ZKU Week 7 Assignment

Email: qadir@xord.com
Discord Id: AbdulQadir#0432

Survey Questions:

Question One:

Celestia aims to focus on the data availability problem within the framework of a modular architecture where each protocol can focus on a specific aspect of decentralization: storing (transaction) data, building consensus, and executing transactions. What does this mean for future scaling solutions?

As we will have all the data on Celestia or other DA solutions and the building consensus will be on the Settlement layer, the transactions will be on L2, essentially the execution layer.
So all the future scaling solutions will be L2, and they will be using settlement layers such as Ethereum or even Cosmos to settle the transactions, i.e., provide security.

Question Two:

The challenging problem of building a zk-EVM seems to be close to resolution. What does it mean for Solidity programmers w/o substantial zk knowledge in the future?

As solidity is one of the most used languages in the world of blockchain, different L2 solutions work on facilitating the developer. For instance, in Starkware, we have "The Warp," which converts solidity code to Cairo. So the solidity developers can use that instead of learning Cairo. Or in ZkSync, unless you're building a ZK-based dApp, you don't need to know about ZK.

Question Three:

How about Polygon's rapid expansion into the zk space (having acquired several of the zk powerhouses within 2021)?

Polygon has released many solutions in the zk space, including Hermez, Miden, Nightfall, and Zero. All of them are helping to scale Ethereum as the base layer and have different architecture and use cases. For example, Polygon has rapidly expanded by building Miden, which uses the ZK-Starks. At the same time, Polygon acquired Hermez and Mir (Zero) based on Zk-Snarks technology.

Question Four:

What are the most interesting use cases of Stark?

The most compelling use cases of the stark are that it removes the need for trust entirely and preserves the system's integrity. It is also quantum-resistant and is especially good at rolling up more complex data.

Part A: Celestia and zkStarks

Question One:

We have all sorts of data on Ethereum, and the data will also increase as the users increase. Ethereum's size is about to reach 1TB, which is a lot of storage. And this creates a bottleneck because every miner has to store the data, increasing the hardware cost. Now we have L2s, which increase the throughput but again, to make data available for everyone optimistic and zk rollups send that data to ethereum. This causes the network to clog at points. We have seen high gas fees in the past several times.
Now history data is essential in blockchains. Celestia aims to preserve that history while enabling everyone to be able to access that. Anyone can use that data for their dApp, which is a plus, but at the same time, it seems that the network will be a bit more complicated. The network needs to ensure that all data is available at all times.

Question Two:

1. **Polygon PoS:**
   Polygon Pos is an L1 scaling solution with an unprecedented transaction speed and less gas cost using sidechain architecture for transaction processing. Its validators and clients separate its security from L1 blockchains such as Ethereum. In addition, Polygon PoS has EVM compatibility means deploying your smart contracts directly on the Polygon chain. Polygon PoS also have separate native tokens Matic to pay gas fees.

2. **Polygon Zero:**

Polygon zero is an L2 scaling solution for Ethereum. What differentiates it from other Zk-based scaling solutions is the use of a plonky2, their groundbreaking prover system, which generates ZK proofs faster than any other existing tech. In addition, Plonky2 supports efficient recursive proof and can generate proof in 170 milliseconds on a commodity laptop. Polygon zero is also fully compatible with EVM.

3. **Polygon Avail:**
   Polygon Avail aims to facilitate the modular blockchain Paradigm by providing data availability and ordering solutions. Polygon Avail does not perform any transaction execution and computation; instead, it saves the data for the light clients to access quickly. As a result, light clients get ~100% confidence in data availability by performing a constant number of queries. Polygon Avail is currently under development.

4. **Polygon Miden**
   Polygon Midan is also an L2 scaling solution. Still, instead of using Snarks, it uses Zk-Starks to Roll-Ups to batch thousands of ethereum transactions into a single ethereum transaction. Polygon Miden uses the Miden VM, which is Turing's complete stark-based virtual machine supporting privacy and extra features. Transparent (no trusted setup) and post-quantum secure. It has fully open and community-driven development which is under process.

5. **Polygon Hermez:**
   Polygon Hermez is a zk-rollup L2 construction on the top of the Ethereum. Hermes solves the problem of scalability and security by using Snarks as mass transaction processing rolled into a single transaction. Instead of saving whole data, only the proof and compressed data are stored on-chain to prove data is valid. Hermes also guarantees Ethereum-level security by using zk-SNARKs Hermez is fully open-source and community-driven and lives on the main net.

