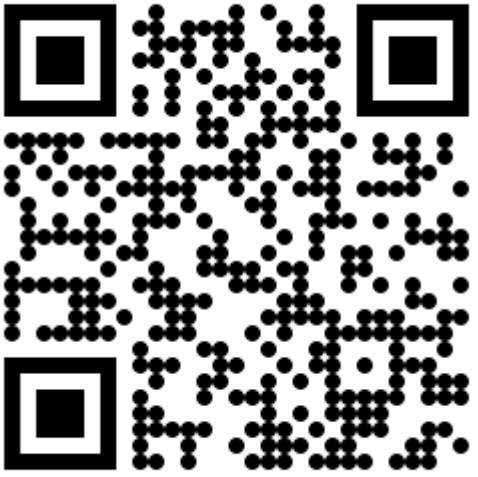


https://n.ethz.ch/~kklier/download/zkp/

Zero-Knowledge Proofs

Exercises Week 5: Circom/SnarkJS Part I

Today: Play around with SNARKs

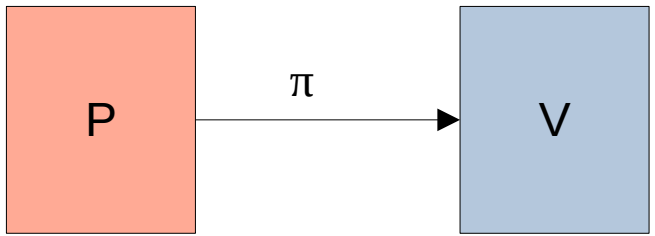
- **S**uccinct
- **N**on-Interactive
- **Ar**gument
- Of **K**nowledge

} **Communication-Efficiency**

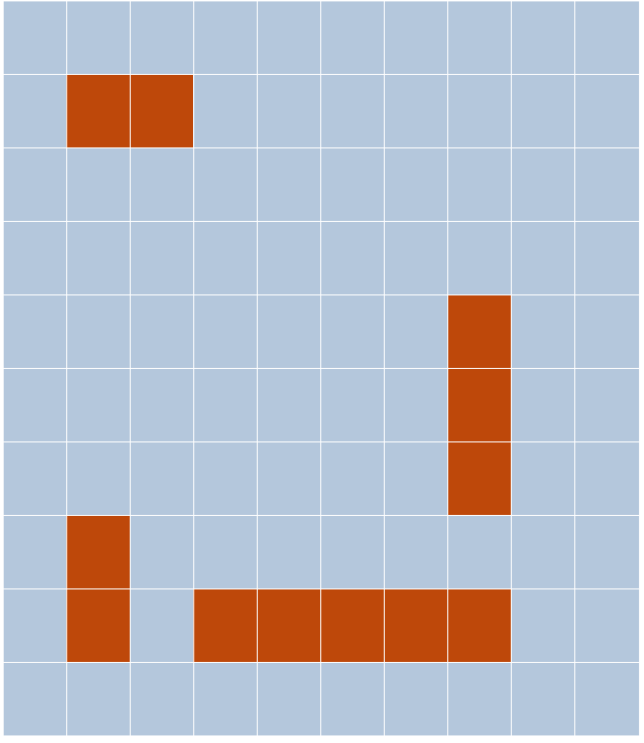
ZCash: SNARK-based crypto currency

roll_up: scaling up Ethereum

Dark Forest: zkSNARK space warfare



Goal: Play Provably Secure “Battleships”



https://c4.wallpaperflare.com/wallpaper/4/181/81/artwork-classic-art-painting-sailing-ship-wallpaper-preview.jpg

$(x, y) \in [10]^2$

hit/miss

https://www.publicdomainpictures.net/pictures/150000/velka/ship-at-sea-1455511625jcd.jpg

CIRCOM & SNARKJS

1. Design your arithmetic circuit and write your circuit using circom
   - Use your own code
   - Use our **safe** templates
2. Compile the circuit to get a low-level representation (R1CS)

