Zero-Knowledge University (ZKU):

Assignment 7

Tal Derei

18 April 2022

Course Registration Email: Tad222@lehigh.edu

Discord Username: Margulus#4273

Github (Personal Repistory): `https://github.com/TalDerei`

Github (ZK Research): `https://github.com/TalDerei/Zero-Knowledge`

Q1. Celestia (Consensus and Data Availability Layer)

Data availability is one of the bottlenecks to scaling blockchains, but I don't think it's the main way. Blockchain call data from layer-2 is slowly becoming cheaper as [1] EIP-4488 (reducing call data costs for data avaiability on L1 by 5x) is being rolled up, and [2] rollups seeing exponential increased adoption which reduces their amortized costs for end-users. Creating an entire seperate data availability layer to reconstruct the state would eliminate the majority of rolllups costs, since most costs come from posting the calldata state from L1. But eventually (and we're slowly seeing this transition happen), the bottleneck is shifting to how we we scale L2s? L2s by definition are touted as being infinitely scalable (since we can just create an inifinite number of L2s), but then the

communication between them becomes the bottleneck and challenege. So we revert back to how do we scale a normal L2, since it still has an upper TPS limit. Some solutions being explored especially by the ZK Polygon stack are scaling L2s with L3s (i.e. L2s would become shards on L3).

Q2. STARKs vs SNARKs

[1] zk-STARK has much higher efficiency for proof generation and verification compared to zk-SNARK.

[2] zk-STARK uses cryptographic primitives such as collision-resistant hash instead of discrete logarithm and elliptic curves as used in zk-SNARK, making them quantum resistant.

[3] zk-STARK is transparent in that it doesn't require a trusted setup as required by most zk-SNARKs.

[4] zk-STARKs are larger (20–200 KB) than zk-SNARKs (1 KB) when storing on the blockchain.

For reasons [1], [2], and [3] zk-STARKS are better (in principle) than SNARKs in the long-term. But there are two things to note about the current landscape of SNARKs vs STARKS...SNARKs are easier to program with circom/snarkjs than AIRs in STARKs, but that paradigm is slowly shifting as these zkEVM teams are implementing their own high level languages. Additionally, there are teams like Polygon Hermez, Zero, and Miden who are using a dual system to recursively aggregate STARKs, and use that as a private input into SNARKs which act as a compression, and post the compressed state on-chain (which solves issue [4] of STARKs being larger).

Q3. Polygon Stack

Polygon Hermez: zkEVM based on bottom-up construction of rebuilding all opcodes for layer-2. Uses a Proof-of-Donation (Auction-Model) for consensus mechamism, and is being replaced by Proof-of-Efficiency.

Polygon Zero (Mir): Recursive SNARK-based zkEVM, and uses zkSNARK Proof System called Plonky2, a Recursive SNARKs w/ PLONK + FRI Polynomial Commitments.

Polygon Miden: First open-source general-purpose, STARK-based L2 zk-Rollup. The core backend is Miden VM, which improves on Distaff VM (zero-knowledge virtual machine written in Rust) by replacing the underlying proving system with Winterfell (STARK prover). It has an single AIR design that can generate proofs for any arbitrary computation without requiring specific AIRs for each computation. For any program executed on Polygon Miden, a STARK-based proof of execution is automatically generated. Compile solidity-based smart contracts directly into Miden Assembly - the native language of Miden VM

Polygon Nightfall: Optimistic ZK-Rollup for privacy and private transactions.

Polygon Avail: Data availibility layer (to Ceslestia) for Ethereum. Avail is a data availability-specific blockchain designed for standalone chains, sidechains, and other scaling technologies

Polygon Edge: open-source modular blockchain development framework previosuly known as Polygon SDK

Q4. Learnings

Throughout this two-month course, the most valuable thing I've learned is how to develope zk-dapps using frameworks/languages like *Circom* and *snarkjs*. I had some a prior knowledge foundation into zk before entering the course, but the past few months have

enhanced and honed that knowledge, and allowed me to use it in practice by building applications. And the TAs are knwoledgeable and always willing to help out and answer questions. For these reasons, ZKU is unique in principle as being (I think) the only ZKU course out there of it's kind to onboard the next wave of zk developers.

Q5. Project Status

Currently, I've achieved an **MVP** allowing a user send a commitment (by hashing their nullifier and secret), check that commitment against a tree of commutments maintained in a centralized MonogDB database (and who's state root is stored in a smart contract), generate a proof in the browser by integrating a REACT/NEXTjs front-end framework, verify the proof is valid by calling the smart contract. My application is a ZK Airdrop system allowing a user to specify an address which will recieve the airdrop (which may not be their original address). So the zero knowledge component is that the verification happens without associated the collector address to the commitment that they provided. Collector can therefore prove to the contract they made a commitment without revealing which commitment is associated to their public address. For that reason, there's no mechanism to connect a wallet on my application yet, since that would dox their public key.

The next steps are [**1**] adding NFTs to the list of private airdrops (instead of simply just ERC-20), and [**2**] adding a seperate developer console to allow them to collect commitments from a users, add the commitment to a tree, and deploy a smart contract and update the state root. These represents completely different business logics.