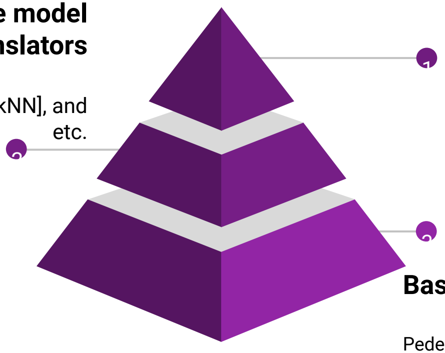
ZoKrates DevEase

Make ZoKrates powerful with Decision Tree Prediction and Pedersen Commitment

Overview

The ZoKrates DevEase aims to bring machine learning abilities, enhancing the existing libraries, and providing the up-to-date tutorial.



General toolbox[Future]

Representative model translators

decision trees, [kNN], and etc.

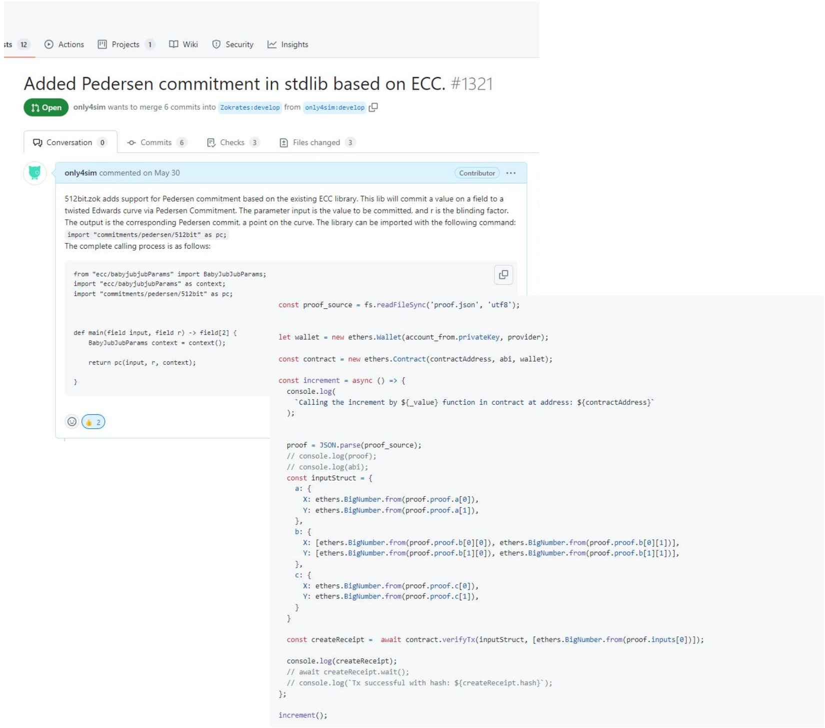
Base Libraries

Pedersen Commitment, ether.js, and etc.

Base libraries

01

ZoKrates DevEase builds frequently used libraries, such as Pedersen Commitment and ether.js call. Developers will be able to use these libraries to simplify the development of on-chain apps in the EVM ecosystem.



Added Pedersen commitment in stdlib based on ECC. #1321

only4sim wants to merge 6 commits into Zokrates:develop from only4sim:develop

only4sim commented on May 30

512bit.zok adds support for Pedersen commitment based on the existing ECC library. This lib will commit a value on a field to a twisted Edwards curve via Pedersen Commitment. The parameter input is the value to be committed, and r is the blinding factor. The output is the corresponding Pedersen commit, a point on the curve. The library can be imported with the following command: `import "commitments/pedersen/512bit" as pc;`

The complete calling process is as follows:

```
from "ecc/babyjubjubParams" import BabyJubJubParams;
import "ecc/babyjubjubParams" as context;
import "commitments/pedersen/512bit" as pc;


def main(field input, field r) -> field[2] {
    BabyJubJubParams context = context();

    return pc(input, r, context);

}
```

