Assignment 1

Here is my submission for the assignment 1 of ZKU.

Part 1:

1:

The commit adding the circuit generating the Merkle root is here (the file name is MerkleTree.circom).

2:

When generating the proof for a size of 2 I encounter no errors. However, when trying for a size of 4 or 8, I encounter the following error (here for a size of 8):

> [ERROR] snarkJS: circuit too big for this power of tau ceremony. 14520*2 > 2**12

Therefore, to overcome this issue, I had to increase the parameter for the "powers of tau" ceremony from 12 to 15 so that tau becomes large enough (2**15 = 32 768 > 14 520 * 2 = 29 040).

3:

No, we don't really need a zero-knowledge proof for this as the inputs are public: in fact, here there is no knowledge that is "hidden" behind the zero-knowledge principle, and simply using a publicly verifiable smart contract would achieve the same result as it would also have all the necessary data to compute the Merkle root.

Zero-Knowledge proofs like this would be useful when proving the integrity of transaction without disclosing what those transactions are.

ZCash is an example of a blockchain using this principle: they also use the ZK-Snarks technology in order to secure the transaction taking place on their network, but without letting anyone know what those transactions are about. When a transaction is made and supposed to be private (not all transactions on ZCash are private), a transaction proof is submitted proving that the transaction is valid (that the sender indeed owns enough coins to proceed), but the data itself is encrypted to prevent people from being able to know who the sender / receiver and what amount of token was exchanged.

Miscellaneous

Here is the commit updating the circuit to handle 8 inputs and adding the public.json file (for the input.json update, see here), the commits adding the screenshot of proof generation being successful, the screenshot of execution on Remix with input size 2 and the screenshot for input size 8.

Part 2:

1 and 2:

Here is the commit adding the Minter source code, with contains both the NFT Minter contract and the Merkle Tree contract source codes.

3:

Here is the commit adding the screenshots of the minting of 6 NFTs once the contract was deployed on Remix. For 6 NFTs minted, we have (295 704 + 307 807 + 297641 + 259 907 + 324 175 + 264 041) / 6 ≅ 291 545 gas per mint (it remains little as the contract simply updates the Merkle tree instead of regenerating it entirely at each new mint).

Part 3:

1:

SNARKs and STARKs are both zero-knowledge technologies, but they have some inherent differences, especially in the way we apply them: SNARK proofs are less computationally expensive both to compute for the prover and the verifier, with even constant asymptotic time for the verifier, and have constant sized generated proof. On the contrary, STARK proofs are much more expensive in these fields, and it doesn't have any constant computations (neither absolutely neither asymptotically). But STARKs rely on a trusted setup which means that a common public honest setup must be generated, on which relies the entire proof reliability. On the other hand, SNARKs don't need nor rely on such setups.

2:

The trusted setup processes for Groth16 and PLONK differ in terms of their range of usability: Groth16 needs to generate a new setup for each new circuit which can be a huge flaw when the circuit is complex and needs highly computationally demanding setups. On the contrary, PLONK will only need one trusted setup which can then act as an almost universal setup, thus leading to a single trusted setup process being needed for the entire development progression.

3:

We could use ZK to create NFTs that hold some data that we don't want to be revealed, such as objects in randomized on-chain games (for example cards that we can manipulate / exchange / discard in some card games, but without disclosing what they are).

4:

We could apply ZK to protect some critical data concerning a DAO. For example, we could keep private the investments a DAO made using with the DAO's money pool, but while still being able to prove some properties on them (such as that not more than a certain percentage of the pool was allocated to a single project, or that all funds have been invested).