Estructura del circuito

El archivo circom define un template Main que verifica que un usuario conoce dos números secretos a y b, y que estos cumplen la relación $c= (a^2 + b^2)\%p$, donde p=37 (un número primo publico) y c es una salida pública. Se utiliza el componente LessThan de la biblioteca circomlib para asegurar que $0 \leq a < p$ y $0 \leq b < p$. Aunque matemáticamente la operación $(a^2 + b^2)\%p$ es válida para cualquier valor de a y b, estas validaciones se incluyen en la implementación porque permite trabajar con valores pequeños, lo cual simplifica el circuito, reduciendo la cantidad de constraints. También previene posibles overflows al limitar los valores del campo finito lo que conllevaría a resultados incorrectos y prevenir ataques con entradas maliciosas. Si a o b superan el valor de p, el circuito genera una restricción incumplida y rechaza la entrada.

Se calcula a^2 y b^2 y se suma ambos valores. Luego, se aplica reducción modular usando el teorema de la división:

sum=q*p+c, donde q es cociente entero, c es el residuo y $0 \leq c < p$.

La restricción sum === q*p + c garantiza que la relación matemática del módulo sea correcta (es decir, que c sea el residuo de dividir sum por p).

Proceso de Generacion de Pruebas

- El script compile.sh convierte el circuito Circom a WebAssembly y genera archivos R1CS. Tambien se encarga de la generacion de testigos para cada archivo de entrada (ej. input1.json). Si una entrada no cumple las restricciones (ej. b=40 en input2), el proceso falla solo para esa entrada sin afectar a las demás.
- El script setup.sh descarga un archivo .ptau y genera claves de prueba/verificación con snarkjs groth16 setup.
- El script generate_proof.sh usa snarkjs groth16 prove para crear pruebas Groth16 basadas en los testigos.

Proceso de Verificación

- El script verify.sh ejecuta snarkjs groth16 verify para validar las pruebas contra la clave de verificación.
- El archivo verifier.html carga las pruebas y permite verificarlas interactivamente usando snarkjs.groth16.verify

Ejemplos de uso

Caso 1: entrada válida (input1.json)

- Entradas: a=5 y b=6
- Cálculo:

$$a^2 + b^2 = 5^2 + 6^2 = 25 + 36 = 61$$
$$c = 61 \% 37 = 24$$

- Resultado: PRUEBA VÁLIDA

Caso 2: entrada inválida (input2.json)

- Entradas: a=10 y b=40
- Validación: b=40 no cumple con $0 \leq b < 37$
- Resultado: PRUEBA INVÁLIDA ya que se falla en generar el testigo

Caso 3: entrada válida (input3.json)

- Entradas: a=8 y b=3
- Cálculo:

$$a^2 + b^2 = 8^2 + 3^2 = 64 + 9 = 73$$
$$c = 73 \,\%\, 37 = 36$$

- Resultado: PRUEBA VÁLIDA