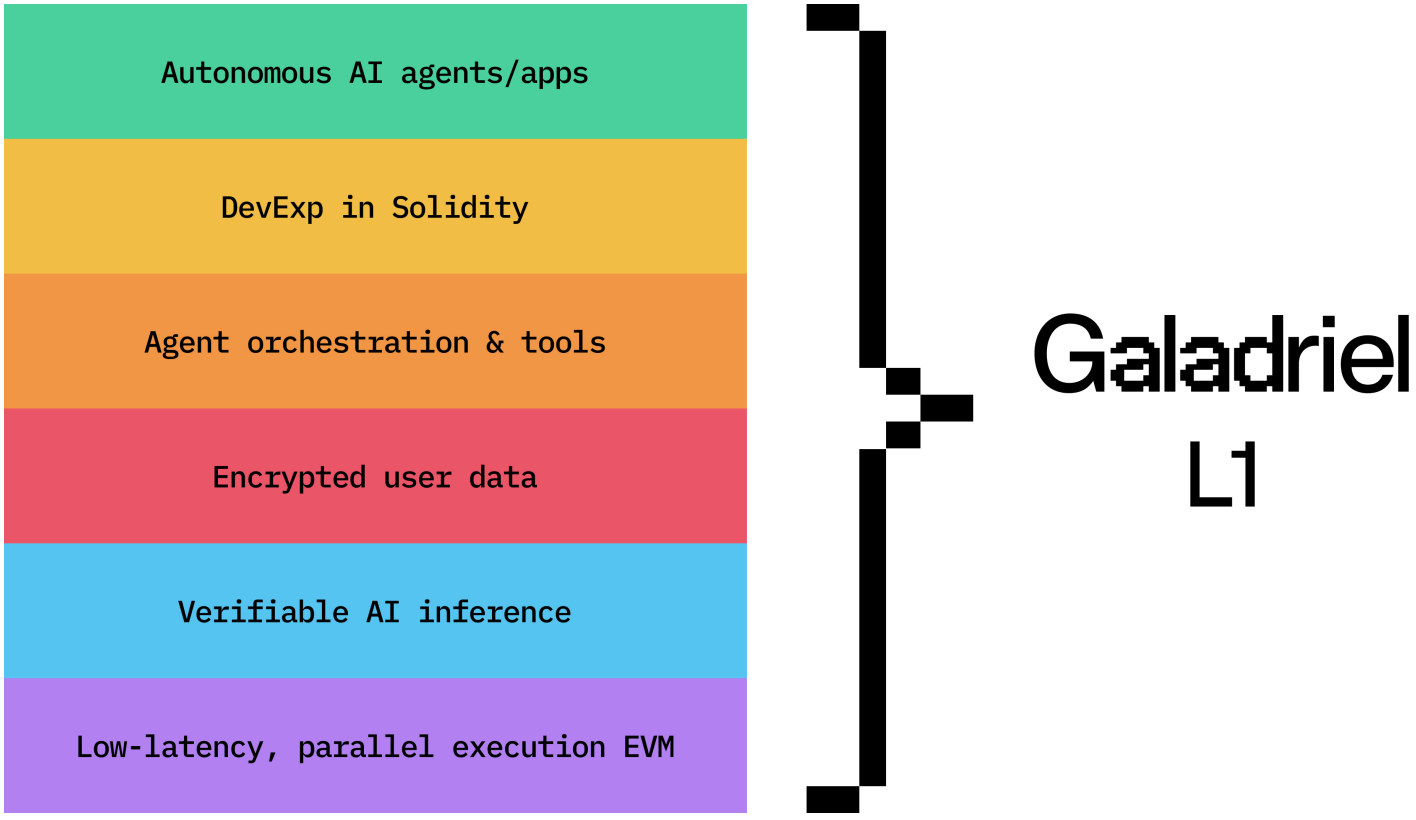
Introduction

Galadriel's mission is to build **safe, citizen owned AI by building ETH for AI**. Our long-term vision can be unpacked as:

1. Build the first L1 for AI which acts as the settlement layer for AI, driving real GDP growth.
2. Provide direct ownership and democratic governance over AI.
3. Become the most secure rails for AI deployment and largest applied safety network where crypto-incentives drive humanity to align AI.

Galadriel brings AI inference on-chain in a low-cost, low-latency manner through teeML (Trusted Execution Environment Machine Learning) which allows querying open and closed-source LLM models in a verifiable way.

