Assignment 1

Github Link: https://github.com/joellje/zku.ONE
Course registration email: joellimjieen@hotmail.com
Discord username: joellje#8135

Question 1: Intro to circom

1.
```
template GetMerkleRoot(N) {
  signal input leaves[N];
  signal output root;
  var nodes[2*N-1];
  component components[N-1];
  var j = 0;
  var k = 0;

  // Initalize node array with leaves array
  for(var i = N-1; i<2*N-1; i++) {
    nodes[i] = leaves[j];
    j++;
  }

  // Prepare comps
  for(var i = 0; i < N-1; i++) {
  components[i] = Hash();
  }


  // Parent node index = (i-1)/2, where i is the index of the child
  // Append all leaves
  for(var i = 2*N-2; i>0; i-=2) {
    components[k].hash1 <== nodes[i-1];
    components[k].hash2 <== nodes[i];
    nodes[(i-2)/2] = components[k].hashedOutput;
    k++;
  }
  root <== nodes[0];
}
```
circuit.circom (GetMerkleRoot function)

```
[
  "7457672556014162487472065518158328090252704233415054189820328174772177160972",
  "1",
  "2",
  "3",
  "4",
  "5",
  "6",
  "7",
  "8"
]
```

public.json

```
{"leaves":[1,2,3,4,5,6,7,8]}
```

Input.json

```
(base) joellim@Joels-MacBook-Pro circuit_js % snarkjs groth16 setup ../circuit.r1cs pot12_final.ptau circuit_0000.zkey

[ERROR] snarkJS: circuit too big for this power of tau ceremony. 9240*2 > 2**12
```

2. I encountered this error when I tried to use a list of 8 numbers. The circuit was too big for this power of tau ceremony, since 18480 was larger than 2^{12}. In the power of tau ceremony, I used 15 instead of 12, as seen here

```
snarkjs powersoftau new bn128 15 pot12_0000.ptau -v
```

These were my outputs for the last few lines of the execution.

```
PROBLEMS 2   OUTPUT   TERMINAL   GITLENS   SQL CONSOLE   DEBUG CONSOLE

[DEBUG] snarkJS: betaTauG1: fft 15 mix end: 5/8
[DEBUG] snarkJS: betaTauG1: fft 15 mix end: 1/8
[DEBUG] snarkJS: betaTauG1: fft 15 mix end: 4/8
[DEBUG] snarkJS: betaTauG1: fft 15 mix end: 0/8
[DEBUG] snarkJS: betaTauG1: fft 15 mix end: 6/8
[DEBUG] snarkJS: betaTauG1: fft 15 mix end: 3/8
[DEBUG] snarkJS: betaTauG1: fft  15  join: 13/15
[DEBUG] snarkJS: betaTauG1: fft 15 join  13/15  3/4 0/1
[DEBUG] snarkJS: betaTauG1: fft 15 join  13/15  4/4 0/1
[DEBUG] snarkJS: betaTauG1: fft 15 join  13/15  2/4 0/1
[DEBUG] snarkJS: betaTauG1: fft 15 join  13/15  1/4 0/1
[DEBUG] snarkJS: betaTauG1: fft  15  join: 14/15
[DEBUG] snarkJS: betaTauG1: fft 15 join  14/15  2/2 1/2
[DEBUG] snarkJS: betaTauG1: fft 15 join  14/15  1/2 1/2
[DEBUG] snarkJS: betaTauG1: fft 15 join  14/15  2/2 0/2
[DEBUG] snarkJS: betaTauG1: fft 15 join  14/15  1/2 0/2
[DEBUG] snarkJS: betaTauG1: fft  15  join: 15/15
[DEBUG] snarkJS: betaTauG1: fft 15 join  15/15  1/1 1/4
[DEBUG] snarkJS: betaTauG1: fft 15 join  15/15  1/1 3/4
[DEBUG] snarkJS: betaTauG1: fft 15 join  15/15  1/1 0/4
[DEBUG] snarkJS: betaTauG1: fft 15 join  15/15  1/1 2/4
(base) joellim@Joels-MacBook-Pro circuit_js % snarkjs groth16 setup ../circuit.r1cs pot12_final.ptau circuit_0000.zkey

[INFO]  snarkJS: Reading r1cs
[INFO]  snarkJS: Reading tauG1
[INFO]  snarkJS: Reading tauG2
[INFO]  snarkJS: Reading alphatauG1
[INFO]  snarkJS: Reading betatauG1
[INFO]  snarkJS: Circuit hash:
                6624611b ed846ce0 e54b072b 925b3abb
                2be73b1c f0bb7f12 9463150a b4c575dd
                e469cf81 f063cb34 b0babc87 cd2d0d57
                ae5de754 c4d4bcfd b1e2b5d4 40dd3997
(base) joellim@Joels-MacBook-Pro circuit_js % snarkjs zkey contribute circuit_0000.zkey circuit_0001.zkey --name="1st Contributor Name" -v -e="randomtext"

[DEBUG] snarkJS: Applying key: L Section: 0/9239
[DEBUG] snarkJS: Applying key: H Section: 0/16384
[INFO]  snarkJS: Circuit Hash:
                6624611b ed846ce0 e54b072b 925b3abb
                2be73b1c f0bb7f12 9463150a b4c575dd
                e469cf81 f063cb34 b0babc87 cd2d0d57
                ae5de754 c4d4bcfd b1e2b5d4 40dd3997
[INFO]  snarkJS: Contribution Hash:
                75bb26b1 caefb284 40a95af7 d458f4de
                25ea10dd 197ed2ab b7e861ea 585f1858
                19171b21 d6b9b44c 273cd7c1 9e1329be
                8b5453b8 3668b82f a79f2c90 1cfce57d
(base) joellim@Joels-MacBook-Pro circuit_js % snarkjs zkey export verificationkey circuit_0001.zkey verification_key.json

(base) joellim@Joels-MacBook-Pro circuit_js % snarkjs groth16 prove circuit_0001.zkey witness.wtns proof.json public.json

(base) joellim@Joels-MacBook-Pro circuit_js % snarkjs groth16 verify verification_key.json public.json proof.json

[INFO]  snarkJS: OK!
(base) joellim@Joels-MacBook-Pro circuit_js %
```

