Lesson 15 - Identity Solutions / Oracles / Risc Zero

Identity Solutions

Polygon ID

See site

Polygon ID has the following properties:

- Blockchain-based ID for decentralised and self-sovereign models
- Zero-knowledge native protocols for ultimate user privacy
- Scalable and private on-chain verification to boost decentralised apps and DeFi
- Open to existing standards and ecosystem development

It uses the Iden3 and Circom toolkit

In comparison with NFTs and VCs

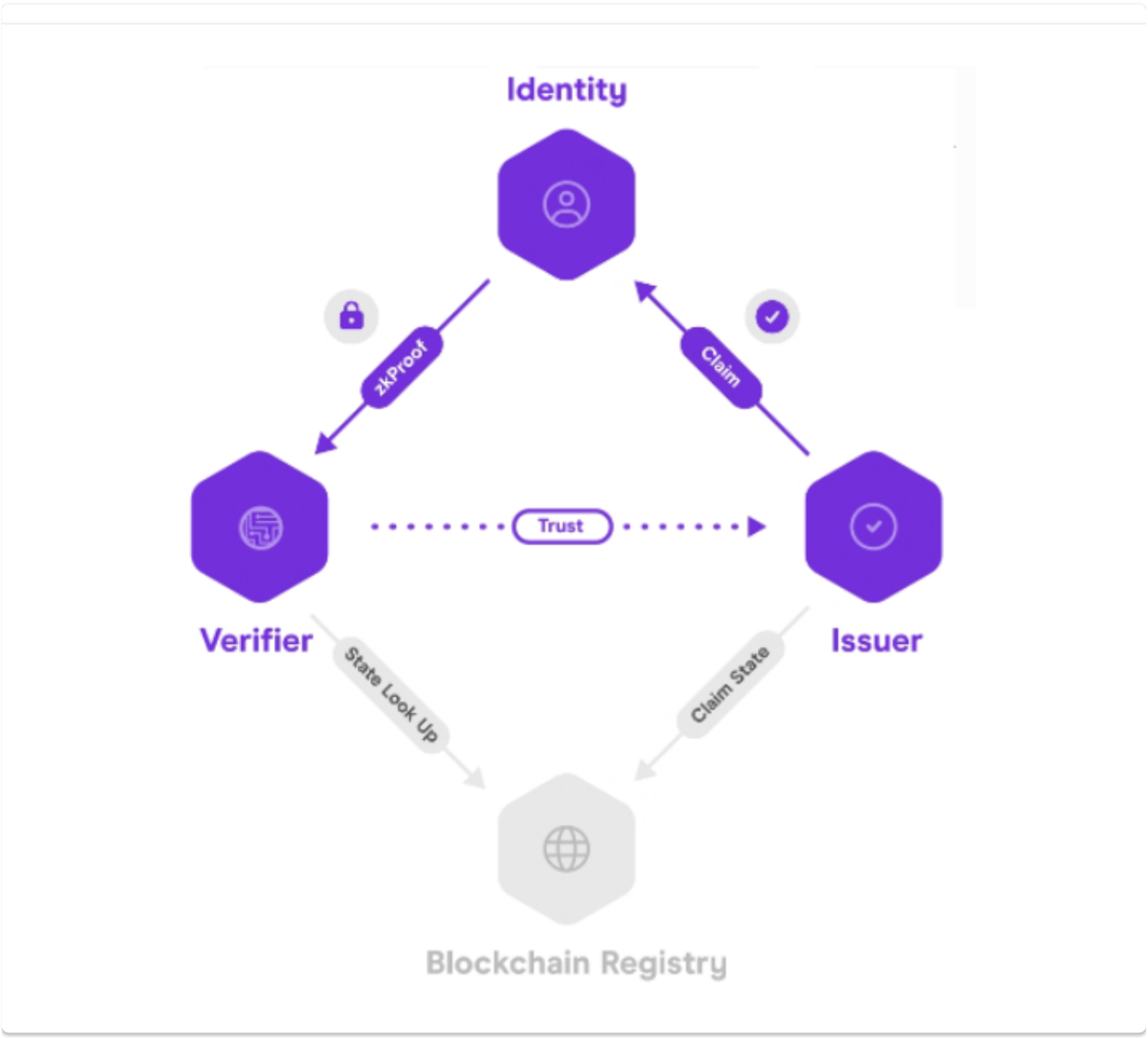
NFTs are not private, and have high minting costs.
While VCs offer some degree of privacy with selective disclosure and ZK add-on, their limitations are in the expressibility and composability, which are required for applications. Verifying VCs on-chain is prohibitively expensive.

