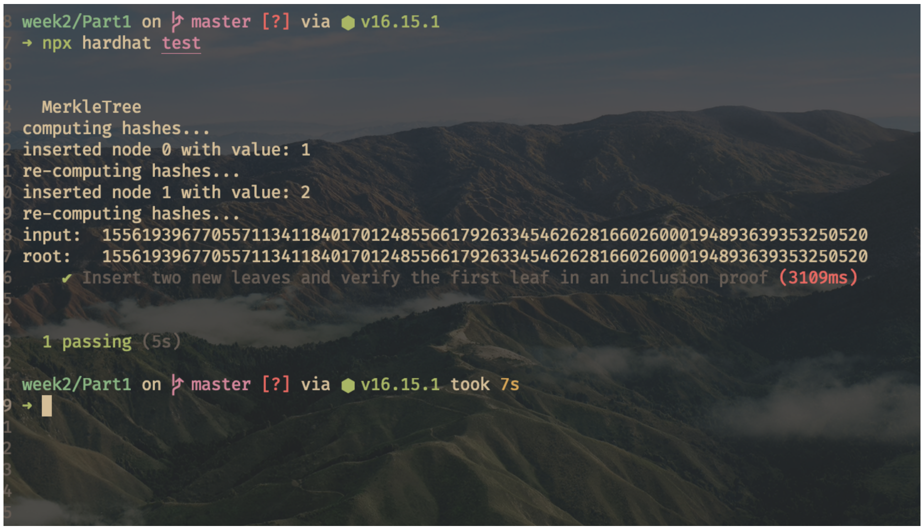
ZKU week 2 solutions

siosw#4738 - hi@simonoswald.xyz

part 1

1. SHA256 gas cost: low proof generation: slow efficiency & proof size: seem to be really fast / small **MiMC** gas cost: around 2x SHA256, higher than Poseidon proof generation: turbo PLONK can generate proofs for MiMC very fast **Poseidon** gas cost: slightly less than MiMC proof generation: much faster than SHA256 **Pedersen** proof generation: like MiMC PLONK really excells here

3.4



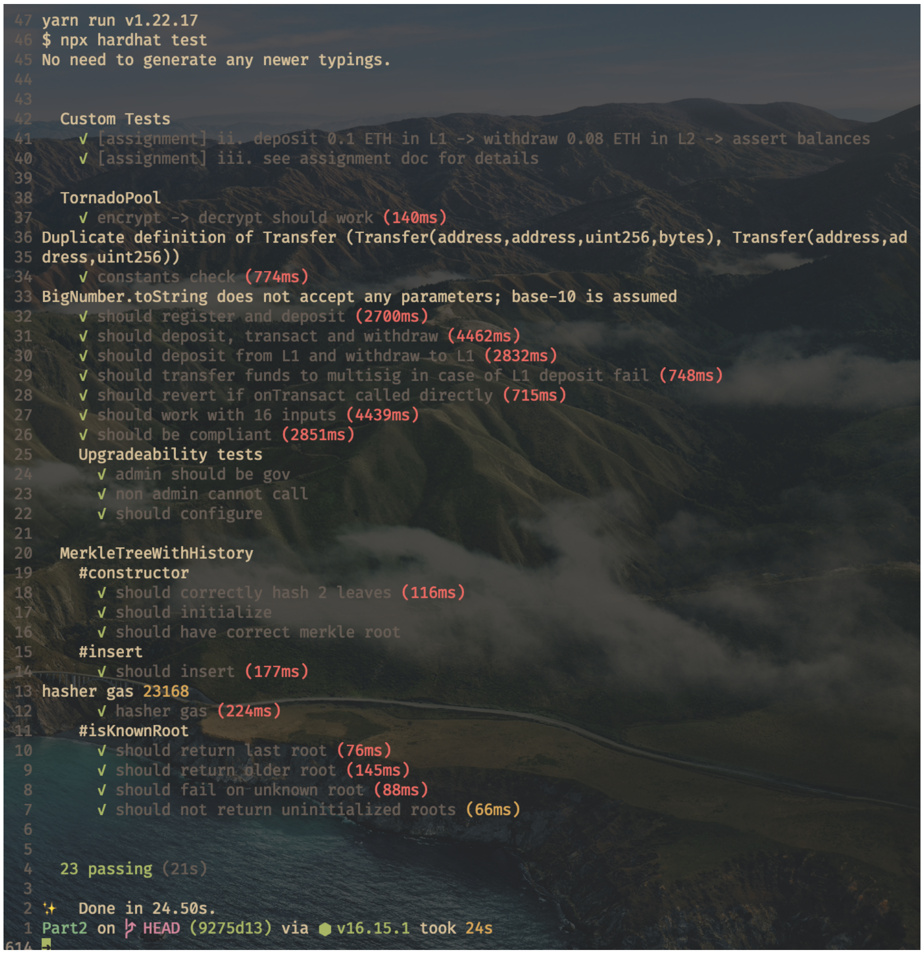
```
week2/Part1 on  master [?] via ⬢ v16.15.1
➜ npx hardhat test


  MerkleTree
computing hashes...
inserted node 0 with value: 1
re-computing hashes...
inserted node 1 with value: 2
re-computing hashes...
input:  15561939677055711341184017012485566179263345462628166026000194893639353250520
root:   15561939677055711341184017012485566179263345462628166026000194893639353250520
    ✔ Insert two new leaves and verify the first leaf in an inclusion proof (3109ms)


  1 passing (5s)

week2/Part1 on  master [?] via ⬢ v16.15.1 took 7s
➜ 
```