These were the commands that I ran in totality. I have included the script in the Github repo.

```
circom circuit.circom --r1cs --wasm --sym --c
cd circuit_js

node generate_witness.js circuit.wasm ../input.json witness.wtns

snarkjs powersoftau new bn128 15 pot12_0000.ptau -v

snarkjs powersoftau contribute pot12_0000.ptau pot12_0001.ptau --name="First contribution" -v -e="randomtext"

snarkjs powersoftau prepare phase2 pot12_0001.ptau pot12_final.ptau -v

snarkjs groth16 setup ../circuit.r1cs pot12_final.ptau circuit_0000.zkey
snarkjs zkey contribute circuit_0000.zkey circuit_0001.zkey --name="1st Contributor Name" -v -e="randomtext"

snarkjs zkey export verificationkey circuit_0001.zkey verification_key.json

snarkjs groth16 prove circuit_0001.zkey witness.wtns proof.json public.json

snarkjs groth16 verify verification_key.json public.json proof.json
```

3. We do need zero-knowledge proof for this. A publicly verifiable smart contract that computes Merkle root will achieve the same. We are still able to construct a Merkle Tree without zero-knowledge proofs. Zero-knowledge proofs allow us to improve the security nature if heightened security is needed. It also allows transactions and interactions to be non-interactive, as well as making transactions cheaper and resource efficient. One such example is ZKash, which is a privacy-focused, blockchain-based payments network that uses zero-knowledge proofs (ZKPs) to shield transactions, making the contents of a transaction private even on a public blockchain.

