Create and verify aProof

Usegnark/backend

Once thecircuit iscompiled , you can run the three algorithms of a zk-SNARK back end:

- Setup
- Prove
- Verify

note Supported zk-SNARK backends are undergnark/backend .gnark currently implementsGroth16 and an experimental version ofPlonK . Use a zk-SNARK back end * Groth16 * PlonK

// 1. One time setup pk , vk , err := groth16 . Setup (cs)

// 2. Proof creation proof , err := groth16 . Prove (cs , pk , witness)

// 3. Proof verification err := groth16 . Verify (proof , vk , publicWitness) // 1. One time setup publicData ,

_

:= plonk . Setup (cs ,

...)

// WIP

// 2. Proof creation proof , err := plonk . Prove (r1cs , publicData , witness)

// 3. Proof verification err := plonk . Verify (proof , publicData , publicWitness)

Construct the witness

Within a Go process, re-use the circuit data structure to construct the witness.

type Circuit struct

{ X frontend . Variable Y frontend . Variable gnark:",public" }

assignment :=

& Circuit { X :

3 , Y :

35 , } witness ,

_

:= frontend . NewWitness (assignment , ecc . BN254) // use the witness directly in zk-SNARK backend APIs groth16 . Prove (cs , pk , witness) // test file --> assert.ProverSucceeded(cs, &witness) tip If witness is not built within the same process, or in another programming language, refer toSerialize .

Verify aProof

on Ethereum

Onecc.BN254 +Groth16 ,gnark can export thegroth16.VerifyingKey as a solidity smart contract.

Refer tothe code example andend-to-end integration test using ageth simulated blockchain.

// 1. Compile (Groth16 + BN254) cs , err := frontend . Compile (ecc . BN254 , r1cs . NewBuilder ,

& myCircuit)

// 2. Setup pk , vk , err := groth16 . Setup (cs)

// 3. Write solidity smart contract into a file err = vk . ExportSolidity (f Edit this page Last updatedonMar 2, 2023