ZKU - Assignment 2


```
Email - stanleychiu@protonmail.com
Discord - HKerStanley#4125
GitHub - https://github.com/HKerStanley/zk-uni (./asset/week_2)
```


Question 1

1. As blockchain is essentially a state machine, any state change is a process of taking inputs, execute the state transition function which can be a simple transaction or smart contract function, that miners or validators will verify the validity. Once majority of them reach an agreement, the chain state will be updated and permanently stored. Anyone should be able to reach the current state of the blockchain by re-executing all transactions. Such re-execution nature meaning that all input data must be public and backup for validation, which bring up privacy concern, hard to scale and limitation on efficiency in both storage and computation. ZK based verification solve above mentioned problem as the input data can be remain private. Also greatly improve the performance as the zero knowledge proof is the only thing needed to be verify

2. A ZK VM is a circuit that takes part of the input of a program, so that a prover can be able to show that with given a set of inputs, they have correctly executed the program code on said inputs.

	StarkVM	zkSync	Miden
ZK Proof	STARK	SNARK	STARK
Data Availability	ZK-Rollup and Validium	On-chain	ZK Rollup
Language Support	Cairo	Solidity, Zinc	Miden Assembly
Privacy	Complete privacy	Transparent	Selective
Scalability	scalable by off-chain prover and on-chain verifier	2000+ TPS	1000+ TPS

Question 2

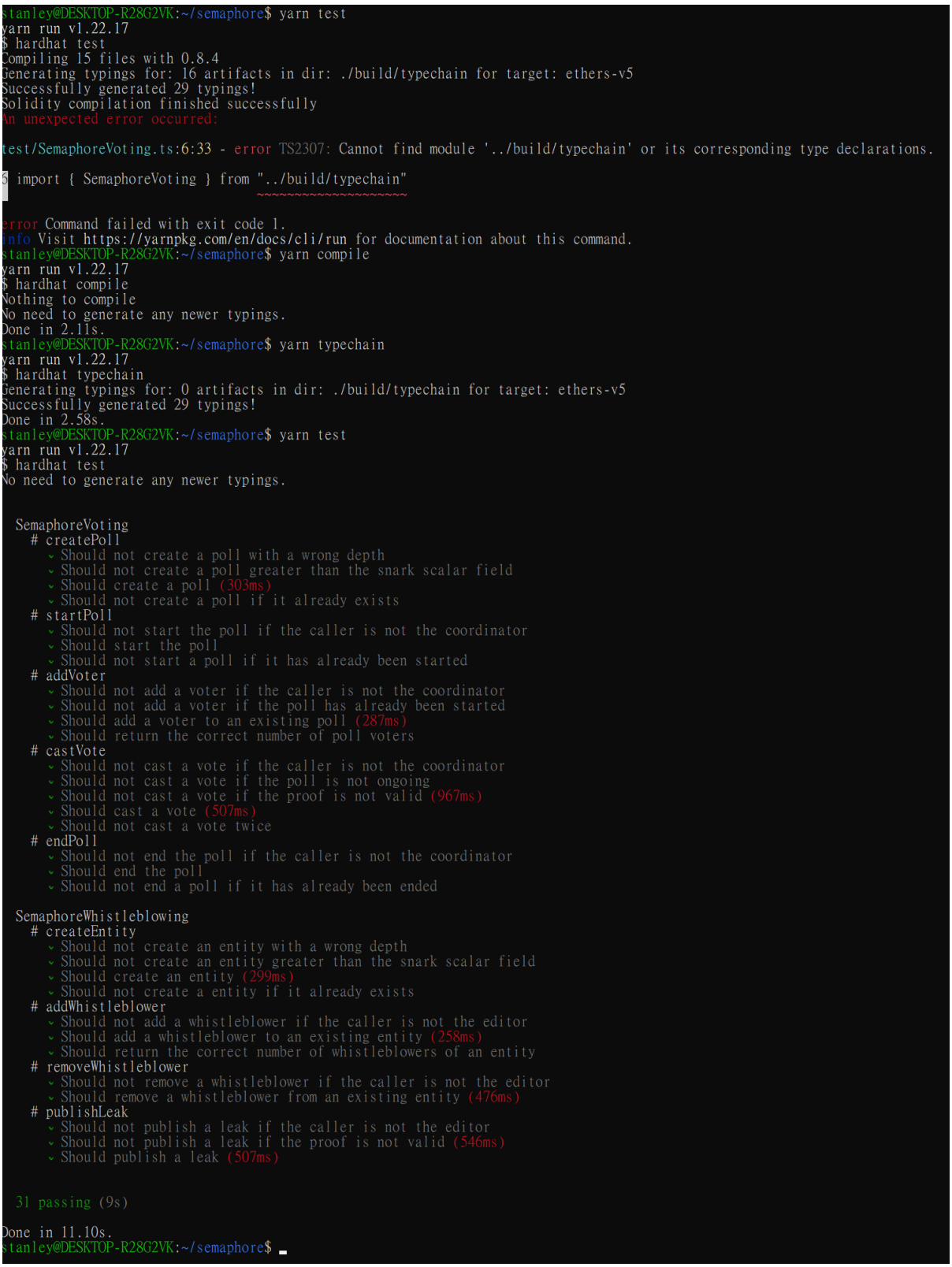
1. By definition, Semaphore is a system which allows any Ethereum user to signal their endorsement of an arbitrary string, revealing only that they have been previously approved to do so, and not