There is an ID client toolkit to facilitate onboarding
On chain verification uses zkProof Request Language, which allows applications to specify which requested private attributes a user needs to prove.

Roles in Polygon ID

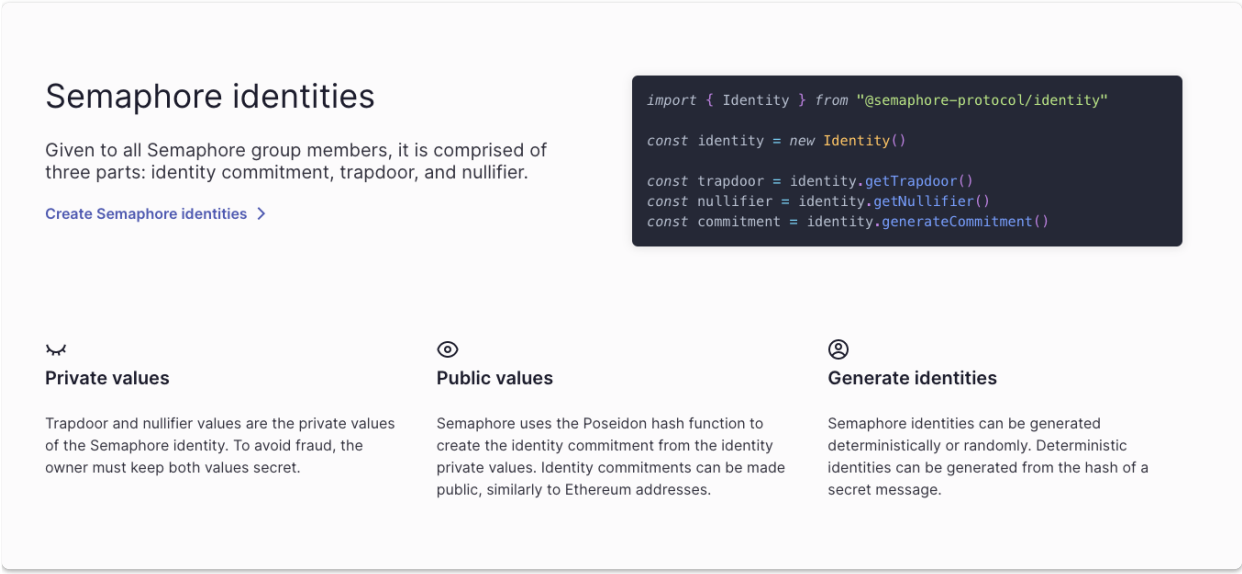
1. Identity Holder: An entity that holds claims in its Wallet. A claim is issued by an Issuer to the Holder. The Identity holder creates the zero-knowledge proofs of the claims issued and presents these proofs to the Verifier (which verifies the correctness and authenticity of the claim). A Holder is also called Prover as it needs to prove to the Verifier that the credential it holds is authentic and matches specific criteria.
2. Issuer: An entity (person, organisation, or thing) that issues claims to the Holders. Claims are cryptographically signed by the Issuer. Every claim comes from an Issuer.
3. Verifier: A Verifier verifies the claims presented by a Holder. It requests the Holder to send proof of the claim issued from an Issuer and on receiving the

zero-knowledge proofs from the Holder, verifies it. The verification process includes checking the veracity of the signature of the Issuer. The simplest real-world examples of a Verifies can be a recruiter that verifies your educational background or a voting platform that verifies your age.



Semaphore

Documentation
Repo
Semaphore is a zero-knowledge protocol that allows you to cast a signal (for example, a vote or endorsement) as a provable group member without revealing your identity.
Use cases include private voting, whistleblowing, anonymous DAOs and mixers.



