Estructura del circuito.

```
pragma circom 2.0.0;

template Mod() {
    // Entradas privadas
    signal input a;
    signal input b;

    // Salida pública
    signal output c;

    // Constante del módulo
    var p = 17;

    // Señales auxiliares
    signal a2;
    signal b2;
    signal sum;
    signal q;

    // Operaciones
    a2 <== a * a;
    b2 <== b * b;
    sum <== a2 + b2;

    // Calcular la división entera q = sum / p
    q <-- sum \ p;

    // Calcular el residuo c = sum % p
    c <== sum - q * p;
}

component main = Mod();
```

El circuito Mod está diseñado para verificar la siguiente operación matemática:

$$c = (a^2 + b^2) \, mod \; p$$

En donde, a y b son entradas privadas, p es un número primo fijo definido en el circuito, y c es la salida pública, resultado de la operación.

Luego se definen variables auxiliares, las cuales serán de ayuda para representar el cuadrado de a y de b, la suma de ellas, y el cociente de la división, el cual será utilizado para realizar la operación de módulo.

En las líneas 21 a las 23 se calculan los valores auxiliares los cuales quedan como relaciones aritméticas válidas dentro del R1CS.

La línea 26 calcula el valor del cociente q, utilizando el operador de división entera \.

En la línea 29 finalmente se calcula el residuo de la división, utilizando el teorema de la división el cual dice que $D = d.c + r$

Proceso de generación de pruebas.

1. Compilación del circuito

```
circom circuit.circom --r1cs --wasm --sym -o circuit
```

2. Crear un archivo de entrada input.json en la carpeta circuit_js
3. Dentro del directorio circuit_js, generar el witness

```
node generate_witness.js circuit.wasm input.json witness.wtns
```

4. Iniciar una nueva ceremonía de powers of tau y contribuir a la misma, esto es necesario solo la primera vez.

```
snarkjs powersoftau new bn128 12 pot12_0000.ptau -v
snarkjs powersoftau contribute pot12_0000.ptau pot12_0001.ptau --name="First contribution" -v
snarkjs powersoftau prepare phase2 pot12_0001.ptau pot12_final.ptau -v
```

5. Iniciar la fase 2, generando un zkey, contribuyendo a la ceremonía y exportando la verification key.

```
snarkjs groth16 setup circuit.r1cs pot12_final.ptau circuit_0000.zkey
snarkjs zkey contribute circuit_0000.zkey circuit_0001.zkey --name="1st Contributor Name" -v
snarkjs zkey export verificationkey circuit_0001.zkey verification_key.json
```

6. Generar la prueba

```
snarkjs groth16 prove circuit_0001.zkey circuit_js/witness.wtns proof.json public.json
```

Proceso de verificación.

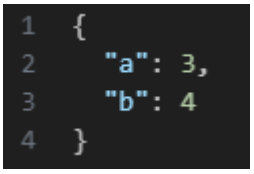
Para verificar la prueba

```
snarkjs groth16 verify verification_key.json public.json proof.json
```

También, dentro del directorio verificador, ejecutar

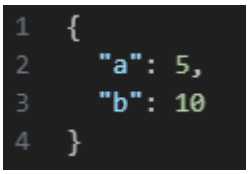
```
node verify.js
```

Ejemplos de uso con valores concretos.



Con a = 3, b = 4, y p = 17 (el cual está definido dentro del archivo) el resultado de $c = (a^2 + b^2) \, mod \; p$ es 8, el cual podemos verificar en el archivo public.json

Con a = 5, b = 10 y p = 17



el resultado es c = 6, como se verifica en el archivo public.json