their specific identity.

Users register their identity with hash of public key and random secrets, which stores in a Merkle Tree for later verification. When broadcasting signal, users will provide zero-knowledge proof of their membership and theres an public nullifier to check if the signal has been broadcasted before.

Example of applications can be messaging application, digital document signing and POAP (Proof-Of-Attendance Protocol)

2. 1. Screenshot of all test passed on main branch, commit `3bce72f` is failing `addWhistleblower` and `removeWhistleblower`



2. Commented version of `semaphore.circom` : https://github.com/HKerStanley/zk-uni/blob/main/asset/week_2/semaphore_with_comment.circom

3. Authenticated on Elefria

Autheticated!

Elefria

Account: 0xa9D224c8d81325EBbb16EF84b68Bb6E328BF73aE

Connect Wallet

Login

Register

Logout

Restore

1. User can lost or expose their secret, just like how they lost or expose their wallet seedphrase or private key. Its is also possible for users to run out of gas, as register and login are all transactions on the blockchain.

2. I think Elefria can support auto-download of the user secret which encrypted with user's private key, so that the secret key saving process can be automated and prevent human error.

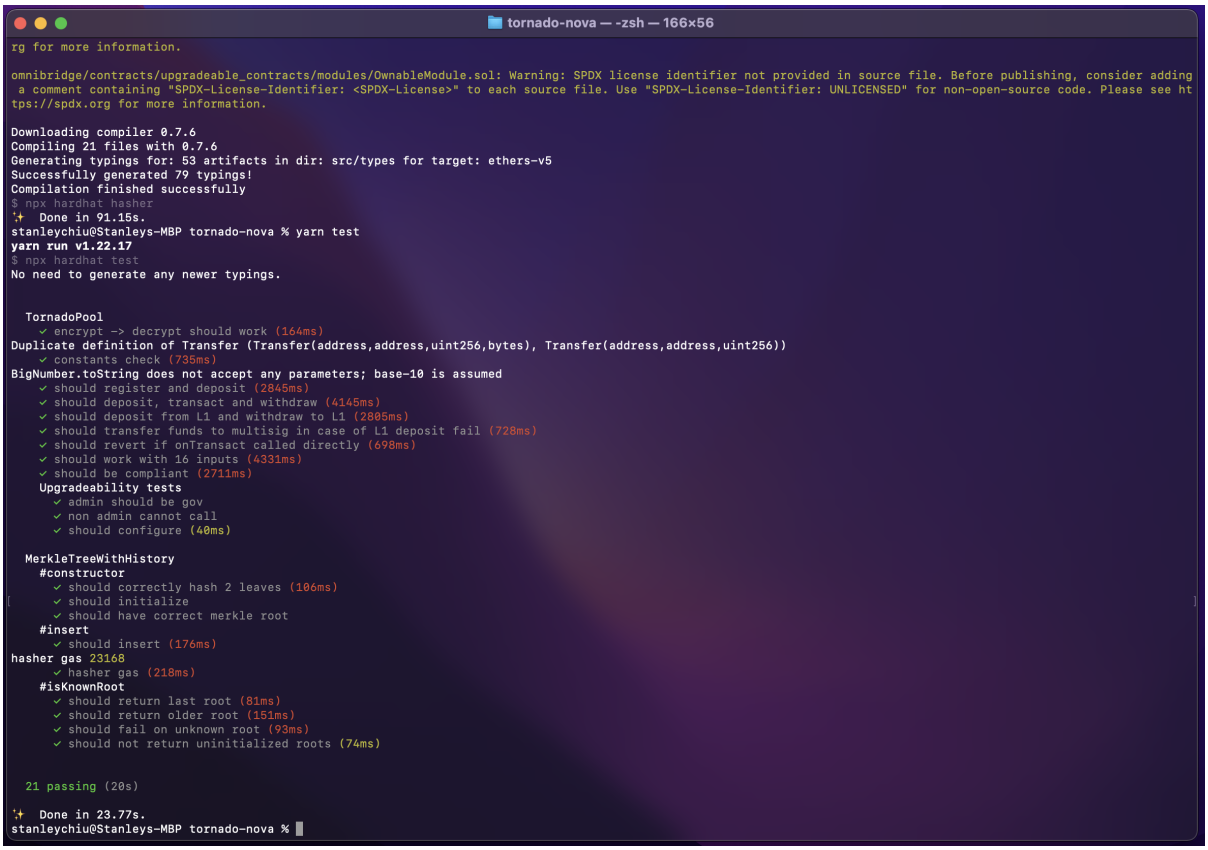
Question 3

1. For `tornado-trees` it is simply an Merkle Tree implementation, which I believe is how Tornado Cash used to verify if a deposit exist and if a withdrawal is valid. But for `tornado-nova` it also act as a mixer, which is an improvement of the prior version because a mixer support users to deposit and withdraw custom amount of fund.

2.
    1. `TreeUpdateArgsHasher.circom` is a circuit to create a SHA256 hash for the smart contract process and validation, which should be the `argsHash` in `TornadoTrees.sol` function `updateWithdrawalTree`. For a withdrawal tree update we need to provide the previous Merkle root, the new Merkle root, the path to the inserted subtree and data of the withdrawal event(block number, address of withdrawal, and the transaction has). These data are passed to the circuit to generate a snark proof, and in the smart contract it will in fact perform the same process, then verify if the samrt contract inputs lead to the same result as the snark proof. If its valid, the smart contract will store the updated Merkle root and complete an update. In the process the smart contract also emit the update event data so it is easier to reconstruct or verify the state of the Merkle Tree.

    2. Snark friendly hash function like Poseidon is expensive on-chain. To optimize the gas fee of a transaction, the process needs to be more dependent to the snark side. Computing SHA256 hash for snark is really heavy that requires a special machine to build the circuit, but once the circuit is built generating a proof with SHA256 hash is managable. This is also a good thing as the difficulty to setup give some level of protection to the protocol.