Question 2: Minting an NFT and committing the mint data to a Merkle Tree

3.

```
[vm] from: 0x5B3...eddC4 to: NFT.mint(bytes32[],uint256,address) 0xe28...4157A value: 0 wei data: 0x23c...a4c53 logs: 1 hash: 0xda3...1691f
status              true Transaction mined and execution succeed
transaction hash    0xda3128de46c36d35909469674d62d27f610c043b5545822d29a9b255ead1691f
from                0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to                  NFT.mint(bytes32[],uint256,address) 0xe2899bddFD890e320e643044c6b95B9B0b84157A
gas                 80000000 gas
transaction cost    112031 gas
execution cost      112031 gas
hash                0xda3128de46c36d35909469674d62d27f610c043b5545822d29a9b255ead1691f
input               0x23c...a4c53
decoded input       {
                        "bytes32[] _merkleProof": [
                            "0xc62ec191e9985a096ba0f0e97ab9447776623701b95c38768736cc57f6aaeb8a",
                            "0x48c4fc8d2d80630dbcd1ffb5d3e254175ead638dc2200a55d838187d39fa4c53"
                        ],
                        "uint256 _tokenId": "1",
                        "address _recipient": "0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2"
                    }
decoded output      {}
logs                [
                        {
                            "from": "0xe2899bddFD890e320e643044c6b95B9B0b84157A",
                            "topic": "0xddf252ad1be2c89b69c2b068fc378daa952ba7f163c4a11628f55a4df523b3ef",
                            "event": "Transfer",
                            "args": {
                                "0": "0x0000000000000000000000000000000000000000",
                                "1": "0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2",
                                "2": "1",
                                "from": "0x0000000000000000000000000000000000000000",
                                "to": "0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2",
                                "tokenId": "1"
                            }
                        }
                    ]
val                 0 wei
```

1st Mint

```
[vm] from: 0x5B3...eddC4 to: NFT.mint(bytes32[],uint256,address) 0xe28...4157A value: 0 wei data: 0x23c...a4c53 logs: 1 hash: 0x396...99bec
status              true Transaction mined and execution succeed
transaction hash    0x396110039b3fbc00062b687a18e1bfc4effe3a5a0a989b8916b64ea47f899bec
from                0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to                  NFT.mint(bytes32[],uint256,address) 0xe2899bddFD890e320e643044c6b95B9B0b84157A
gas                 80000000 gas
transaction cost    94909 gas
execution cost      94909 gas
hash                0x396110039b3fbc00062b687a18e1bfc4effe3a5a0a989b8916b64ea47f899bec
input               0x23c...a4c53
decoded input       {
                        "bytes32[] _merkleProof": [
                            "0x7b021fbb755ef68defccfa51adbdaaa9c291110f45003994370545a161c8ef27",
                            "0x48c4fc8d2d80630dbcd1ffb5d3e254175ead638dc2200a55d838187d39fa4c53"
                        ],
                        "uint256 _tokenId": "2",
                        "address _recipient": "0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db"
                    }
decoded output      {}
logs                [
                        {
                            "from": "0xe2899bddFD890e320e643044c6b95B9B0b84157A",
                            "topic": "0xddf252ad1be2c89b69c2b068fc378daa952ba7f163c4a11628f55a4df523b3ef",
                            "event": "Transfer",
                            "args": {
                                "0": "0x0000000000000000000000000000000000000000",
                                "1": "0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db",
                                "2": "2",
                                "from": "0x0000000000000000000000000000000000000000",
                                "to": "0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db",
                                "tokenId": "2"
                            }
                        }
                    ]
val                 0 wei
```

2nd Mint

```
[vm] from: 0x5B3...eddC4 to: NFT.mint(bytes32[],uint256,address) 0xe28...4157A value: 0 wei data: 0x23c...ab5f7 logs: 1 hash: 0xdfc...5bd36
status              true Transaction mined and execution succeed
transaction hash    0xdfcaf3ae7f67dec4b931e7035f274ccb9a21e8e9510a02a2fc79f73e52a5bd36
from                0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to                  NFT.mint(bytes32[],uint256,address) 0xe2899bddFD890e320e643044c6b95B9B0b84157A
gas                 80000000 gas
transaction cost    94941 gas
execution cost      94941 gas
hash                0xdfcaf3ae7f67dec4b931e7035f274ccb9a21e8e9510a02a2fc79f73e52a5bd36
input               0x23c...ab5f7
decoded input       {
                        "bytes32[] _merkleProof": [
                            "0x762e94764cb463037ff7881ec4a472c8f9bb61541bf4ecf091537023103d60fd",
                            "0x8aeee7cfcbb99fcbf59ed8365e219efb28c396cbaf15bdf797ec10341f9ab5f7"
                        ],
                        "uint256 _tokenId": "3",
                        "address _recipient": "0x78731D3Ca6b7E34aC0F824c42a7cC18A495cabaB"
                    }
decoded output      {}
logs                [
                        {
                            "from": "0xe2899bddFD890e320e643044c6b95B9B0b84157A",
                            "topic": "0xddf252ad1be2c89b69c2b068fc378daa952ba7f163c4a11628f55a4df523b3ef",
                            "event": "Transfer",
                            "args": {
                                "0": "0x0000000000000000000000000000000000000000",
                                "1": "0x78731D3Ca6b7E34aC0F824c42a7cC18A495cabaB",
                                "2": "3",
                                "from": "0x0000000000000000000000000000000000000000",
                                "to": "0x78731D3Ca6b7E34aC0F824c42a7cC18A495cabaB",
                                "tokenId": "3"
                            }
                        }
                    ]
val                 0 wei
```