Galadriel is built on a parallel-execution EVM stack which enables high throughput and low latency while providing a familiar experience to Solidity developers. It brings AI inference on-chain in a low-cost, low-latency manner through teeML (Trusted Execution Environment Machine Learning) which allows querying open and closed-source LLM models in a verifiable way.



zk Hardware Updates

See updates from ZK Summit 11
video

Accseal

See Site

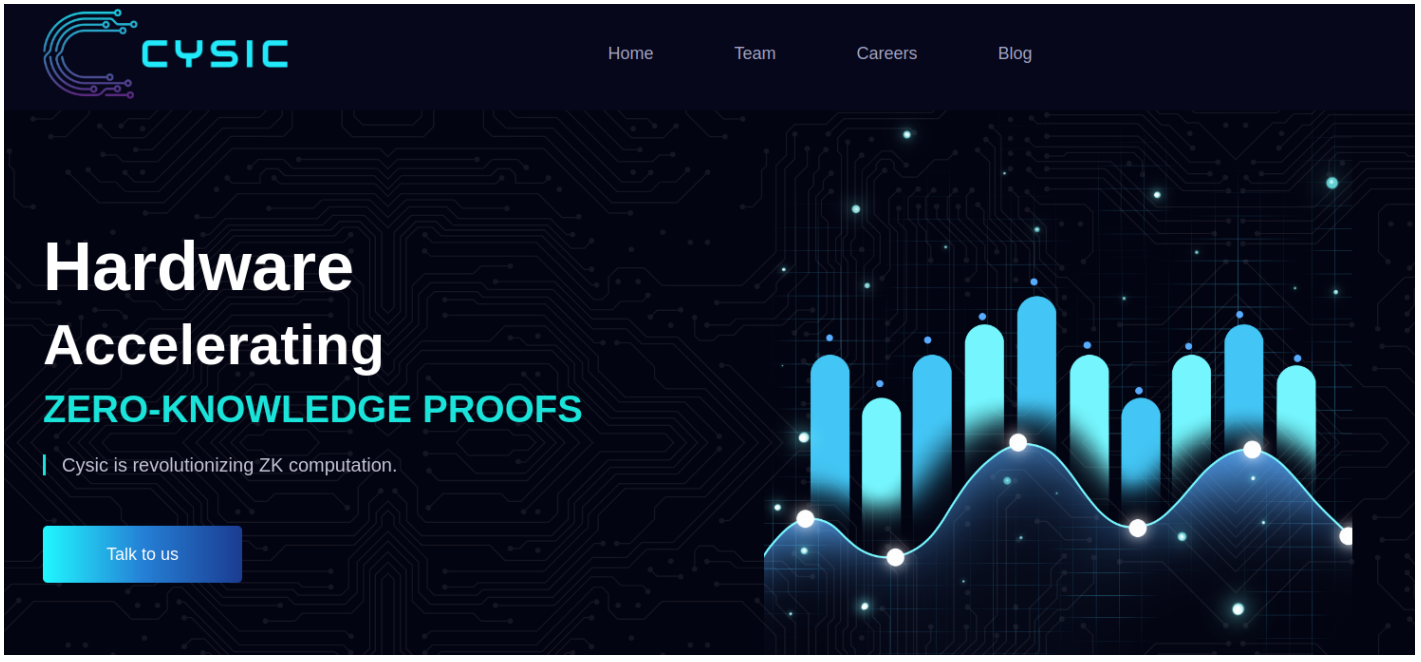
Deep research on privacy computing

Privacy computing is a data security management system and method covering the entire life cycle of private information ; A type of information technology that allows users to realize data analysis and calculation without the data itself being leaked to the outside world.

ACCSEAL will develop a high-efficiency ASIC dedicated chip for privacy computing, which will greatly improve the computing speed. At the same time, the cost of chips is reduced, making the widespread application of privacy computing possible.