Circuit

The Semaphore circuit is the heart of the protocol and consists of three parts:

- Proof of membership
- Nullifier hash
- Signal

They provide tools to create and verify proofs.

Setup

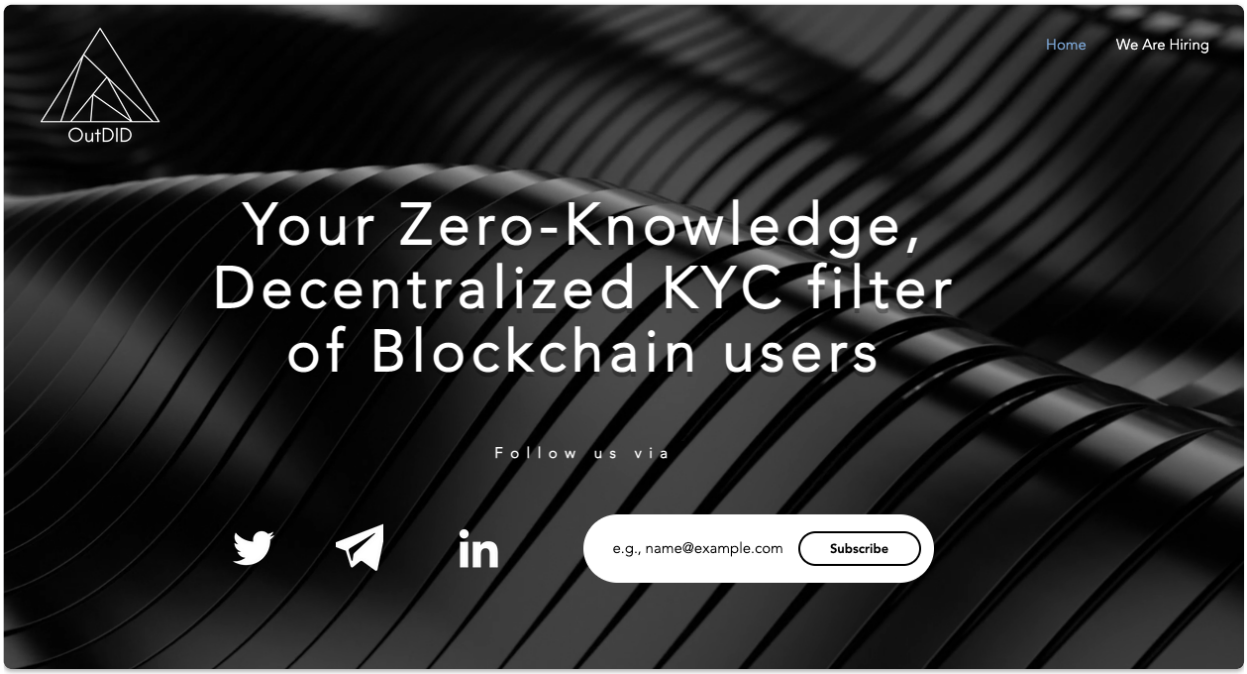
The semaphore CLI will set up a hardhat project

```
npx @semaphore-protocol/cli@latest create my-app
```

Example contract

OutDID.io

See docs



Providing KYC and Identity using Circom with for example passport data.

zCloak Network

zCloak Network provides Zero-Knowledge Proof as a Service based on the Polkadot Network

