ZKU – Cohort 4 (Jul-Aug 2022)
Week 1: Introduction to ZKP
Assignment ♯1

Iskander Andrews[1]

Abstract— **This document is for answering ZKUC04, Week-1, Introduction to ZKP Assignment. It consists of three main parts.**

Part 1: Theoretical background of zk-SNARKs and zk-STARKS.
Part 2: Getting started with circom and snarkjs.
Part 3: Reading and designing circuits with Circom.

The total number of questions and points is 13**, including the bonus questions.**

I. PART 1: THEORETICAL BACKGROUND OF ZK-SNARKS AND ZK-STARKS

A. *1.1 Explain in 2-4 sentences why SNARK requires a trusted setup while STARK doesn't.*

zkSTARKs requires a trusted setup because it requires an initial creation event of the keys, which are used to create proofs for private transactions, and used in the verification of those proofs.
On the other hand, zkSTARKs does not require a trusted-setup for utilizing the network because it is transparent which means that they use publicly verifiable randomness to create a trustless verifiable computing system. [**?**][**?**]

B. *1.2 Name two more differences between SNARK and STARK proofs.*

1) zkSTARKs rely on hash functions, so it would be a quantum resistant, on the other hand zkSNARKs are not quantum resistant so the privacy of SNARKs could be broken. [**?**]
2) zkSTARKs have larger proof sizes than zkSNARKs, which means that the verification in STARKs takes more time than in SNARKs. [**?**]

II. PART 2: GETTING STARTED WITH CIRCOM AND SNARKJS:

A. *Question (1)*

1) *2.1 What does the circuit in HelloWorld.circom do?*

It implements a Circom circuit to prove the multiplication of two private inputs signal identifiers a, and b,and calculate the result in a public output signal identifier c.

iskander.s.andrews@gmail.com
Isk0996
iskdrews

B. *Question (2)*

1) *2.2.1 What is a Powers of Tau ceremony?*

The Power of Tau (τ) is a secure multi-party computation (*MPC*) ceremony. It is used in generating the parameters of the first phase (1) in zkSNARK which consists of two phases. It can generate paramters for all the circuits up to a depth of 2^{21}. It proceeds in turns, one turn for each party, and the result of the computation of each party is then added to a public transcript, which will allow all the system be verified. The result parameters of the system then becomes secure if one party was able to destroy the random toxic waste. [**?**]

2) *2.2.2 Explain why this is important in the setup of zk-SNARK applications.*

In order to deploy a zkSNARK circuits, a developer must perform a computation for generating the "Proving Key", and the "Verifying key", and this process called a "Trusted Setup". There is a downside for this process because it produces bad numbers called "toxic waste" file that needs to be destroyed otherwise it will produce fake proofs which will violate the security of the system. So to resolve that, the "Trusted Setup" can be setup using specific Cryptographic ceremony like the "Powers of Tau" ceremony, which will be used only in generating the first phase of parameter generation of all the projects, since zk-SNARKs requires two phases of parameter generation. [**?**]

Power of Tau ceremony is helping in creating what is called a "Chain of trusted-setup", through allowing multiple parties to setup a trusted setup and then adding the results to a public transcript, and the system can be verified by publishing the results. Moreover, the system will be considered secure, if only one of the parties has successfully destroying the "toxic waste" file. [**?**]

3) *2.2.3 How are Phase 1 and Phase 2 trusted setup ceremonies different from each other?*

zkSNARK projects consists of two phases of parameter generation, The Power of Tau can be only used in the first phase for all the projects. While the second phase if a circuit-speicfic that needs to be done for each circuit, and the individual teams are responsible on it.
Phase-one is a universal phase ceremony for the entire community, while to start phase-two, any zkSNARK projects can pick any point of the ceremony to begin their circuit-specific second phase. [**?**] [**?**]

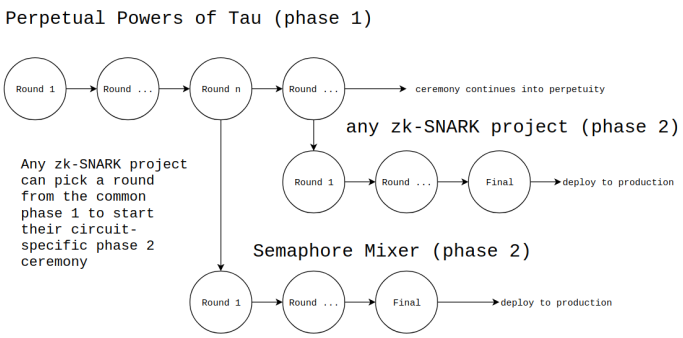

Fig. 1. Image borrowed from Wei Jie Koh's post. [**?**]

C. *Question (3)*

*1) **2.3.2 Try to run compile-Multiplier3-groth16.sh. You should encounter an error[T3001] with the circuit as is. Explain what the error means and how it arises.***

error[T3001] means that the constraint is Non-quadratic expression, because it should be one of those constraints: [**?**]

Constant values
Linear expression $2 * x + 3 * y + 2$
Quadratic expression $(2*x+3*y+2)*(x+y)+6*x+y-2$

D. *Question (4)*

*1) **2.4.1 What are the practical differences between Groth16 and PLONK? Hint: compazre and contrast the resulted contracts and running time of unit tests from the two protocols.***

