

Circom

December 05, 2024

Distributed Lab

- 🌐 zkdl-camp.github.io
- /github.com/ZKDL-Camp



Plan

1 Introduction

2 Circom

Introduction

Why do we need ZK?

Why do we need ZK?

Option

Solution to privacy

Why do we need ZK?

Option

Solution to privacy

Example

1. *I know the private key that corresponds to this public key*
2. *I know a private key that corresponds to a public key from this list*

Why do we need ZK?

Option

Solution to privacy

Example

1. *I know the private key that corresponds to this public key*
2. *I know a private key that corresponds to a public key from this list*

Option

Solution to scalability

Why do we need ZK?

Option

Solution to privacy

Example

1. *I know the private key that corresponds to this public key*
2. *I know a private key that corresponds to a public key from this list*

Option

Solution to scalability

Example

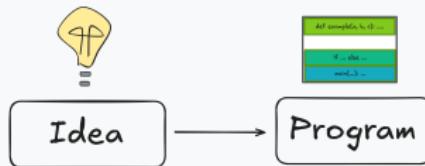
This is the hash of a blockchain block that does not produce negative balances

Using ZKP

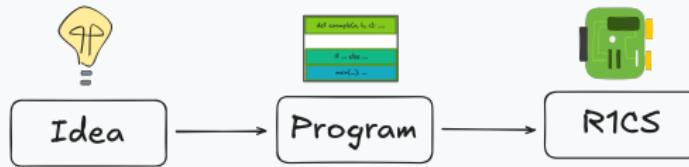
Using ZKP



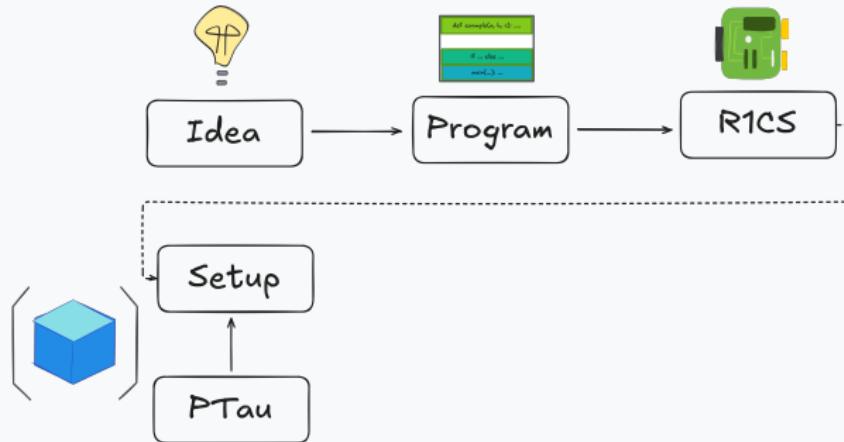
Using ZKP



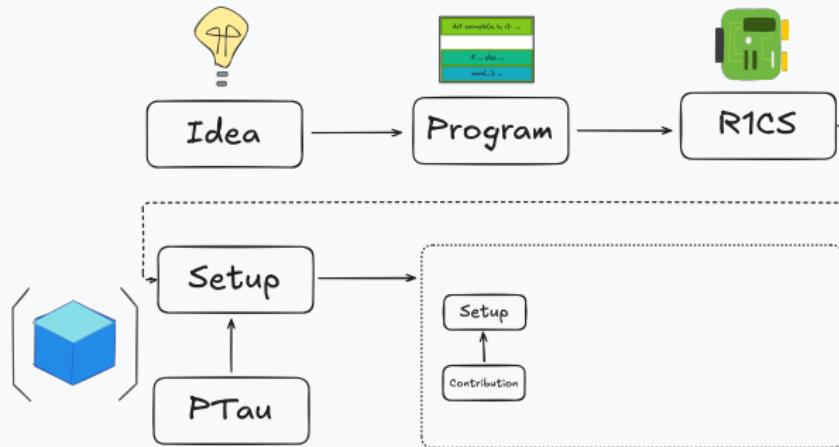
Using ZKP



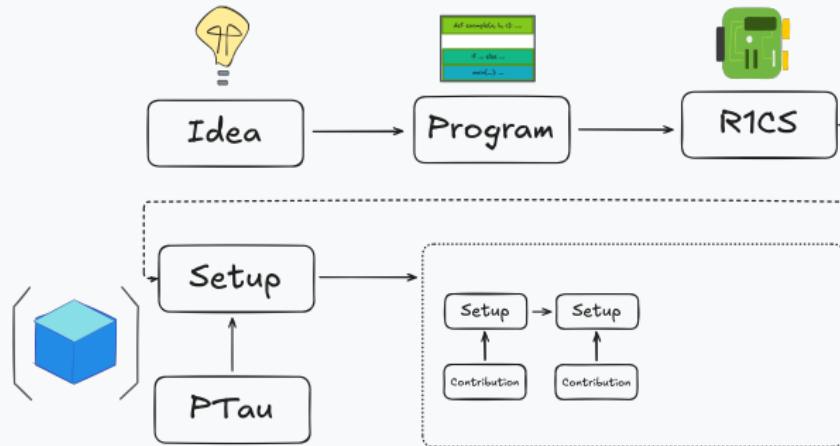
Using ZKP



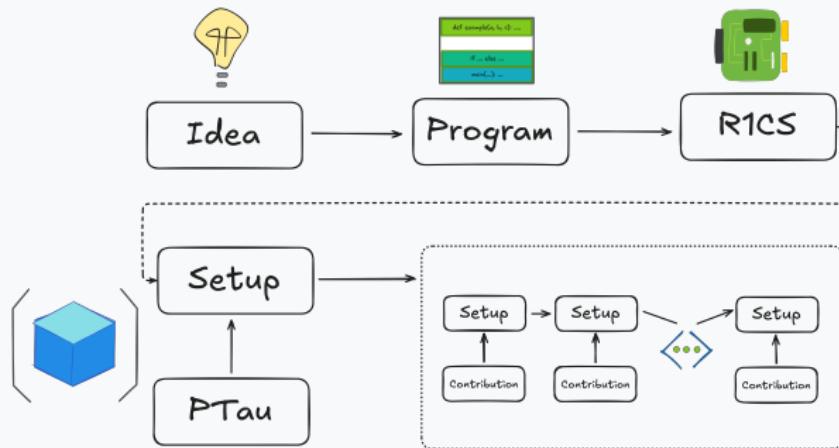
Using ZKP



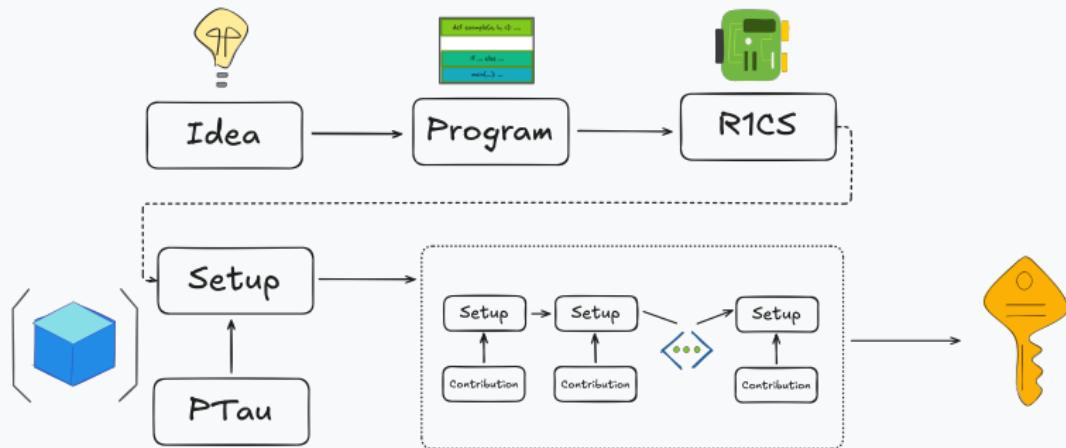
Using ZKP



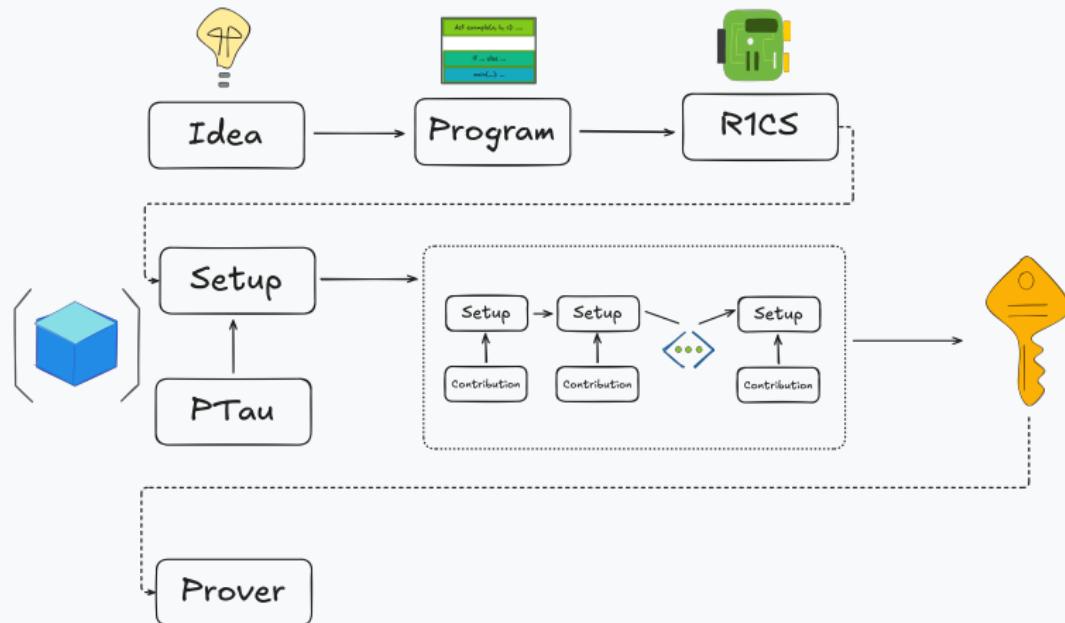
