# *Advanced circom programming*

A learning group for ZK and SNARK application development

*Daniel Szego*
In: [https://www.linkedin.com/in/daniel-szego/](https://www.linkedin.com/in/daniel-szego/)

## *Logistics: ZK Learning Group*

Every month, third thursday in 2025, from 18 (CET)

One hour, presentation + short discussion

Different topics on zero knowledge proof,

- mostly from programmer and application developers perspective
- with some theory

Coordination:

- Discord channel: LF Decentralized Trust
  https://discord.com/channels/905194001349627914/1329201532628898036
- Meetup.com: https://www.meetup.com/lfdt-hungary/events/305634614/
- Repo with all the contents:
  https://github.com/LF-Decentralized-Trust-labs/zk-learning-group

Quizzes and small programming challenges, LFDT merchs at the end

## *Logistics: Hunting for the SNARK*

February - Introduction, Theory : Definitions and building blocks

March - Theory : Polynomial commitments

April - Theory : Interactive oracle proofs

May - Programming : Circom basics

**June** - Programming : Circom advanced

July - Programming : Noir

August - Programming : Noir

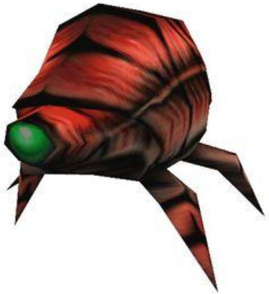
September : Applications : Off-chain transaction

October : Applications : Proving solvency

November : Applications : Rollup

December : Wrap up, Applications

*Subject to change based on community discussion ….*



## Agenda

- *zkSNARK*
- *Circom*
- *Programming 1*
- *Programming 2*
- *Circomlib*
- *Power of Tau*
- *Tips and tricks*
- *Demo*
- *Link, Resources, Challenge*
- *Q&A*

## *(zk)SNARK - Succinct Non-interactive ARgument of Knowledge*

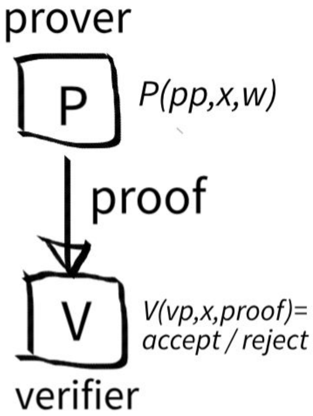
***Computation***: arithmetic circuit : *C( x, w ) -> F*
- *x* public input
- *w* private input, witness
- high level computation
- arithmetic circuit
- polynomials

***Prover*** algorithm: *P (pp, x, w) -> proof*

***Verifier*** algorithm: *V (vp, x, proof) -> accept / reject*

***Properties:***

- *Succinct:*
- *Complete:*
- *Knowledge sound:*
- *Zero knowledge*



## *Circom*

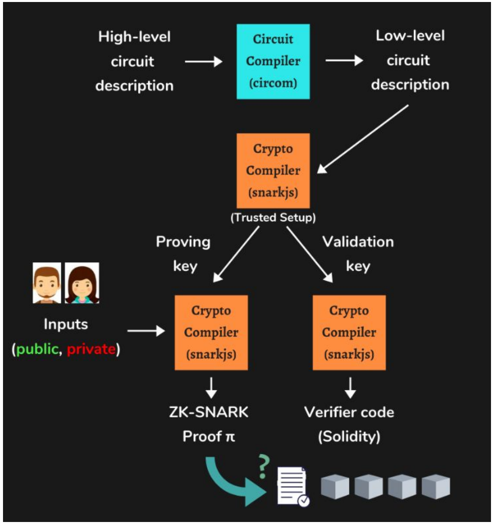
**DLS / circuit programming language and development environment for arithmetic circuits and constraints**

**Used e.g. in tornado cash**

**Supports:**
- **Groth16**
- **Plonk**

**Well established (exist 3 years :)**

**Supported integration:**
- **javascript (snarkjs)**
- **cpp**
- **solidity verifier**


High-level circuit description
Circuit Compiler (circom)
Low-level circuit description
Crypto Compiler (snarkjs)
(Trusted Setup)
Proving key
Validation key
Inputs (public, private)
Crypto Compiler (snarkjs)
Crypto Compiler (snarkjs)
ZK-SNARK Proof π
Verifier code (Solidity)
?