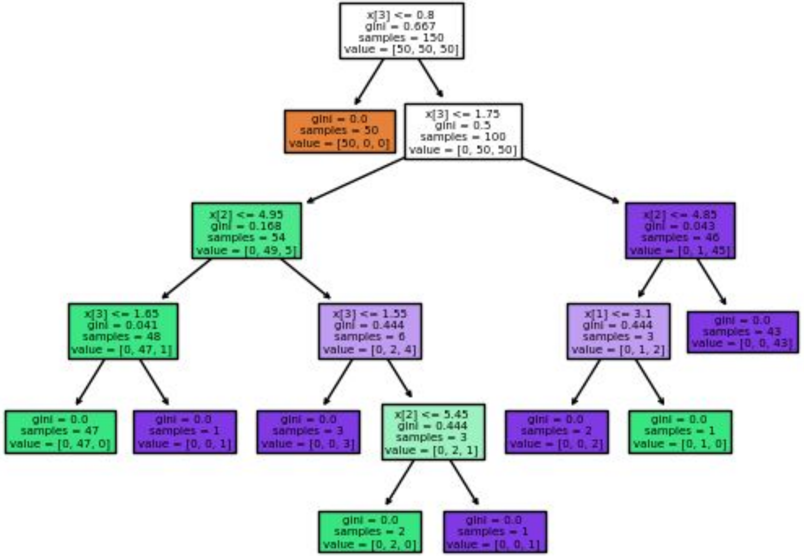
```
const proof_source = fs.readFileSync('proof.json', 'utf8');


let wallet = new ethers.Wallet(account_from.privateKey, provider);

const contract = new ethers.Contract(contractAddress, abi, wallet);

const increment = async () => {
  console.log(
    `Calling the increment by ${_value} function in contract at address: ${contractAddress}`
  );


  proof = JSON.parse(proof_source);
  // console.log(proof);
  // console.log(abi);
  const inputStruct = {
    a: {
      X: ethers.BigNumber.from(proof.proof.a[0]),
      Y: ethers.BigNumber.from(proof.proof.a[1]),
    },
    b: {
      X: [ethers.BigNumber.from(proof.proof.b[0][0]), ethers.BigNumber.from(proof.proof.b[0][1])],
      Y: [ethers.BigNumber.from(proof.proof.b[1][0]), ethers.BigNumber.from(proof.proof.b[1][1])],
    },
    c: {
      X: ethers.BigNumber.from(proof.proof.c[0]),
      Y: ethers.BigNumber.from(proof.proof.c[1]),
    }
  }

  const createReceipt =  await contract.verifyTx(inputStruct, [ethers.BigNumber.from(proof.inputs[0])]);

  console.log(createReceipt);
  // await createReceipt.wait();
  // console.log(`Tx successful with hash: ${createReceipt.hash}`);
};

increment();
```

Representative model translators

02

ZoKrates DevEase builds a translators for representative a ML model – decision tree, making it easier for developers to customize their own ML models. This phase was implemented with reference to keras2circom and ZK-DTP.

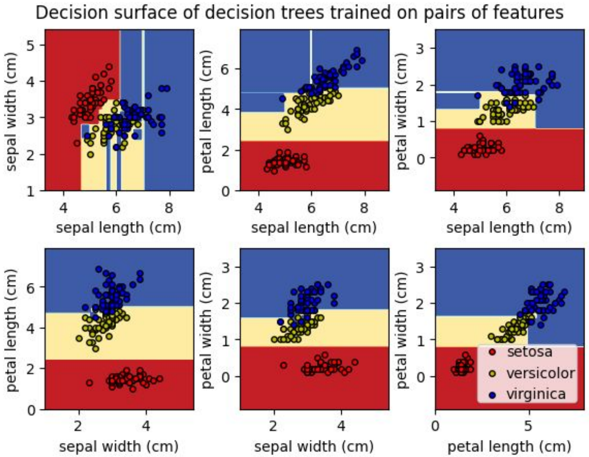
Decision tree trained on all the iris features