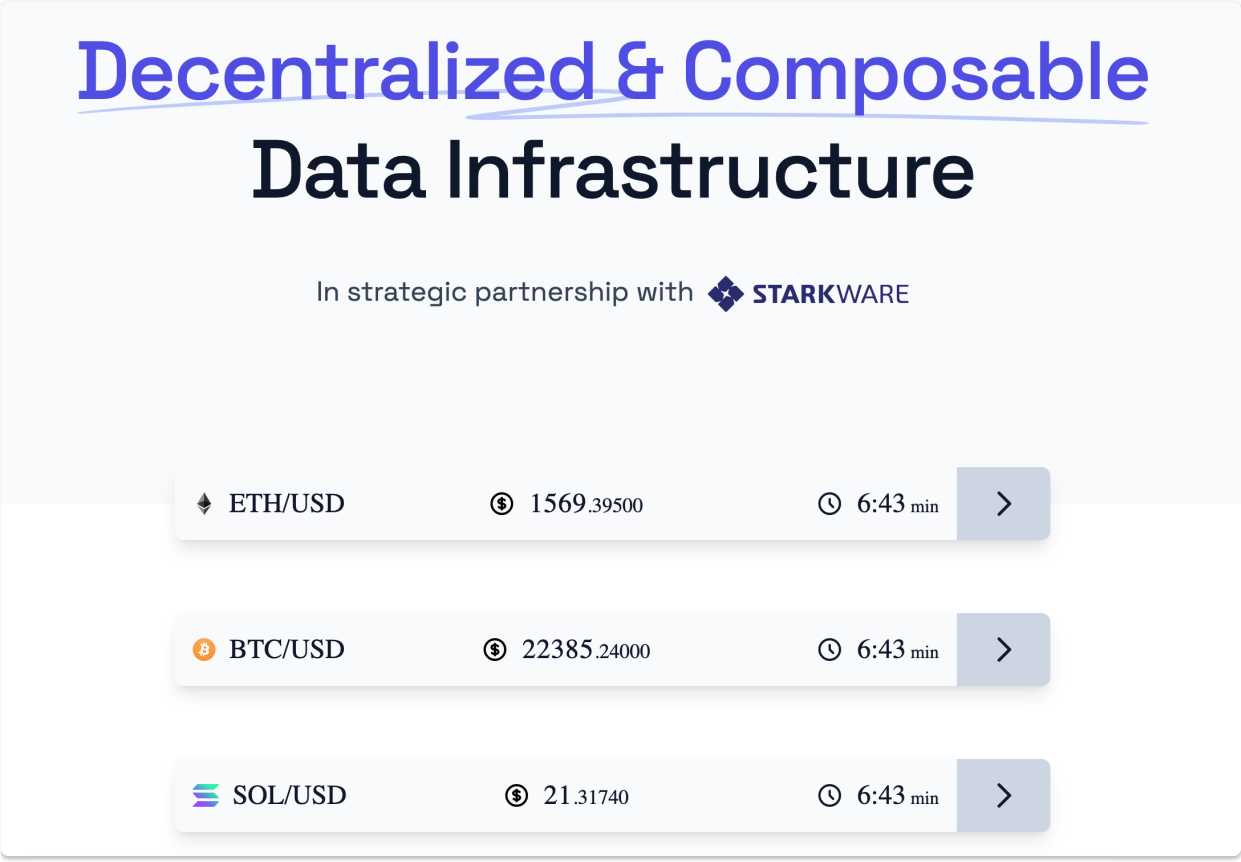
In the 'Cloaking Space' you control your own data and you can run all sorts of computation without sending your data away.
Note that the data stored in the Cloaking Space is not just some arbitrary data on your device, but they are attested by some credible network/organization to garantee its authenticity.
The type of computation can range from

- a regular state transition of a blockchain,
- a check of your income for a bank loan to
- an examination of your facial features to pass an airport checkpoint.

zCloak uses the Distaff VM
Distaff is a zero-knowledge virtual machine written in Rust.
For any program executed on Distaff VM, a STARK-based proof of execution is automatically generated. This proof can then be used by anyone to verify that a program was executed correctly.