3rd Mint

```
[vm] from: 0x5B3...eddC4 to: NFT.mint(bytes32[],uint256,address) 0xe28...4157A value: 0 wei data: 0x23c...ab5f7 logs: 1 hash: 0xb8c...01de4
status              true Transaction mined and execution succeed
transaction hash    0xb8c525ce23ac4b568de9c3362779a431cc48ebb633b66e5f632eb8bec7001de4
from                0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to                  NFT.mint(bytes32[],uint256,address) 0xe2899bddFD890e320e643044c6b95B9B0b84157A
gas                 80000000 gas
transaction cost    94931 gas
execution cost      94931 gas
hash                0xb8c525ce23ac4b568de9c3362779a431cc48ebb633b66e5f632eb8bec7001de4
input               0x23c...ab5f7
decoded input       {
                        "bytes32[] _merkleProof": [
                            "0x379dfb9706f0aea22cd1a01963349323101470a9aaf87ce10fb28fdd82d52f83",
                            "0x8aeee7cfcbb99fcbf59ed8365e219efb28c396cbaf15bdf797ec10341f9ab5f7"
                        ],
                        "uint256 _tokenId": "4",
                        "address _recipient": "0x617F2E2fD72FD9D5503197092aC168c91465E7f2"
                    }
decoded output      {}
logs                [
                        {
                            "from": "0xe2899bddFD890e320e643044c6b95B9B0b84157A",
                            "topic": "0xddf252ad1be2c89b69c2b068fc378daa952ba7f163c4a11628f55a4df523b3ef",
                            "event": "Transfer",
                            "args": {
                                "0": "0x0000000000000000000000000000000000000000",
                                "1": "0x617F2E2fD72FD9D5503197092aC168c91465E7f2",
                                "2": "4",
                                "from": "0x0000000000000000000000000000000000000000",
                                "to": "0x617F2E2fD72FD9D5503197092aC168c91465E7f2",
                                "tokenId": "4"
                            }
                        }
                    ]
val                 0 wei
```


4th Mint

Question 3: Understanding and generating ideas about ZK technologies

Understanding and generating ideas about ZK technologies

1. SNARK (zero-knowledge succinct **non-interactive** argument of knowledge)
   - not quantum resistant
   - cheaper than STARKs
   - smaller than STARKs
   - requires a common reference string (CRS) generated in advance

   STARK (zero-knowledge scalable **transparent** argument of knowledge)
   - relies on hash functions
   - quantum resistant
   - no need trusted set-up

   SNARKs require a trusted set up while STARKs do not.
   One example of SNARKs is Zcash.
   Transactions in the network can remain encrypted but still be verified by using ZKPs. Those that are enforcing the consensus rules (trusted setup) do not need to know all of the data underlying each transaction

2. Groth16 requires a trusted ceremony for each circuit. PLONK does not require it, the powers of tau ceremony, which is universal is enough. This means that Groth16 has to conduct a ceremony every time the smart contract is edited. PLONK's universal setup means that only one setup must be done, but it works for all future circuits. This will only costs 10% more gas than Groth16.

3. Whitelisting in NFTs. When a user is added to the whitelist, his/ her wallet address can be added to the leaf nodes. These leaf nodes are used to generate the Merkle Root. When the user wants to verify that he is part of the whitelist during mint, the verifyProof function will run and check if the root generated is that same as that which was previously generated with the initial whitelist.