**Zokrates Installation**

curl -LSfs get.zokrat.es | sh

Create root.zok file as,

def main(private field a, field b):

assert(a \* a == b)

return

Compile the Program

zokrates compile -i root.zok

Perform the Setup Phase

zokrates compile -i root.zok

Execute the Program

zokrates compute-witness -a 337 113569

Generate the Proof of computation

zokrates generate-proof

Export the Solidity Verifier

zokrates export-verifier

Verify natively

zokrates verify

The Verifier.sol file generated has the verifyTx method as,

function verifyTx(

Proof memory proof, uint[1] memory input

) public view returns (bool r) {

uint[] memory inputValues = new uint[](1);

for(uint i = 0; i < input.length; i++){

inputValues[i] = input[i];

}

if (verify(inputValues, proof) == 0) {

return true;

} else {

return false;

}

}

**Refactored code**

function verify(uint256[] memory input, Proof memory proof)

internal

view

returns (uint256)

{

uint256 snark\_scalar\_field = 21888242871839275222246405745257275088548364400416034343698204186575808495617;

VerifyingKey memory vk = verifyingKey();

require(input.length + 1 == vk.gamma\_abc.length);

// Compute the linear combination vk\_x

Pairing.G1Point memory vk\_x = Pairing.G1Point(0, 0);

for (uint256 i = 0; i < input.length; i++) {

require(input[i] < snark\_scalar\_field);

vk\_x = Pairing.addition(

vk\_x,

Pairing.scalar\_mul(vk.gamma\_abc[i + 1], input[i])

);

}

vk\_x = Pairing.addition(vk\_x, vk.gamma\_abc[0]);

if (

!Pairing.pairingProd4(

proof.a,

proof.b,

Pairing.negate(vk\_x),

vk.gamma,

Pairing.negate(proof.c),

vk.delta,

Pairing.negate(vk.alpha),

vk.beta

)

) return 1;

return 0;

}

function verifyTx(

uint256[2] memory a,

uint256[2][2] memory b,

uint256[2] memory c,

uint256[2] memory input

) public view returns (bool r) {

Proof memory proof;

proof.a = Pairing.G1Point(a[0], a[1]);

proof.b = Pairing.G2Point([b[0][0], b[0][1]], [b[1][0], b[1][1]]);

proof.c = Pairing.G1Point(c[0], c[1]);

uint256[] memory inputValues = new uint256[](2);

for (uint256 i = 0; i < input.length; i++) {

inputValues[i] = input[i];

}

if (verify(inputValues, proof) == 0) {

return true;

} else {

return false;

}

}