Oracle Solutions

Empiric



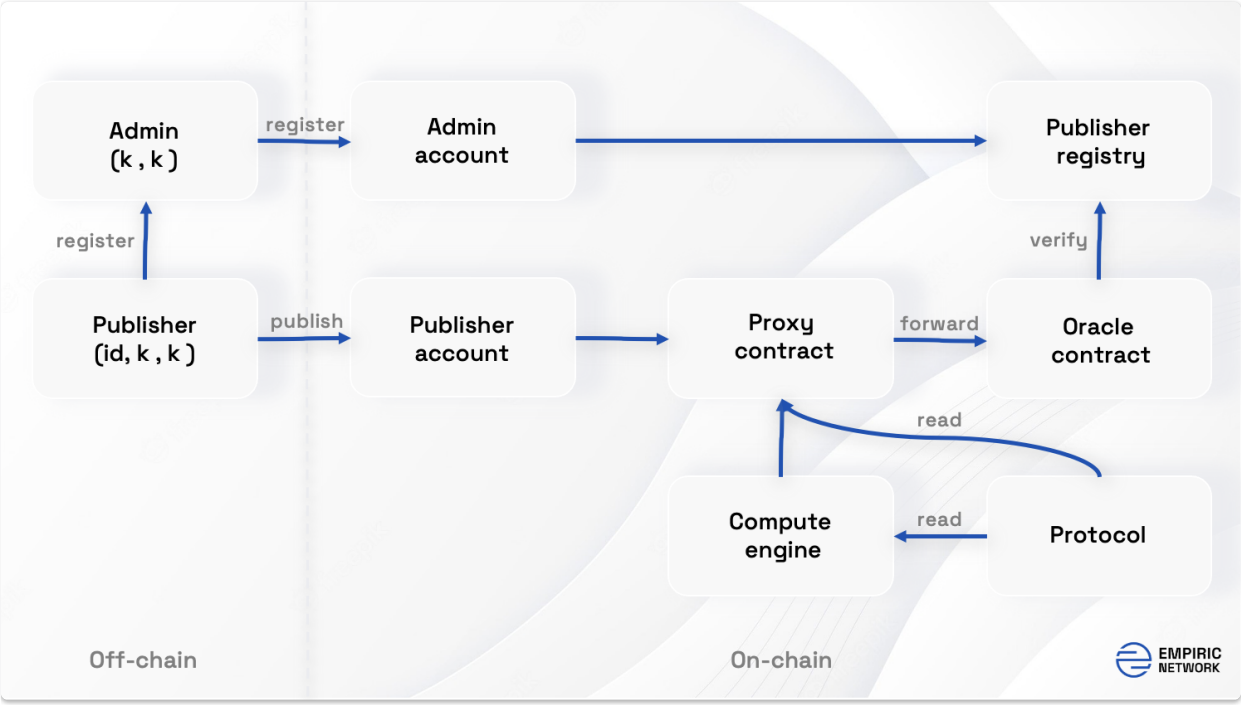
Empiric is available on Starknet and coming soon to Consensys zkEVM and already integrated with leading protocols such as ZKLend, Magnety, Serity, CurveZero, Canvas, and FujiDAO.

Assets supported on Starknet

- BTC/USD,
- BTC/EUR,
- ETH/USD,
- ETH/MXN
- SOL/USD,
- AVAX/USD,
- DOGE/USD,
- SHIB/USD,
- BNB/USD,
- ADA/USD,
- XRP/USD,
- MATIC/USD,

- USDT/USD,
- DAI/USD,
- USDC/USD,
- TUSD/USD,
- BUSD/USD,

Contracts



Code snippet

```
%lang starknet

from starkware.cairo.common.cairo_builtins import HashBuiltin

# Oracle Interface Definition
const EMPIRIC_ORACLE_ADDRESS =
0x012fadd18ec1a23a160cc46981400160fbf4a7a5eed156c4669e39807265bcd4
const KEY = 28556963469423460  # str_to_felt("eth/usd")
const AGGREGATION_MODE = 120282243752302  # str_to_felt("median")

@contract_interface
namespace IEmpiricOracle:
    func get_value(key : felt, aggregation_mode : felt) -> (
        value : felt,
        decimals : felt,
        last_updated_timestamp : felt,
        num_sources_aggregated : felt
    ):
```

```
    end
end

# Your function
@view
func my_func{
    syscall_ptr : felt*,
    pedersen_ptr : HashBuiltin*,
    range_check_ptr
}() -> ():
    let (eth_price,
        decimals,
        last_updated_timestamp,
        num_sources_aggregated) = IEmpiricOracle.get_value(
            EMPIRIC_ORACLE_ADDRESS, KEY, AGGREGATION_MODE
        )
    # Your smart contract logic!
    return ()
end
```

Empiric SDK

Installation
`pip install empiric-network`

Computational Feeds

You can compose and program data with Cairo, in order to get the right computed data for your protocol.
Empiric has designed compute engines that use the same raw market data underlying our price feeds, but calculate different metrics to produce feeds of processed data.
For example

- Realised Volatility see Docs
- Yield Curve , see Docs

DECO

Deco

- works with *modern TLS versions*
- requires *no trusted hardware*
- requires *no server-side modifications*

See paper

DECO Short for decentralized oracle, DECO is a new cryptographic protocol that enables a user (or oracle) to prove statements in zero knowledge about data obtained from HTTPS-enabled servers. DECO consequently allows private data from unmodified web servers to be relayed safely by oracle networks. (It does not allow data to be sent by a prover directly on chain.)
DECO has narrower capabilities than Town Crier, but unlike Town Crier, does not rely on a trusted execution environment.