[https://docs.circom.io/](https://docs.circom.io/)

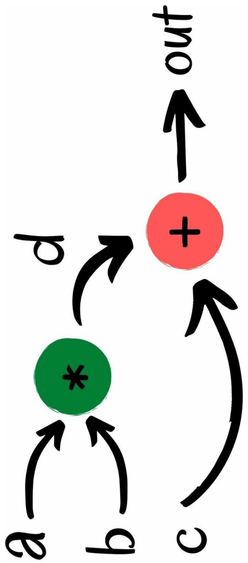
## *Programming 1*

**Template:**
- **abstract circuit**
- **input signals, output signals, internal signals**
- **public vs private inputs**
- **R1C, rank one constraints:**
  **a * b = c, where a, b, and c are linear combinations of variables: a * b === c**
- **setting value and calculations: a * b —> c**

**Arrays of circuits (fix size)**

**Instantiating a circuit**
- **“component” keyword**
- **main component, subcomponent**

**Wiring inputs and output**



# *Programming 2*

**Variables:**

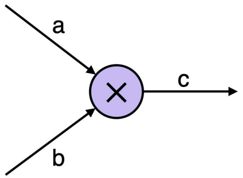
- **signal is immutable / variable mutable**
- **variable is not part of R1CS (no === or <==)**
- **var vs signal keyword**
- **signal versus variable assignment**

**metaprogramming with C like syntax:**

- **loop**
- **if - then on variables**
- **one big generated "physical" R1CS circuit**
- **everything is fixed size**

**Template / circuit arguments**

**Functions**



## *Circomlib*

**Different practical preprogrammed circom templates:**

- **basic logical gates**
- **bit operations**
- **isZero**
- **multiplexer**
- **conditions: lessThan, GreaterThan, etc**
- **switcher**

**cryptographic primitives**

- **EDDSA, Edwards-curve Digital Signature Algorithm)**
- **MIMC:hash function**
- **Pedersen commitment**
- **Poseidon hash function**
- **sha256**

**npm install circomlib**

iden3/**circomlib**

Library of basic circuits for circom

[https://github.com/iden3/circomlib/tree/master/circuits](https://github.com/iden3/circomlib/tree/master/circuits)

# *Power of tau*

**Trusted setup:**
- **per circuit / universal /transparent**

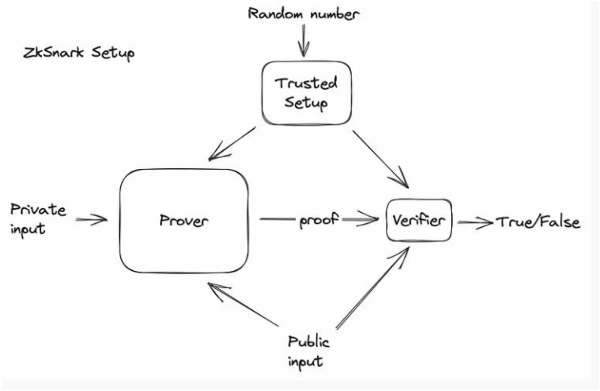
**a multi-party computation (MPC) ceremony**

**generate parameters for zk-SNARKs**

**multiple participants contributing randomness**

**one participant acts honestly, the resulting parameters remain secure and confidential**

**"tau" refers to a secret value or trapdoor used in the parameter generation process.**



**snarkjs powersoftau new**
**snarkjs powersoftau contribute**

https://docs.circom.io/getting-started/proving-circuits/

# *Tips and tricks*

**Mixing variables and signals**

- **e.g. comparison / if-then**

**Security:**

- **under constraints**
- **using ‘<--’, “<==”, “<==”**
- **optimizer considerations**
- **template programming: no constraints**

**Performance:**

- **number of constraints: Groth 16 / Plonk**

**Test, debug:**

- **‘log’ command**
- **unit test:**
  **https://github.com/iden3/circom_tester**

## *Demo*

**Templates**

**Circuits / components**

**Wiring**

**Variables**

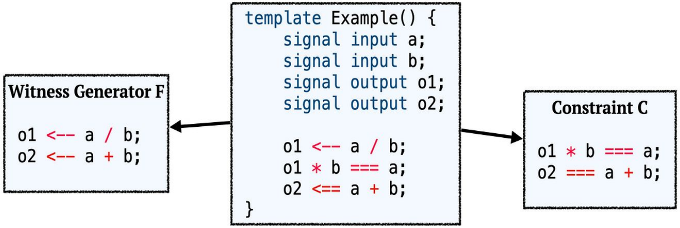
**Loops**

**Template argument**

**Constraints**

**Log**

**Test**



```
template Example() {
    signal input a;
    signal input b;
    signal output o1;
    signal output o2;

    o1 <-- a / b;
    o1 * b === a;
    o2 <== a + b;
}
```

```
o1 <-- a / b;
o2 <-- a + b;
```

```
o1 * b === a;
o2 === a + b;
```

**code:** */circom_advanced*

## *Challenge*

**Developer challenge:**

Create a circom implementation for a simple 3x3 sudoku

## *Links, Resources, Literature*

Circom tutorial:
*https://www.rareskills.io/post/circom-tutorial*

Circomlib:
https://github.com/iden3/circomlib

Zero knowledge puzzles:
https://github.com/RareSkills/zero-knowledge-puzzles

ZK book:
*https://github.com/RareSkills/zk-book*

Circom 101
https://circom.erhant.me/

License: Apache 2.0

## *Happy Hunting for the SNARK :)*

## *Q & A*

Daniel Szego
In: https://www.linkedin.com/in/daniel-szego/