3.
    1. Theres an error when I run with Windows WSL2 so I switch to use my Mac and all test passed.



2. I worte the test script in `custom.text.js` and copied it to this assignment folder: https://github.com/HKerStanley/zk-uni/blob/main/asset/week_2/custom.test.js

```
stanleychiu@Stanleys-MBP tornado-nova % yarn test
yarn run v1.22.17
$ npx hardhat test
No need to generate any newer typings.


  Custom Test
Duplicate definition of Transfer (Transfer(address,address,uint256,bytes), Transfer(address,address,uint256))
Gas Estimate:  23168
BigNumber.toString does not accept any parameters; base-10 is assumed
    ✓ should deposit from L1 and withdraw to L2 (5115ms)

  TornadoPool
    ✓ encrypt -> decrypt should work
Duplicate definition of Transfer (Transfer(address,address,uint256,bytes), Transfer(address,address,uint256))
    ✓ constants check (407ms)
    ✓ should register and deposit (1339ms)
    ✓ should deposit, transact and withdraw (4020ms)
    ✓ should deposit from L1 and withdraw to L1 (2722ms)
    ✓ should transfer funds to multisig in case of L1 deposit fail (718ms)
    ✓ should revert if onTransact called directly (710ms)
    ✓ should work with 16 inputs (3799ms)
    ✓ should be compliant (2675ms)
    Upgradeability tests
      ✓ admin should be gov
      ✓ non admin cannot call
      ✓ should configure (51ms)

  MerkleTreeWithHistory
    #constructor
      ✓ should correctly hash 2 leaves (125ms)
      ✓ should initialize
      ✓ should have correct merkle root
    #insert
      ✓ should insert (188ms)
hasher gas 23168
      ✓ hasher gas (177ms)
    #isKnownRoot
      ✓ should return last root (88ms)
      ✓ should return older root (165ms)
      ✓ should fail on unknown root (101ms)
      ✓ should not return uninitialized roots (86ms)


  22 passing (23s)

✨  Done in 25.54s.
stanleychiu@Stanleys-MBP tornado-nova %
```

4. L1Unwrapper is act as a bridge between L1 and the Gnosis Chain. When users withdraw assets from Tornado Cash Nova, the token will be unwrapped on L1 so that users can use them as they wish. The bridge will be able to collect transaction fee from the withdrawal(unwrap) process from user.

Question 4

1. Tornado Cash's name is constantly brought up with criminal events like assets stealing or fraud, what is the team's stance for these kind of usage and how they put the balance between privacy and regulations?

2. It is hard to develop and maintain such one single circuit for all dapps, but there should be a set of circuits for a dapp domain. I am thinking this set of circuit like the programming language, that developers can leverage this set of well defined circuits to develop dapps. I believe in the future we will have many different set of circuits for zk-dapp development, just like today we have so many different programming language for different purpose.