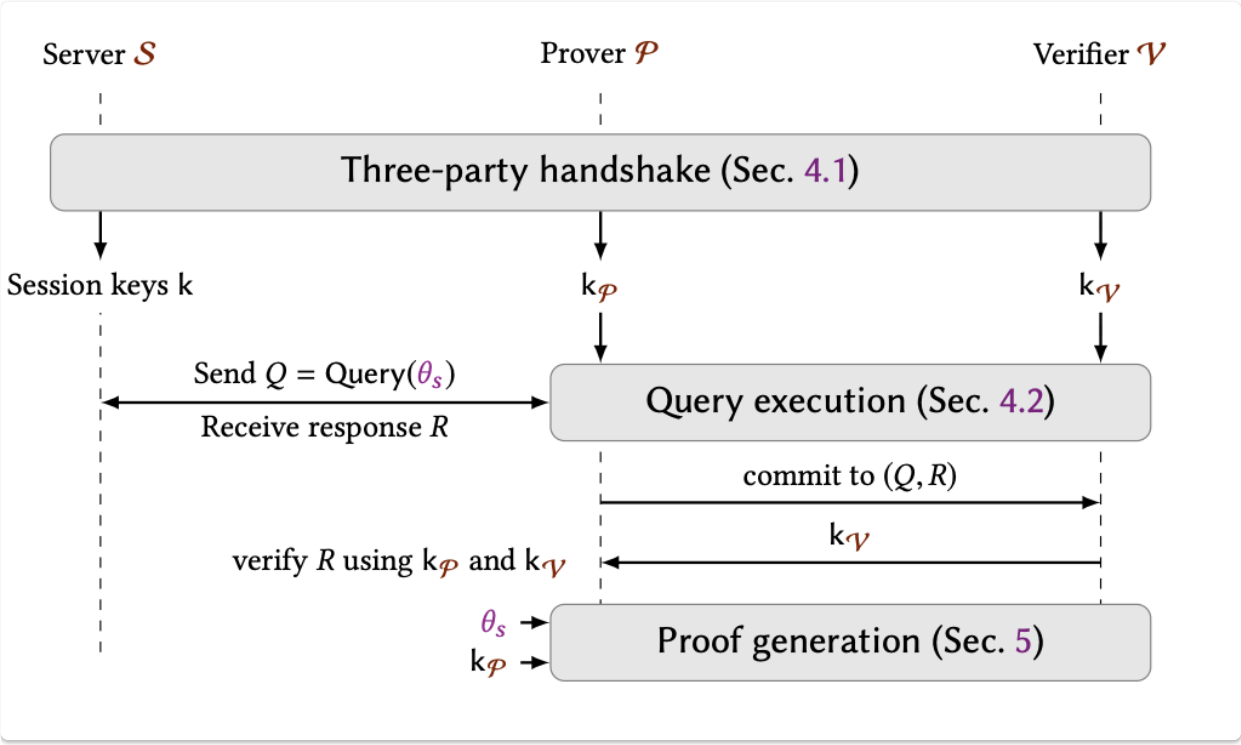
DECO can also be used to power the creation of decentralized identity (DID) protocols such as CanDID, where users can obtain and manage their own credentials, rather than relying on a centralized third party.
Such credentials are signed by entities called issuers that can authoritatively associate claims with users such as citizenship, occupation, college degrees, and more. DECO allows any existing web server to become an issuer and provides key-sharing management to back up accounts, as well as a privacy-preserving form of Sybil resistance based on definitive unique identifiers such as Social Security Numbers (SSNs).

