

1. Estructura del Circuito

El circuito (`prueba.circom`) fue diseñado utilizando Circom 2.1.6 para realizar la siguiente verificación:

Verificar que un usuario conoce dos números secretos a y b , tales que se cumple:

$$c = (a^2 + b^2) \bmod p$$

Donde:

a y b son entradas privadas (inputs secretos del usuario).

p es un input público (un número primo conocido).

c es el output público (resultado de la operación, que cualquier verificador puede ver y comprobar).

Descripción de los componentes:

Square()

Calcula el cuadrado de un número ($\text{out} = \text{in} * \text{in}$).

Mod()

Calcula el módulo de un número respecto a p , sin usar directamente $\%$ (porque Circom no lo permite).

Utiliza un valor intermedio q para expresar la relación:

$$\text{in} === q * p + \text{out}$$

Se asegura que $\text{out} < p$ usando el comparador LessThan de circomlib.

Circuito Principal: SquareSumMod()

Recibe como inputs privados:

a y b .

Recibe como input público:

p .

Calcula $a^2 + b^2$ y lo reduce módulo p , exponiendo el resultado c como output público.

2. Proceso de Generación de Pruebas

El proceso completo de generación de pruebas usa la herramienta snarkjs.

Pasos detallados:

Compilación del circuito Circom

Genera los archivos .r1cs, .wasm, y .sym:

```
circom prueba.circom --r1cs --wasm --sym -o ../build -I .
```

Generación del witness

Es el testigo que contiene todos los valores intermedios, demostrando que el cálculo fue correcto sin revelar las entradas privadas.

El witness representa los valores intermedios que prueban el cálculo correcto sin revelar a ni b:

```
node prueba_js/generate_witness.js prueba_js/prueba.wasm input.json witness.wtns
```

Ejemplo de input.json:

```
{  
  "a": 3,  
  "b": 4,  
  "p": 5  
}
```

Ceremonia de Poderes de Tau (Trusted Setup - Phase 1 y 2)

Prepara los archivos para la generación de claves de prueba:

```
snarkjs powersoftau new bn128 12 pot12_0000.ptau
```

```
snarkjs powersoftau contribute pot12_0000.ptau pot12_0001.ptau --name="Ariel"
```

```
snarkjs powersoftau prepare phase2 pot12_0001.ptau pot12_final.ptau
```

Generación de las claves zkey

Crea el archivo final .zkey que contiene el sistema para generar pruebas:

```
snarkjs groth16 setup ../build/prueba.r1cs pot12_final.ptau prueba_0000.zkey
```

```
snarkjs zkey contribute prueba_0000.zkey prueba_final.zkey --name="Ariel"
```

Exportación de la clave de verificación

Se utiliza en la verificación:

```
snarkjs zkey export verificationkey prueba_final.zkey verification_key.json
```

Generación de la prueba

Utiliza el witness y la zkey para generar el proof.json:

```
snarkjs groth16 prove prueba_final.zkey witness.wtns proof.json public.json
```

3. Proceso de Verificación

Se realizó la verificación de dos maneras: en Node.js y en el navegador web.

Verificación en Node.js

Se encuentra en el archivo verificar.js.

Se ejecuta con node verificar.js

Verificación en Navegador

Se encuentran en los archivos:

index.html

script.js

4. Ejemplos de Uso con Valores Concretos

Archivo input.json:

```
{  
  "a": 3,  
  "b": 4,  
  "p": 5  
}
```

Proceso

$$a^2 + b^2 = 3^2 + 4^2 = 9 + 16 = 25$$

$$25 \bmod 5 = 0$$

Entonces $c = 0$

Public.json generado:

[


"0",


"5"


]

Salida en Node.js:

Con "a": 3,"b": 4

 Verificando prueba...

 Input público p = 5


 Output público c = 0

 ¡Prueba válida!

Salida en el navegador:

Con "a": 3,"b": 4

 Verificando prueba...

 p = 5

 c = 0

 ¡Prueba válida!

ESTRUCTURA DEL PROYECTO

1P BLOCKCHAIN/

- ├── circuits/
 - | ├── prueba.circom
 - | └── (otros archivos circom...)
- ├── build/
 - | ├── prueba.r1cs
 - | ├── prueba.wasm
 - | └── (archivos de compilación)
- ├── zkbuild/
 - | ├── input.json
 - | ├── proof.json
 - | ├── public.json
 - | ├── verification_key.json
 - | ├── prueba_final.zkey
 - | ├── verificar.js
 - | ├── index.html
 - | ├── script.js
 - | └── (otros archivos necesarios)
- ├── DOC.md
- └── README.md