Week 2 assignment

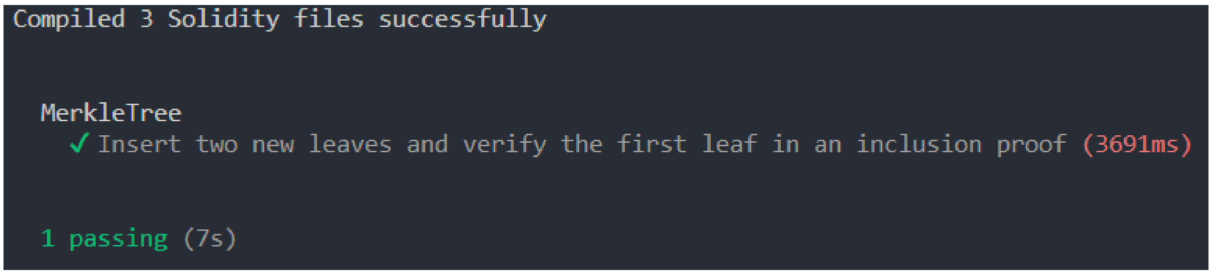
Part 1: hashes and Merkle tree

1. Gas cost: SHA256 < Poseidon < MiMC ≈ Pedersen
   MiMC and Poseidon have similar gas costs for both initialization and insertion, but MiMC is significantly more expensive when it comes to verification of circuit constraints.

   Capacity: SHA256 < Pedersen ≈ MiMC < Poseidon
   Source: article1

   Proof generation efficiency: Poseidon < Pedersen ≈ MiMC < SHA-256
   As shown in this article, prover time for SHA-256 is lower than the time for MiMC. Pedersen gives results similar to MiMC, varying in performance depending on the SNARK used. Poseidon is about 30 times slower than SHA256, which makes it the slowest of the four.
   Additional source: article

   Proof size: SHA256 < MiMC < Pedersen < Poseidon
   Sources: article1, article2, article3

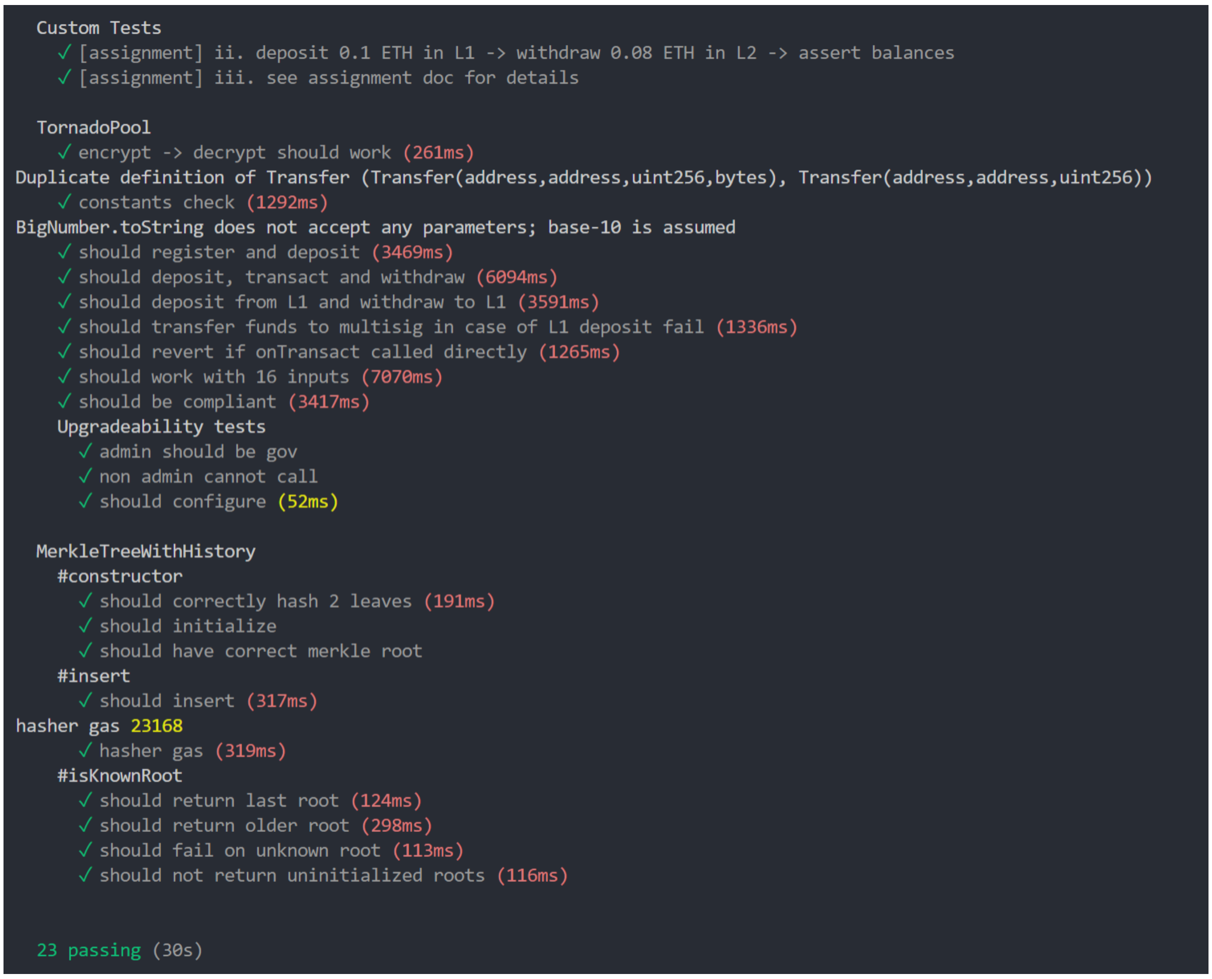
2. 

```
Compiled 3 Solidity files successfully


  MerkleTree
    ✓ Insert two new leaves and verify the first leaf in an inclusion proof (3691ms)


  1 passing (7s)
```