Using ZKP



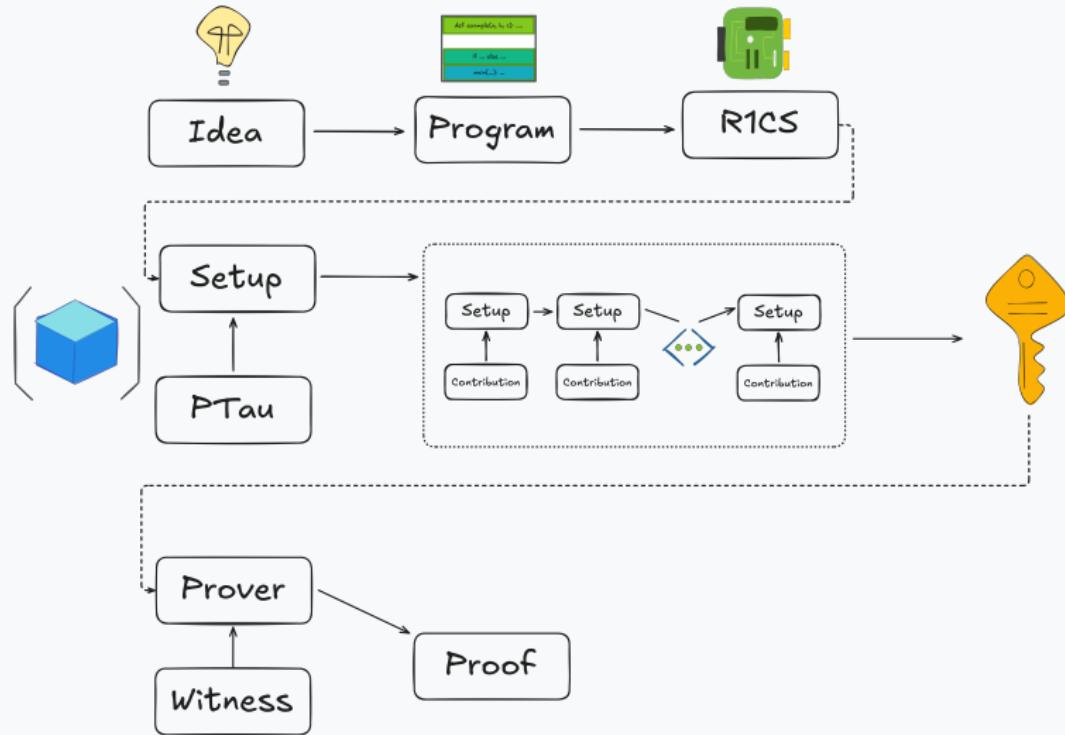
Using ZKP



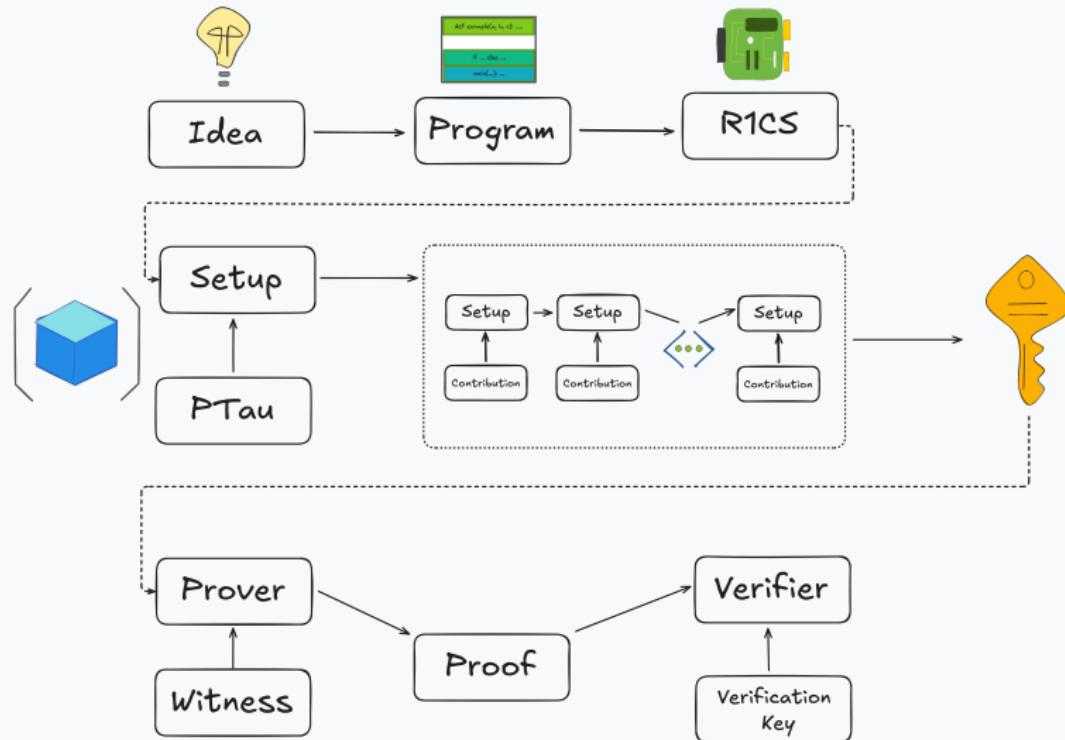
Using ZKP



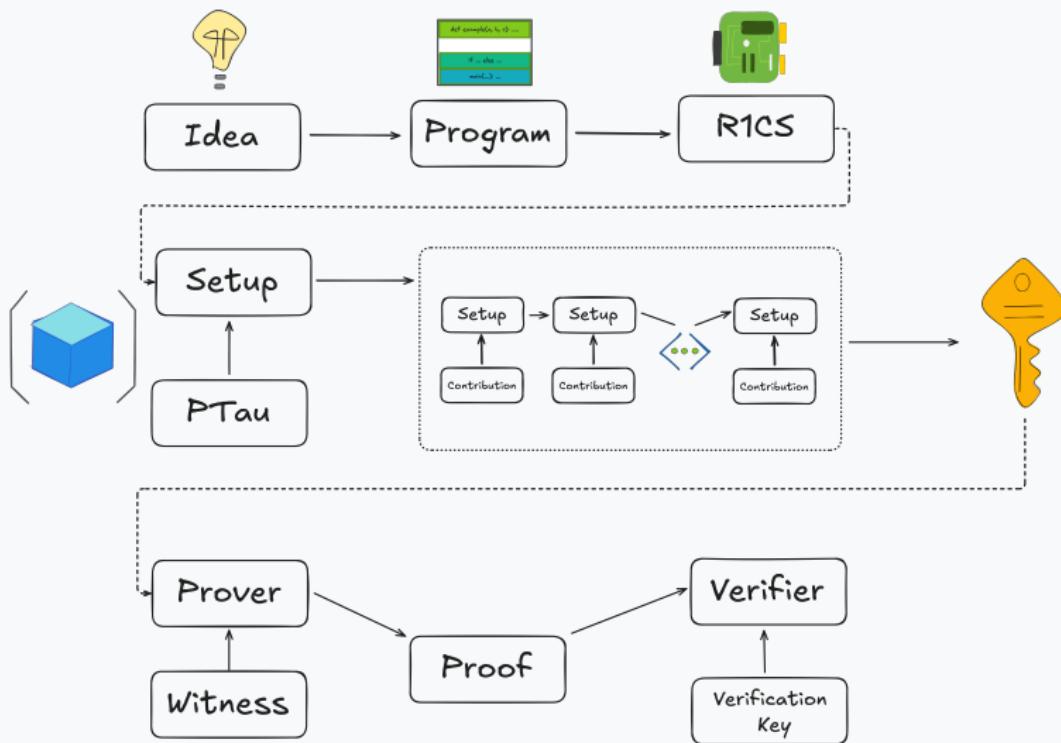
Using ZKP



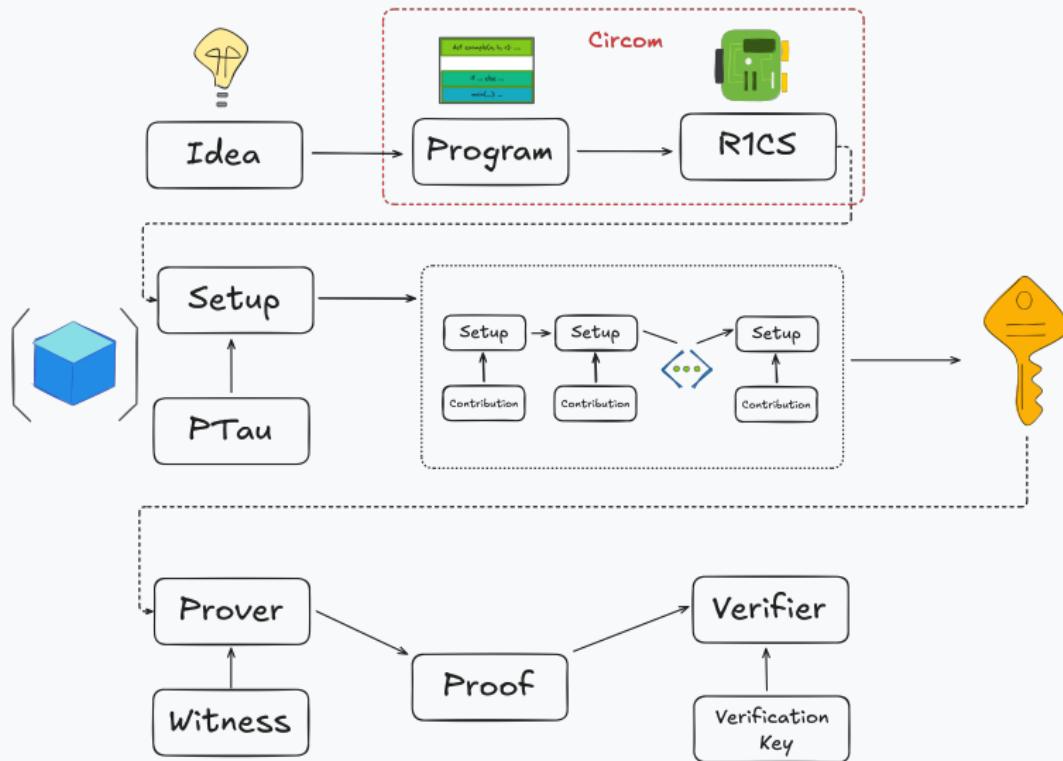
Using ZKP



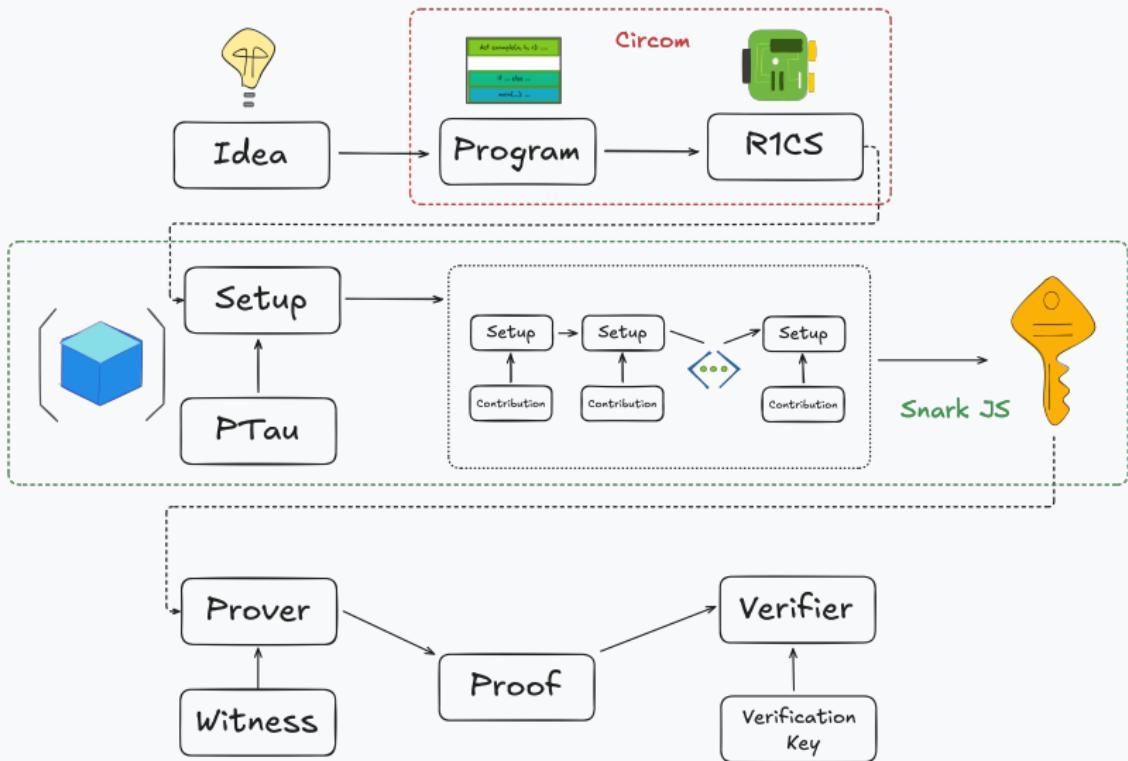
Toolchain



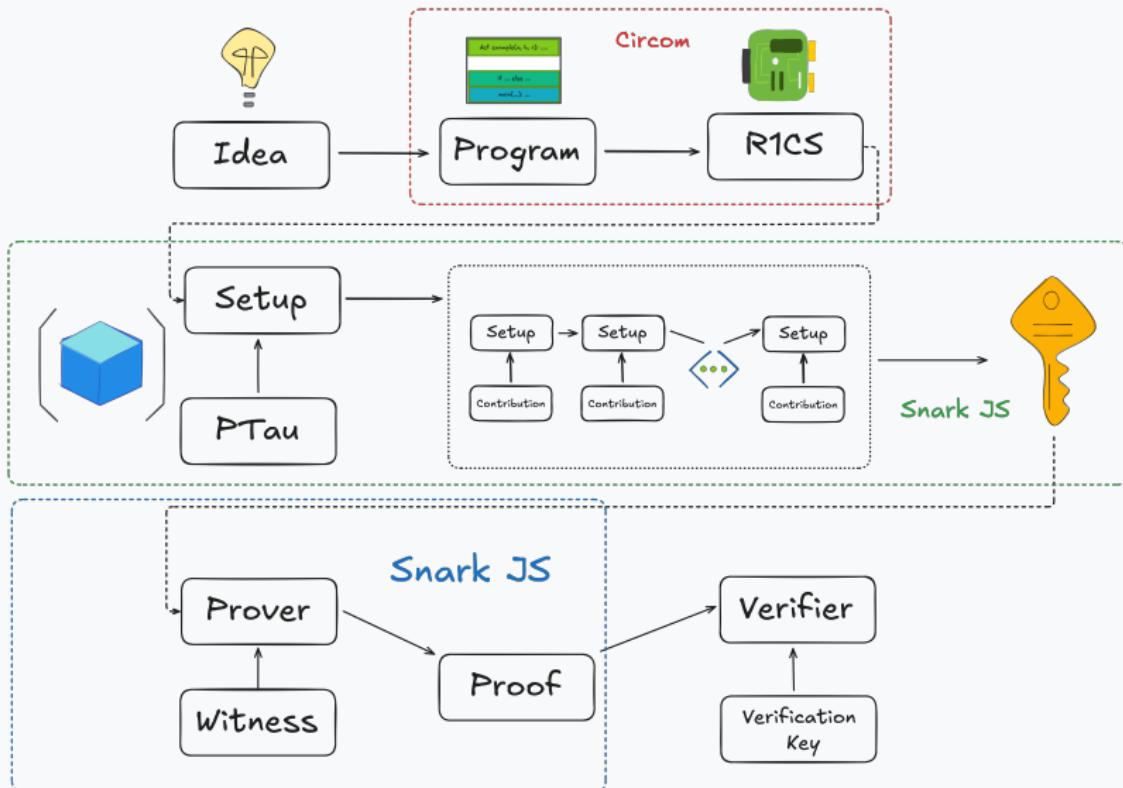
Toolchain



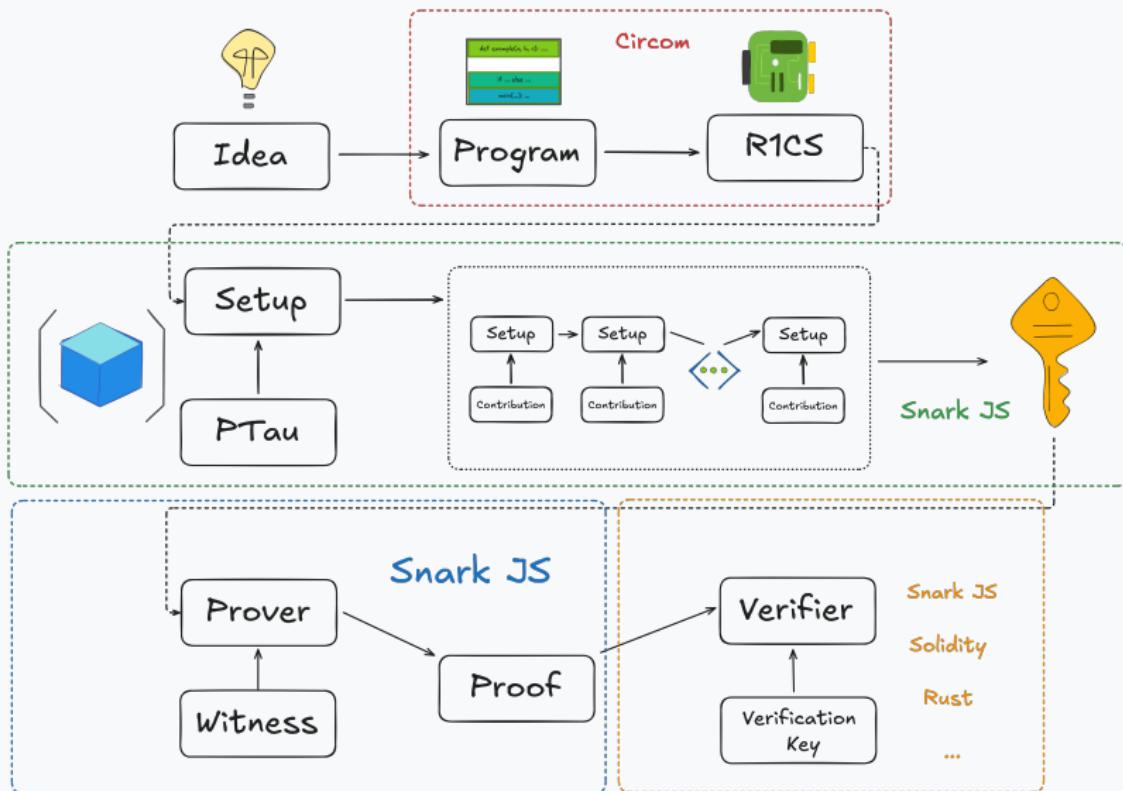
Toolchain



Toolchain



Toolchain



Circom

Previously on ZKDL Camp