1) Groth16 requires a trusted ceremony to each circuit, while PLONK does not require it, its just enough to use The Power of Tau universal setup ceremony. [**?**]
2) After running the test/test.js in Question (5), it turns out that Groth16 proofing system took shorter time than PLONK proofing system, where Groth16 took $1132ms$, and PLONK took $1462ms$.
3) The `verifyProof` in Multiplier3Verifier.sol of Groth16 is simpler than The `verifyProof` in Multiplier3-plonkVerifier.sol of PLONK

E. *Question (5)*

*1) **2.5.3 In test/test.js, add the unit tests for Multiplier3 for both the Groth16 and PLONK versions. Include a screenshot of all the tests (for HelloWorld, Multiplier3 with Groth16, and Multiplier3 with PLONK) passing in your PDF file.***

Look at Fig, [2].

III. **Part 3:** Reading and Designing Circuits with Circom

A. *Question (1)*

*1) **3.1.1 What does the 32 in Line 9 stand for?***

n variable in the template LessThan(n) function refers to the number of bits for the input signal, and 32 stands for the unsigned integers, non-negative integer in the range $[0 to 4294967295]$.

```
[        @archlinux ~/        /                /zku/week1/Q2]$ npm run test

> test
> node scripts/bump-solidity.js && npx hardhat test

Bump Solidity contract: HelloWorldVerifier
Bump Solidity contract: Multiplier3-plonkVerifier
Bump Solidity contract: Multiplier3Verifier


  HelloWorld
    ✔Circuit should multiply two numbers correctly (884ms)
    ✔Should return true for correct proof (2136ms)
    ✔Should return false for invalid proof (290ms)

  Multiplier3 with Groth16
    ✔Circuit should multiply three numbers correctly
    ✔Should return true for correct proof (1132ms)
    ✔Should return false for invalid proof (334ms)

  Multiplier3 with PLONK
    ✔Should return true for correct proof (1462ms)
    ✔Should return false for invalid proof


  8 passing (35s)
```

Fig. 2. All tests are passing.

*2) **3.1.2 What are the possible outputs for the LessThan template and what do they mean respectively?***

The expected outputs are either 0 or 1 which is a boolean return datatype, so if the number $n1$ is not less than number $n2$ the output will be 0 which means false $n1 > n2$. Otherwise the output will be 1 which means true $n1 < n2$.

B. *Question (2)*

*1) **3.2.2 You can run npm run test:fullProof while inside the zkPuzzles directory to test your modified circuit. You are expected to encounter an error. Record the error, resolve it by modifying project/zkPuzzles/scripts/compile-circuits.sh, and explain why it has occurred and what you did to solve the error?***

The error was [ERROR] snarkJS: circuit too big for this power of tau ceremony. $97588 > 2^{16}$. It means that the current circuit requires a bigger power of tau ceremony, because the current one is a 16 ceremony $2^{16} < 97588$, which is less than the expected from the circuit 97588. To resolve this error, just change the downloaded Power of Tau ceremony file to be bigger than 16, so I downloaded Power of Tau with the power of 20 therefore, $2^{20} > 97588$.

```
# ./zku/week1/Q3/projects/zkPuzzles/scripts
if [ -f ./powersOfTau28_hez_final_20.ptau ]; then
    echo "powersOfTau28_hez_final_20.ptau already
    exists. Skipping."
else
    echo 'Downloading powersOfTau28_hez_final_20.
    ptau'
    wget https://hermez.s3-eu-west-1.amazonaws.com/
    powersOfTau28_hez_final_20.ptau
fi
```

Listing 1. Expected solution

C. *Question (3) [Bonus]*

*1) **3.3 Run npm run test to prove that the solution to the following system of equations is $x = 15$, $y = 17$, $z = 19$.***

Look at fig, [3].

Fig. 3. All tests are passing.

D. *Question (4) [Bonus]*

1) *3.4 what other libraries do you think could be created to help foster the growth of ZK applications?*

I think it would be better if there is a support for strings in the debugging log() library, that would be useful to separate between the printed logs, for exmaple: log("Single input in", in);. Also, it would be useful if there are a static typing system for the templates and the functions, typing the data-types of the expected inputs and the expected return.

ACKNOWLEDGMENT

I would like to thank so much Heather @Giveth who told me to register in this great course, and would like to thank hadzija0842 and cs6500 for their great help and support. I am really thankful to them to everyone working behind the science to create, maintain, review, manage, and produce this great ZKP course, I see great efforts.

REFERENCES

[1] Mattison Asher, Coogan Brennan, Zero-Knowledge Proofs: STARKs vs SNARKs, May 18, 2021. link
[2] EthHub, ZK-STARKs. link
[3] github.com/ebfull/powersoftau. link
[4] Koh Wei Jie, Announcing the Perpetual Powers of Tau Ceremony to benefit all zk-SNARK projects, Sep 11, 2019. link
[5] Polygon Hermez, Hermez Zero-Knowledge Proofs, 2020. link
[6] Matthew Finestone, Loopring Begins zkSNARK Trusted Setup Multi-Party Computation Ceremony, Nov 10, 2019 link
[7] Circom official documentations. link
[8] Snarkjs official documentation link