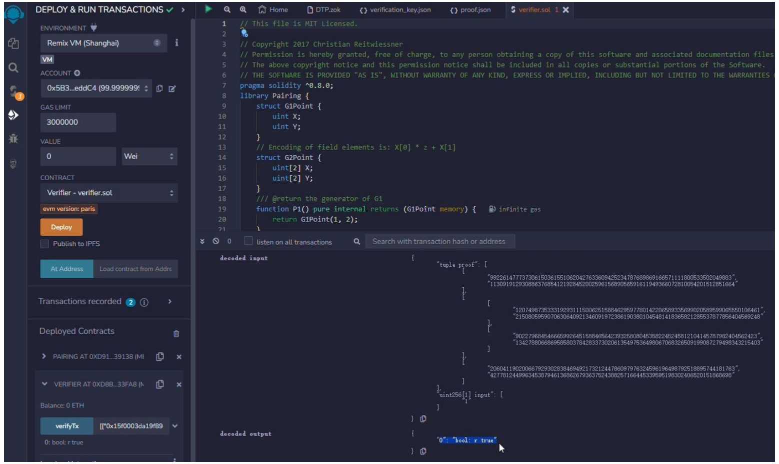


x[3] <= 0.8 gini = 0.667 samples = 150 value = [50, 50, 50]
gini = 0.0 samples = 50 value = [50, 0, 0]
x[3] <= 1.75 gini = 0.5 samples = 100 value = [0, 50, 50]
x[2] <= 4.95 gini = 0.168 samples = 54 value = [0, 49, 5]
x[2] <= 4.85 gini = 0.043 samples = 46 value = [0, 1, 45]
x[3] <= 1.65 gini = 0.041 samples = 48 value = [0, 47, 1]
x[3] <= 1.55 gini = 0.444 samples = 6 value = [0, 2, 4]
x[1] <= 3.1 gini = 0.444 samples = 3 value = [0, 1, 2]
gini = 0.0 samples = 43 value = [0, 0, 43]
gini = 0.0 samples = 47 value = [0, 47, 0]
gini = 0.0 samples = 1 value = [0, 0, 1]
gini = 0.0 samples = 3 value = [0, 0, 3]
x[2] <= 5.45 gini = 0.444 samples = 3 value = [0, 2, 1]
gini = 0.0 samples = 2 value = [0, 0, 2]
gini = 0.0 samples = 1 value = [0, 1, 0]
gini = 0.0 samples = 2 value = [0, 2, 0]
gini = 0.0 samples = 1 value = [0, 0, 1]

Representative model translators

02





ENVIRONMENT
Remix VM (Shanghai)
VM
ACCOUNT
0x5B3...eddC4 (99.999999!
GAS LIMIT
3000000
VALUE
0 Wei
CONTRACT
Verifier - verifier.sol
evm version: paris
☐ Publish to IPFS
Transactions recorded 2
Deployed Contracts
PAIRING AT 0XD91...39138 (ME
VERIFIER AT 0XD8B...33FA8 (M
Balance: 0 ETH
verifyTx [["0x15f0003da19f89
0: bool: r true

```
// This file is MIT Licensed.

// Copyright 2017 Christian Reitwiessner
// Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files
// The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES
pragma solidity ^0.8.0;
library Pairing {
    struct G1Point {
        uint X;
        uint Y;
    }
    // Encoding of field elements is: X[0] * z + X[1]
    struct G2Point {
        uint[2] X;
        uint[2] Y;
    }
    /// @return the generator of G1
    function P1() pure internal returns (G1Point memory) {    infinite gas
        return G1Point(1, 2);
```

```
decoded input       {
                        "tuple proof": [
                            [
                                "9922614777373061503615510620427633609425234787689869166571111800533502049883",
                                "11309191293088637685412192845200259615689056591611949366072810054201512851664"
                            ],
                            [
                                [
                                    "12074987353331929311150062515884629597780142206589335699020589599065550106461",
                                    "21508059590706306409213460919723861903801045481418365821285537877856404569248"
                                ],
                                [
                                    "9022796845466659926451588465642393258080453582245245812104145787982404562423",
                                    "13427880668695858037842833730206135497536498067068326509199087279498343215403"
                                ]
                            ],
                            [
                                "20604119020066792930283846949217321244786097976324596196498792518895744181763",
                                "4277812449963453879461368626793637524388257166445339595198302406520151868698"
                            ]
                        ],
                        "uint256[1] input": [
                            "1"
                        ]
                    }

decoded output      {
                        "0": "bool: r true"
                    }
```