3.

Part 2: Tornado cash

1. Tornado Nova allows users to choose custom amounts of deposit (even if it's suggested to use standard amounts to blend with the crowd) and allows transfer of funds between users without leaving the pool.
2. Relayers are accounts that pay gas fees on behalf of another user when they are withdrawing. If a user had to pay gas fees when withdrawing, they would need to already have some ETH in their wallet, which can't be provided in a fully anonymous way (direct deposit from another wallet or via a centralized exchange). Relayers solve this problem in exchange for a fee for their service, so they help maintain the tornado ecosystem allowing privacy in withdrawals.
3.



```
  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 (261ms)
Duplicate definition of Transfer (Transfer(address,address,uint256,bytes), Transfer(address,address,uint256))
    ✓ constants check (1292ms)
BigNumber.toString does not accept any parameters; base-10 is assumed
    ✓ should register and deposit (3469ms)
    ✓ should deposit, transact and withdraw (6094ms)
    ✓ should deposit from L1 and withdraw to L1 (3591ms)
    ✓ should transfer funds to multisig in case of L1 deposit fail (1336ms)
    ✓ should revert if onTransact called directly (1265ms)
    ✓ should work with 16 inputs (7070ms)
    ✓ should be compliant (3417ms)
    Upgradeability tests
      ✓ admin should be gov
      ✓ non admin cannot call
      ✓ should configure (52ms)

  MerkleTreeWithHistory
    #constructor
      ✓ should correctly hash 2 leaves (191ms)
      ✓ should initialize
      ✓ should have correct merkle root
    #insert
      ✓ should insert (317ms)
hasher gas 23168
      ✓ hasher gas (319ms)
    #isKnownRoot
      ✓ should return last root (124ms)
      ✓ should return older root (298ms)
      ✓ should fail on unknown root (113ms)
      ✓ should not return uninitialized roots (116ms)


  23 passing (30s)
```

Part 3: Semaphore

1. Semaphore is a system that allows the user to get an authorization for an action and execute it, without revealing any information about himself. The protocol allows to generate a proof off chain and then allows to broadcast a signal on chain with the proof of verification.
2. To prevent double signaling, semaphore stores a hash of the signal, so it can only be used once.
3. Authentication for web3 social dapps like lens protocol, paywall contents or build anonymous credit score.