Part B: Flawed Circom Circuits

Question One:

There are two flaws in this both circuits.

The user can add the secret number other than he selects before the round starts. What can be done is the user commits the hash of the chosen number and first verify the commitNumber (hash) of the number before checking less than or more than.

The second fatal issue is the knowledge revealed. As we know, the zero-knowledge property of zero knowledgness that verifies should not know any knowledge other than whether it is true or false. As more and more rounds pass in these circuits, it is easy to predict the user's secret number, as we know the upper and lower bounds.

Question Two:

An Attacker can easily brute force the circuit and generate proof for all the possibilities. A brute force attack can be solved using the salt in the circuit and the commitCard.

Question Three:

In this circuit, there is a design flaw of not using the inclusion proof. Suppose a person is proving a location; it also has to give a reference point to where the place belongs. Merkle tree can be used here for inclusion proof and to prove that the user commitment exists in the Merkle tree.

Question Four: [Bonus]

There is no assert check for the booleanUseCard1 and booleanUseCard2, which checks that their values are either 0 or 1. As a result, an attacker can easily use other numbers as a booleanUseCard to attack the system.

Let's suppose an attacker chooses cards 2 and 4; after submitting proof, when round 1 starts, the contract shows that the required card is 14. An attacker can easily generate the fraud-proof by using the booleanUseCard1 = 1 and booleanUseCard2 = 3 hence

```
signal temp1 <== booleanUseCard1 * card1; // 2 * 2 = 2
signal temp2 <== booleanUseCard2 * card2; // 4 * 3 = 12

requiredCard === temp1 + temp2;
14 === 2 + 12
```

Part C: Final Project:

Project Name: ZkPoolTogether
Project Manager: Abdul Qadir
Report Date: 06/20/2022

Overall Project Status ⏳
On-track / At-risk / Off-track

Notable Changes 📋
None so far.

Timelines & Targets 📅

- Core Development - 22nd of June
- Front-end - 25th of June
- Launch on testnet - 27th of June

Top Level Summary 📝
Finalization of circuits and smart contracts. Added test cases and proof generation of core development. Home page UI setup and wallet connection functionality.

Project Components Overview	Question	Comments
Schedule	Is the project running on schedule?	Yes, the core development is going a little off track due to Merkle tree proof generation, and winner selection process
Resources	Are you low or have excess resources?	Yes, Resources are enough for the V1 launch.
Quality	Is the project's quality being jeopardized?	No,
Roadblocks	Potential risks & roadblocks that could affect the project timeline.	No, Currently, no roadblocks have been faced in the Development.

Completed Tasks:

1. Setting up the ZKPT-Core repo: [link]
    a. Setting up project structure - Week 1
    b. Dependencies and package setup - Week 1
    c. Hardhat support setup - Week 1
    d. Linter, Prettier, and config setup - Week 1
    e. Setup type chain - Week 2

2. Circom circuits:
    a. Merkle Tree inclusion circuit completion - Week 1
    b. Withdraw circuit completion - Week 1
    c. Compilation and setups - Week 1
    d. Fixed flaws in the circuits - Week 2

3. Smart contract development:
    a. MerkleTreeWithHistory contract - Week 1
    b. Pool contract (allowing withdrawal and deposit) - Week 1
    c. DrawManager (85% completed) - Week 2
    d. Mock contract for yield generation - Week 2
    e. Vrf contract for random number - Week 2

4. Unit test and interaction test:
    a. Basic Smart contract compilation - Week 1
    b. Verification test for Poseidon hash function - Week 1
    c. MerkleTreeWithHistory insertion tests - Week 1
    d. Deposit and withdrawal verification of the proof in verifier contract - Week 1
    e. New draw creation test and bare assertions - Week 1
    f. Draws regression testing and winner selection tests - Week 2

5. Setting up the ZKPT_UI repo: [link]
    a. Setting up NextJs boilerplate - Week 2
    b. Home page - Week 2
    c. Adding wallet connect functionality - Week 2
    d. Added wallet balance and contract integration utils - Week 2
    e. Worked on app page UI - Week 2