Project Vision for Future

Generalized toolbox enabling developers to directly migrate Dapps and trained Python models to ZoKrates.

Target audience

Financial, commercial, health, medical, game developers

01 | Financial firms, on-chain authentication, KYC

02 | Commercial users, provable models

03 | Certifiable health testing, information sharing

04 | Medical Image Mining

05 | On-chain intelligent games

Final Delivery Targets

Base libraries

10+

ether.js call, argMax, averagePooling2D, conv1D, and more.

Representative ML model templates/translators

5

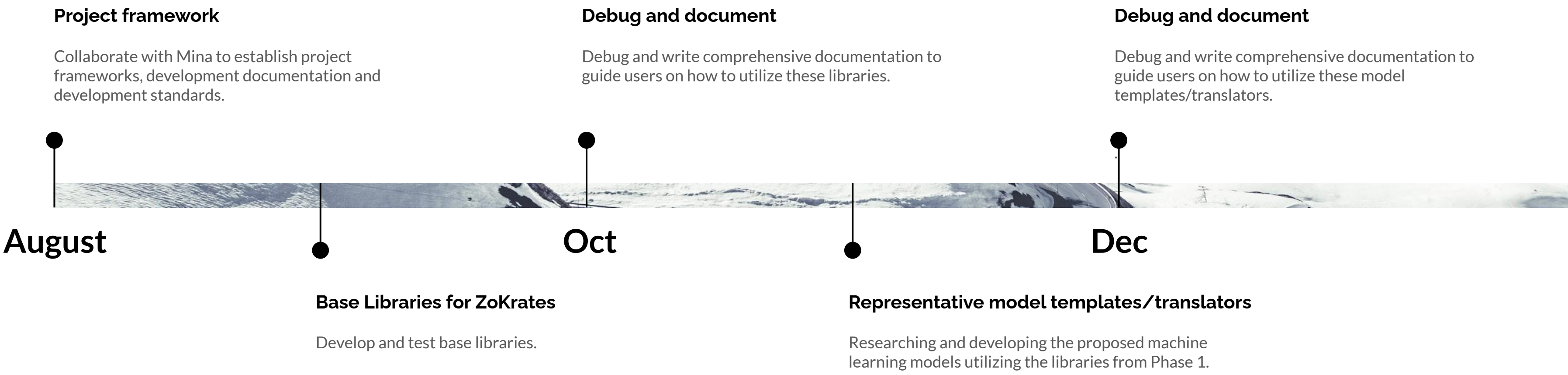
decisionTree, linearRegression, kNN, and more.

Documentation, tutorials and videos

25+

Development documentation, usage tutorials, instructional videos.

Vision



August

Project framework

Collaborate with Mina to establish project frameworks, development documentation and development standards.

Base Libraries for ZoKrates

Develop and test base libraries.

Oct

Debug and document

Debug and write comprehensive documentation to guide users on how to utilize these libraries.

Representative model templates/translators

Researching and developing the proposed machine learning models utilizing the libraries from Phase 1.

Dec

Debug and document

Debug and write comprehensive documentation to guide users on how to utilize these model templates/translators.

Demo

- Decision Tree Prediction(ZoKrates-DTP)
- Pedersen Commitment Library

Contributor

Li @only4sim

- Marie Curie Fellow
- Currently pursuing a PhD related to database and zero-knowledge proofs

Thank you.