Probably you can recall the function

```
def r(x1: F, x2: F, x3: F) -> F:  
    return x2 * x3 if x1 else x2 + x3
```

Previously on ZKDL Camp

Probably you can recall the function

```
def r(x1: F, x2: F, x3: F) -> F:  
    return x2 * x3 if x1 else x2 + x3
```

That can be expressed as:

$$r = x_1 \times (x_2 \times x_3) + (1 - x_1) \times (x_2 + x_3)$$

Previously on ZKDL Camp

Probably you can recall the function

```
def r(x1: F, x2: F, x3: F) -> F:  
    return x2 * x3 if x1 else x2 + x3
```

That can be expressed as:

$$r = x_1 \times (x_2 \times x_3) + (1 - x_1) \times (x_2 + x_3)$$

We need a boolean restriction for x_1 :

$$x_1 \times (1 - x_1) = 0$$

Previously on ZKDL Camp

Probably you can recall the function

```
def r(x1: F, x2: F, x3: F) -> F:  
    return x2 * x3 if x1 else x2 + x3
```

That can be expressed as:

$$r = x_1 \times (x_2 \times x_3) + (1 - x_1) \times (x_2 + x_3)$$

We need a boolean restriction for x_1 :

$$x_1 \times (1 - x_1) = 0$$

Thus, the next constraints can be build:

$$x_1 \times x_1 = x_1 \quad (\text{binary check}) \tag{1}$$

$$x_2 \times x_3 = \text{mult} \tag{2}$$

$$x_1 \times \text{mult} = \text{selectMult} \tag{3}$$

$$(1 - x_1) \times (x_2 + x_3) = r - \text{selectMult} \tag{4}$$

Previously on ZKDL Camp

The witness vector: $\mathbf{w} = (1, r, x_1, x_2, x_3, \text{mult}, \text{selectMult})$.