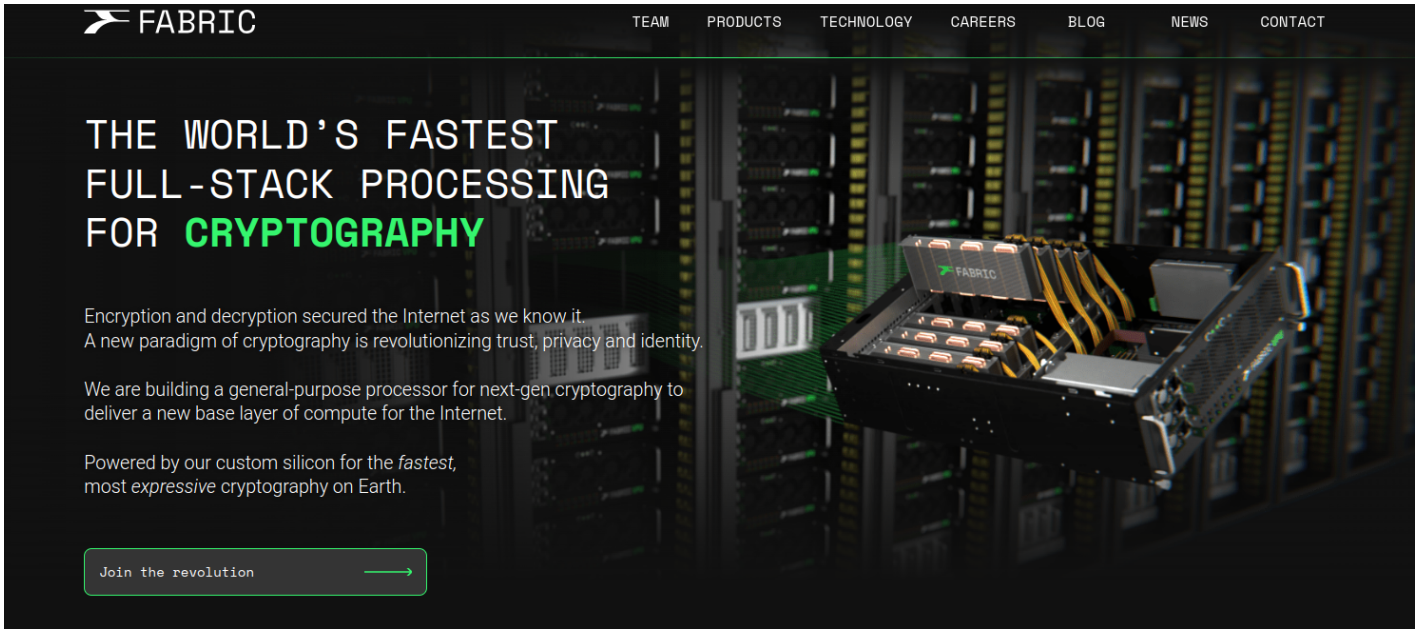
Cysic

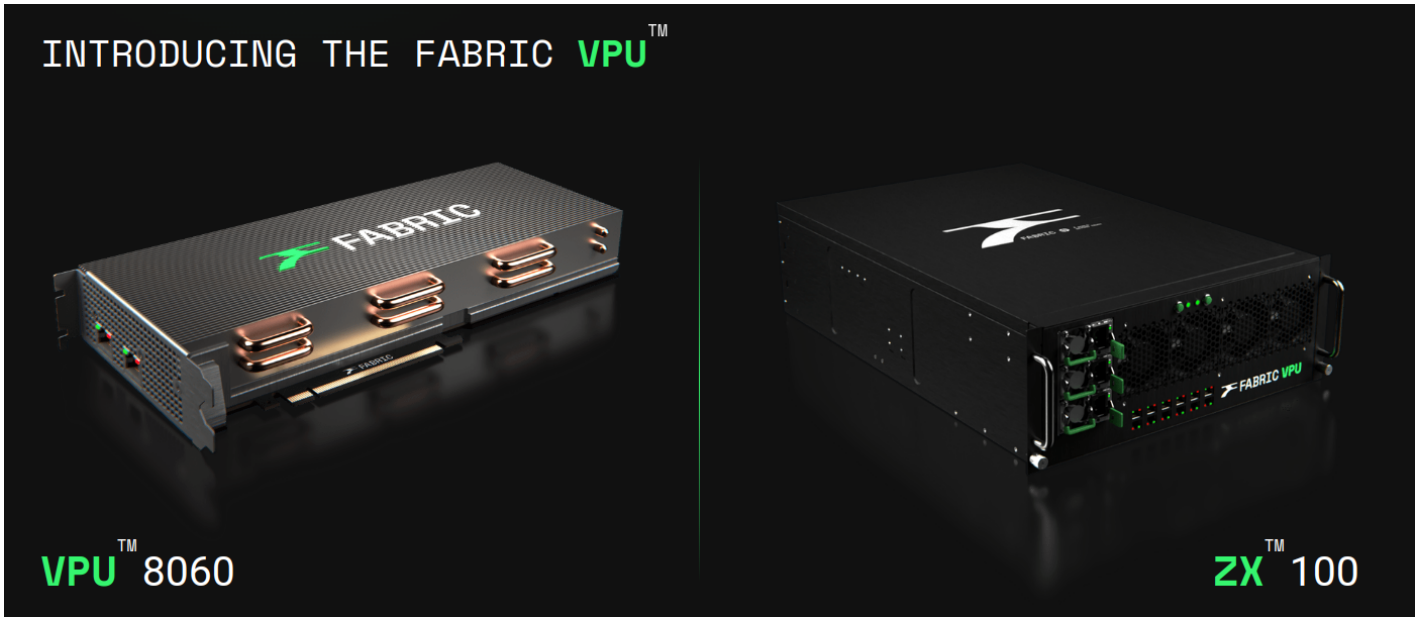
See site
See article



Fabric

See site





VPU™ 8060

ZX™ 100

The **Fabric Verifiable Processing Unit (VPU)** is the world's first massively parallel processor for cryptography. The VPU's cryptography-native instruction set can run a wide range of cryptography workloads, including **zero knowledge proofs (ZKP),** and **fully homomorphic encryption (FHE)**, much like a GPU facilitates a wide range of graphics and AI applications.