ZKP solutions like DECO benefit not only the users, but also enable traditional institutions and data providers to monetize their proprietary and sensitive datasets in a confidential manner.
Instead of posting the data directly on-chain, only attestations derived from ZKPs proving facts about the data need to be published.
This opens up new markets for data providers, who can monetize existing datasets

and increase their revenue while ensuring zero data leakage. When combined with Chainlink Mixicles, privacy is extended beyond the input data executing an agreement to also include the terms of the agreement itself.

Process

Diagram from the white paper



Three party handshake

Essentially, P and V jointly act as a TLS client. They negotiate a shared session key with S in a secret shared form

Query Execution

Since the session keys are secret-shared, as noted, P and V execute an interactive protocol to construct a TLS message encrypting the query. P then sends the message to S as a standard TLS client.

P commits to the session data before receiving V's key share, making the commitment unforgeable. Then P can verify the integrity of the response, and prove statements about it.

Proof Generation

With unforgeable commitments, if P opens the commitment to the messages completely (i.e., reveals the encryption key) then V could easily verify the authenticity of the messages by checking MACs on the decryption.
Revealing the encryption key for the messages , however, would breach privacy: it would reveal all session data exchanged between P and S. In theory, P could instead

prove any statement about the messages in zero knowledge (i.e., without revealing the encryption key).
Generic zero-knowledge proof techniques, though, would be prohibitively expensive for many natural choices of the statement.

DECO instead introduces two techniques to support efficient proofs for a broad, general class of statement, namely selective opening of a TLS session transcript, see the white paper for details.

A web server itself could assume the role of an oracle, e.g., by simply signing data. However, server-facilitated oracles would not only incur a high adoption cost, but also put users at a disadvantage: the web server could impose arbitrary constraints on the oracle capability.

- Thus a single instance of DECO could enable anyone to become an oracle for any website
- Importantly, DECO does not require trusted hardware, unlike alternative approaches that could achieve a similar vision

DECO end-to-end performance depends on the available TLS ciphersuites, the size of private data, and the complexity of application specific proofs.
It takes about 13.77s to finish the protocol, which includes the time taken to generate unforgeable commitments (0.50s), to run the first stage of two-stage parsing (0.30s), and to generate zero-knowledge proofs (12.97s).

Roadmap

- Chainlink plans to do an initial PoC of DECO, with a focus on decentralized finance applications such as Mixicles.

For more details see their blog

Risc Zero

Introduction

The RISC Zero zkVM is an open-source, zero-knowledge virtual machine designed for constructing trustless, verifiable software applications.

Risc Zero's goal is to integrate existing programming languages and developer tools into the zero-knowledge realm. This is accomplished through a high-performance ZKP prover, which provides the performance allowance necessary to create a zero-knowledge virtual machine (zkVM) implementing a standard RISC-V instruction set.

In practical terms, this allows for smooth integration between "host" application code, written in high-level languages running natively on host processors (e.g., Rust on arm64 Mac), and "guest" code in the same language executing within the zkVM.

RISC Zero's zkVM implementation, based on the RISC-V architecture, executes code and generates a computational receipt. Anyone possessing a copy of this receipt can verify the program's execution and access its publicly shared outputs.

Bonsai

See lite paper
Bonsai is a versatile zero-knowledge proof network designed to facilitate the integration of ZK proofs for scalability, privacy, and other benefits across any chain, protocol, or application. This includes ETH, L1 Blockchains, Cosmos App Chains, L2 Rollups, and DApps.

The unique Bonsai network is built on three essential components, paving the way for the development of innovative applications in both blockchain and traditional application domains:

- A flexible zkVM that can operate any VM within a zero-knowledge/verifiable context
- A proving system that seamlessly connects to any smart contract or chain
- A universal rollup that distributes all computations proven on Bonsai across every chain

```
// Without Bonsai
contract simulation_normal {
 function some_really_hard_work() {
   // A large amount of gas heavy code
       // code ...
       // code ...
```

```
        // code ...
        // code ...
        // code ...
        //
 }
}
```

```
// With Bonsai
contract simulation_bonsai {
 function some_really_hard_work() {
   bonsai_proving_network.call(“some_really_hard_work”);
 }
}
```