Previously on ZKDL Camp

The witness vector: $\mathbf{w} = (1, r, x_1, x_2, x_3, \text{mult}, \text{selectMult})$. The coefficients vectors:

$$\begin{aligned}\mathbf{a}_1 &= (0, 0, 1, 0, 0, 0, 0), & \mathbf{b}_1 &= (0, 0, 1, 0, 0, 0, 0), & \mathbf{c}_1 &= (0, 0, 1, 0, 0, 0, 0) \\ \mathbf{a}_2 &= (0, 0, 0, 1, 0, 0, 0), & \mathbf{b}_2 &= (0, 0, 0, 0, 1, 0, 0), & \mathbf{c}_2 &= (0, 0, 0, 0, 0, 1, 0) \\ \mathbf{a}_3 &= (0, 0, 1, 0, 0, 0, 0), & \mathbf{b}_3 &= (0, 0, 0, 0, 0, 1, 0), & \mathbf{c}_3 &= (0, 0, 0, 0, 0, 0, 1) \\ \mathbf{a}_4 &= (1, 0, -1, 0, 0, 0, 0), & \mathbf{b}_4 &= (0, 0, 0, 1, 1, 0, 0), & \mathbf{c}_4 &= (0, 1, 0, 0, 0, 0, -1)\end{aligned}$$

Previously on ZKDL Camp

The witness vector: $\mathbf{w} = (1, r, x_1, x_2, x_3, \text{mult}, \text{selectMult})$. The coefficients vectors:

$$\mathbf{a}_1 = (0, 0, 1, 0, 0, 0, 0), \quad \mathbf{b}_1 = (0, 0, 1, 0, 0, 0, 0), \quad \mathbf{c}_1 = (0, 0, 1, 0, 0, 0, 0)$$

$$\mathbf{a}_2 = (0, 0, 0, 1, 0, 0, 0), \quad \mathbf{b}_2 = (0, 0, 0, 0, 1, 0, 0), \quad \mathbf{c}_2 = (0, 0, 0, 0, 0, 1, 0)$$

$$\mathbf{a}_3 = (0, 0, 1, 0, 0, 0, 0), \quad \mathbf{b}_3 = (0, 0, 0, 0, 0, 1, 0), \quad \mathbf{c}_3 = (0, 0, 0, 0, 0, 0, 1)$$

$$\mathbf{a}_4 = (1, 0, -1, 0, 0, 0, 0), \quad \mathbf{b}_4 = (0, 0, 0, 1, 1, 0, 0), \quad \mathbf{c}_4 = (0, 1, 0, 0, 0, 0, -1)$$

Using the arithmetic in a large \mathbb{F}_p , consider the following values:

$$x_1 = 1, \quad x_2 = 3, \quad x_3 = 4$$

Verifying the constraints:

$$1. \quad x_1 \times x_1 = x_1 \quad (1 \times 1 = 1)$$

$$2. \quad x_2 \times x_3 = \text{mult} \quad (3 \times 4 = 12)$$

$$3. \quad x_1 \times \text{mult} = \text{selectMult} \quad (1 \times 12 = 12)$$

$$4. \quad (1 - x_1) \times (x_2 + x_3) = r - \text{selectMult} \quad (0 \times 7 = 12 - 12)$$

Previously on ZKDL Camp

By Groth16 Protocol the verifier ...

Previously on ZKDL Camp

By Groth16 Protocol the verifier should check the following condition:

$$e(\pi_L, \pi_R) = e(g_1^\alpha, g_2^\beta)e(\pi_{\text{io}}, g_2^\gamma)e(\pi_O, g_2^\delta)$$

Previously on ZKDL Camp

By Groth16 Protocol the verifier should check the following condition:

$$e(\pi_L, \pi_R) = e(g_1^\alpha, g_2^\beta)e(\pi_{\text{io}}, g_2^\gamma)e(\pi_O, g_2^\delta)$$

Recall

For BN254 (BN128), we have:

- Left inputs to e is of form $(x, y) \in \mathbb{G}_1$ — regular curve.
- Right inputs to e is of form $((x_1, y_1), (x_2, y_2)) \in \mathbb{G}_2$ — “complex” curve, consisting of two \mathbb{F}_{p^2} coordinates.
- $e(g_1^\alpha, g_2^\beta)$ is of form $(x_1, \dots, x_{12}) \in \mathbb{F}_{p^{12}}$

Thank you for your attention



 zkdl-camp.github.io
 github.com/ZKDL-Camp

