<u>ZKU Course - Week 1</u>

From: @Rotcivegaf
Email: victorfage@gmail.com
Discord Handle: rotcivegaf#4718

Part 1

1) The structured reference strings(SRS), universal and updatable SNARKS with fast for all circuits verification time, like Sonic, Marlin, Plonk
The common reference strings(CRS), transparent arguments SNARKS with linear verification time, like Ligero, Aurora

	Common Reference String (CRS)		Structured Reference String (SRS)		
			Universal		Circuit Specific
	Transparent arguments		Updatable	Static	
Verification time					
Linear	Ligero, Aurora	Bulletproofs, Halo		AuroraLight	
Fast for all circuits	Fractal	Spartan, SuperSonic-CG	Sonic, Marlin, SuperSonic-RSA, Plonk		Groth16, BTCV14
Fast for optimized circuits	zk-STARK, Succinct Aurora	Hyrax		Libra	

2) The SNARK requires a trusted setup because if someone have access to the randomness what created the parameters of the setup he can create false proofs what look valid to the verificator
The STARK don't require because it base in symmetric cryptographic, use hash functions resistant to colitions

3) The SNARKs require an initial trusted setup and the STARKs not
The size of the SNARKs proofs is smaller than STARKs

Part 2

1) Done

2.1) The HelloWorld.circom checks that c is the multiplication of a and b
declares 2 input signals: a, b
declares a output signal: c
And the constraint: c <== a * b

2.2) Powers of Tau ceremony is a multi-party computation (MPC) ceremony which constructs partial zk-SNARK parameters for all circuits up to a depth of 2**21. It works by taking a step that is performed by all zk-SNARK MPCs and performing it in just one single ceremony. This makes individual zk-SNARK MPCs much cheaper and allows them to scale to practically unbounded numbers of participants.

This is important because if we want deploys a zk-SNARK circuit we need a trusted setup to generate a proving key and verifying key and this process produces toxic waste which must be discarded because can be used to produce fake proofs with Powers of Tau ceremony we resolve this problem.

2.3) Phase-one: Powers of Tau
Is a general setup for all circuits up to a given size, is for the entire community and is perpetual(there is no limit to the number of participants required).

Phase-two: Converts the output of the Powers of Tau phase into a relation-specific CRS, Zk-SNARK project can pick any round of the powers of Tau ceremony to begin their circuit-specific.

3.1) [commit]()

3.2)

```
error[T3001]: Non quadratic constraints are not allowed!
    ┌─ "Multiplier3.circom":14:4
    │
 14 │     d <== a * b * c;
    │     ^^^^^^^^^^^^^^^ found here
    │
    = call trace:
      ->Multiplier3

previous errors were found
```

In circom all constraints must be quadratic of the form A*B + C = 0, where A, B and C are linear combinations of signals.

3.3) [commit]()

4.1) Groth16 requires a trusted ceremony for each circuit. PLONK does not require it, it's enough with the powers of tau ceremony which is universal.

4.2) Groth16 is best suited when an application needs to generate many proofs for the same circuit (for instance a single logic computation) and performance is critical, while PlonK is best suited when it needs to handle many different circuits (for example different arbitrary business logics) with reasonably fast performance.

5.1) [commit]()

5.2) commit

5.3)

```
~/W/ZKU-Course/week1/Q2 master !7 ?7 > npx hardhat test


  HelloWorld
3 * 4 = 12
    ✔ Should return true for correct proof (1803ms)
    ✔ Should return false for invalid proof (194ms)

  Multiplier3 with Groth16
3 * 4 * 5 = 60
    ✔ Should return true for correct proof (1074ms)
    ✔ Should return false for invalid proof (184ms)

  Multiplier3 with PLONK
3 * 4 * 5 = 60
    ✔ Should return true for correct proof (1360ms)
    ✔ Should return false for invalid proof


  6 passing (5s)

~/W/ZKU-Course/week1/Q2 master !7 ?7 >
```

commit

Part 3

1.1) N is the number of bits the input have, in our case it's 32, 2**32 = 4294967296 It's the maximum it support

1.2) There are two possible outputs, 0(True) or 1(False) it's a boolean function
If the signal input its less than 10 return 1(True) otherwise return 0(False)

1.3) commit

2.1) commit

2.2) commit

“[ERROR] snarkJS: circuit too big for this power of tau ceremony. 69562 > 2**16”
The max power of powersOfTau28_hez_final_16.ptau ceremony its 2**16 = 65536 and the sudoku needs at least 69562. We need a little more power 2**17 should be okey, with power of powersOfTau28_hez_final_17.ptau we get that

2.3) commit

2.4) The brute force method require a $(9^n) * rows * columns$ complexity, check each row in solved has all the numbers from 1 to 9, came with the column
For the our method check all rows and columns and blocks sum to 45 and sum of squares = 285, this lowers the complexity a lot, we get a $rows * columns$ complexity

3.[bonus]) commit

4.[bonus]) The circomlib and circomlib-matrix libraries need more documentation, for example in the matMul the parameters m, n and p there are no documented

I did not find any library for manipulation and comparison of strings
I'm still not sure of its usefulness

Sources:

- Comparing General Purpose zk-SNARKs
- ebfull/powersoftau/README.md
- Announcing the Perpetual Powers of Tau Ceremony to benefit all zk-SNARK projects
- The Power of Tau or: How I Learned to Stop Worrying and Love the Setup
- zk-SNARKs y zk-STARKs Explicadas
- Constraint Generation
- More basic circuits
- iden3/snarkjs/README.md
- Prove schemes and curves