part 2

1. Nova allows users to transfer funds within the privacy pool, allows custom amounts, and reduces gas fees by moving the pool to L2.

2. Relayers pay the gas cost for users when bridging funds from L1 to L2. This is needed because paying for gas from the users wallet would compromise privacy.

3.1



```
yarn run v1.22.17
$ npx hardhat test
No need to generate any newer typings.


  Custom Tests
    √ [assignment] ii. deposit 0.1 ETH in L1 -> withdraw 0.08 ETH in L2 -> assert balances
    √ [assignment] iii. see assignment doc for details

  TornadoPool
    √ encrypt -> decrypt should work (140ms)
Duplicate definition of Transfer (Transfer(address,address,uint256,bytes), Transfer(address,ad
dress,uint256))
    √ constants check (774ms)
BigNumber.toString does not accept any parameters; base-10 is assumed
    √ should register and deposit (2700ms)
    √ should deposit, transact and withdraw (4462ms)
    √ should deposit from L1 and withdraw to L1 (2832ms)
    √ should transfer funds to multisig in case of L1 deposit fail (748ms)
    √ should revert if onTransact called directly (715ms)
    √ should work with 16 inputs (4439ms)
    √ should be compliant (2851ms)
    Upgradeability tests
      √ admin should be gov
      √ non admin cannot call
      √ should configure

  MerkleTreeWithHistory
    #constructor
      √ should correctly hash 2 leaves (116ms)
      √ should initialize
      √ should have correct merkle root
    #insert
      √ should insert (177ms)
hasher gas 23168
      √ hasher gas (224ms)
    #isKnownRoot
      √ should return last root (76ms)
      √ should return older root (145ms)
      √ should fail on unknown root (88ms)
      √ should not return uninitialized roots (66ms)


  23 passing (21s)

✨  Done in 24.50s.
Part2 on HEAD (9275d13) via ⬢ v16.15.1 took 24s
```

part 3

1. A Semaphore is a means of anonymous signaling. A set of users is approved by a smart contract, now any member of the set can publish and thereby endorse unique strings without revealing their identity. The only public information is that the string was published by some user within the approved set.

2. The external nullifier can also be understood as a topic. While users remain anonymous, the semaphore guarantees that every user can only publish a single commitment per topic. To achieve this the hash of a private identity nullifier, the merkle proof of membership, and the external nullifier must match the public nullifier hash. Every nullifier hash can only be published once.

3. Claiming airdrops anonymously by publishing a commitment. A whistleblowing mechanism, sources can prove they are reliable without revealing their identity (of course this must be setup beforehand).