The **Fabric ZX 100 server**, our inaugural server product, has garnered over $60 million in customer pre-orders and will enter mass production in 2024. The **Fabric VPU 8060,** our PCIe card offering, will also be available in 2024 for development partners and researchers to accelerate their projects and research on the VPU.

Research Areas

zkML

opML Paper

Secure and Verifiable Data Collaboration with Low-Cost Zero-Knowledge Proofs Paper

Verifiable evaluations of machine learning models using zkSNARKS - Paper

Zero-Knowledge Machine Learning to Enhance Trust in Generative AI Interactions Paper

FairProof Paper

ezDPS: An Efficient and Zero-Knowledge Machine Learning Inference Pipeline Paper

ZKP

A good overview is given in this report for Q4 2023 from ZKV

Updates from their latest report

Research

STIR - Reed-Solomon Proximity Testing with Fewer Queries

Paper

Beyond the Circuit

Minimising foreign arithmetic in ZKP circuits
Paper

Circle Starks

See Paper
Introduces a simpler construction similar to EC FFT using the circle curve $x^2 + y^2 = 1$, giving a speed up of 1.4

zkPi

Proving Lean Theorems in zero knowledge
See Paper
Lean is a interactive theorem prover for formal verification.

Parallel zkVM

See Paper
This introduces a zkVM using data parallel circuits, the parallelisation happens at the opcode and the basic block level.

Products

SP1 (Succinct)

See repo
SP1 is a performant, 100% open-source, contributor-friendly zero-knowledge virtual machine (zkVM) that can prove the execution of arbitrary Rust (or any LLVM-compiled language) programs. SP1 democratizes access to ZKPs by allowing developers to use programmable truth with popular programming languages.

zkSnap

See paper
Introduces private voting for DAOs

Lilith (EZKL)

A high-performance backend system for zero-knowledge proof generation at scale.
For Beta access use this form

Brevis

A Smart ZK Coprocessor for Blockchains
See article

Nebra

Universal proof aggregation
See site

ML

A Survey on Large Language Model-Based Game Agents
Autonomous Code Improvement
Tiny Llama - An Open-Source Small Language Model

Course Review

Lesson 1 - Fundamentals

- Fundamentals of ZKP and ML
  - Maths
  - Cryptography
  - Introduction to ZKP Theory
  - Computer architecture
    - Virtual machines
    - Opcodes

Lesson 2 - Introduction to ML

- Maths for ML
- Machine Learning Introduction
- (Un) Supervised Learning
- Neural network introduction
- Components
  - Nodes
  - Weights
  - Activation functions
- Hardware

Lesson 3 - Intro to zkML / Use Cases

- zkML introduction
- Ecosystem
- Use cases

Lesson 4 - EZKL Workshop

Lesson 5 - zkML timeline / challenges / Worldcoin

- Timeline of zkML
- Challenges
- dcBuilder AMA

Lesson 6 - Giza / ZKP Process

- Giza
- zkSNARK process
- Polynomial Commitment Schemes
- Fiat Shamir Heuristic

- Activation Functions
- ONNX

Lesson 7 - Modulus Workshop

Lesson 8 - Ingonyama / zk Games

- Reproducibility in zkML
- zk Games and AI

zkGames is in the video for Lesson 9

Lesson 9 - zkML Projects

- Tensor Plonk
- Zero Gravity

Lesson 10 - zkML Projects / Hardware

- Axiom
- ERC-7007: Verifiable AI-Generated Content Token
- Hardware for zkML
- Netron
- Taceo
- Modulus - GPT2

Lesson 11 - GKR / Federated ML

- Sumcheck protocol
- GKR
- IVC
- Folding Schemes
- Federated Machine Learning
- Aligned Layer

Lesson 12 - ZAMA / FHE

Lesson 13 - Review

- zk modularity
- ORA.IO
- zkAudits
- Galadriel
- Hardware updates
- Course Review
- Next Steps

Next Steps

Encode Events and Bootcamps

See Events

Events

I will be giving a zkML workshop at ETH Prague (31/5 - 2/6)

Grants programs

Many, for example
Polygon
ZkSync
Mina
Worldcoin
Ora Scholar program