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

3. Use snarkjs to compute your witness

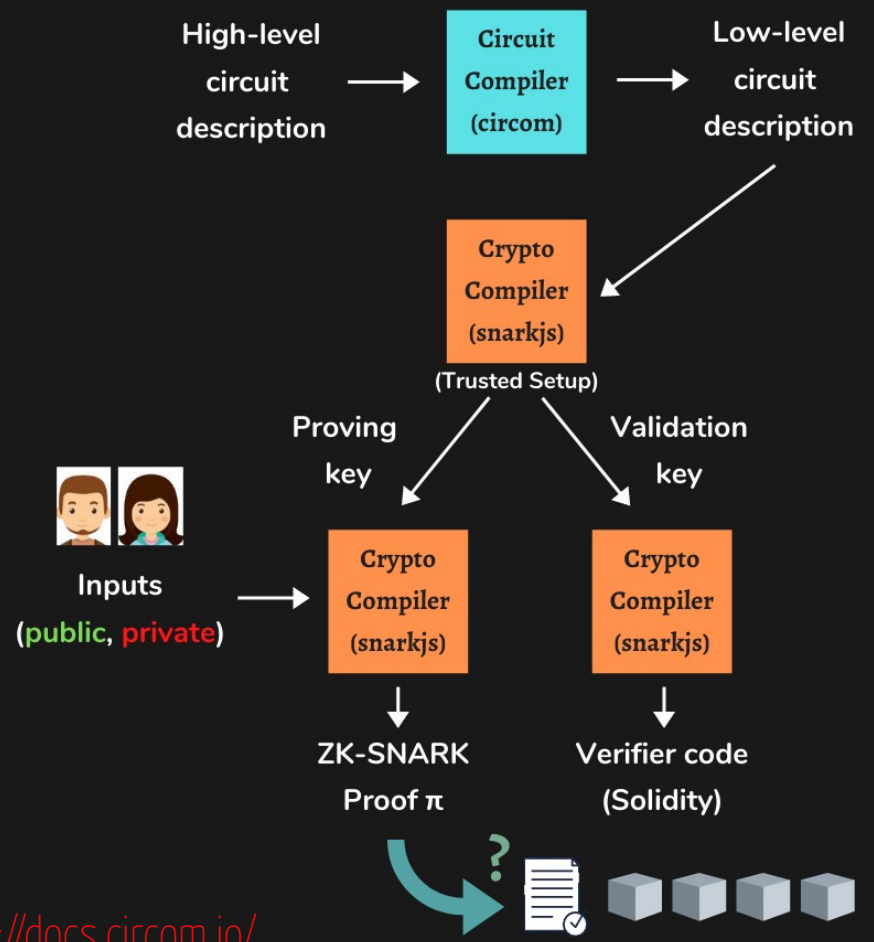
```
$ snarkjs calculatewitness --wasm circuit.wasm
  --input input.json --witness witness.json
```

4. Generate a trusted setup and get your zk-SNARK proof

```
$ snarkjs setup
$ snarkjs proof
```

5. Validate your proof or have a smart-contract validate it!

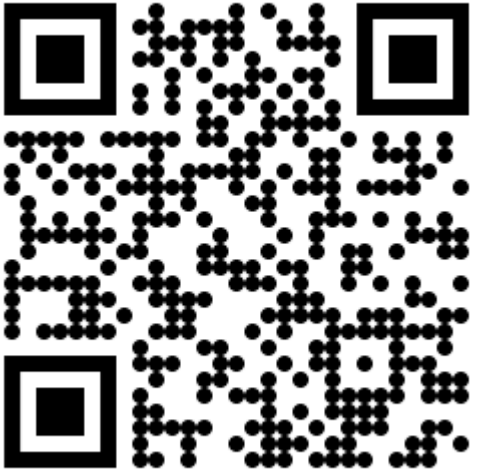
```
$ snarkjs validate
$ snarkjs generateverifier
```



Source: https://docs.circom.io/

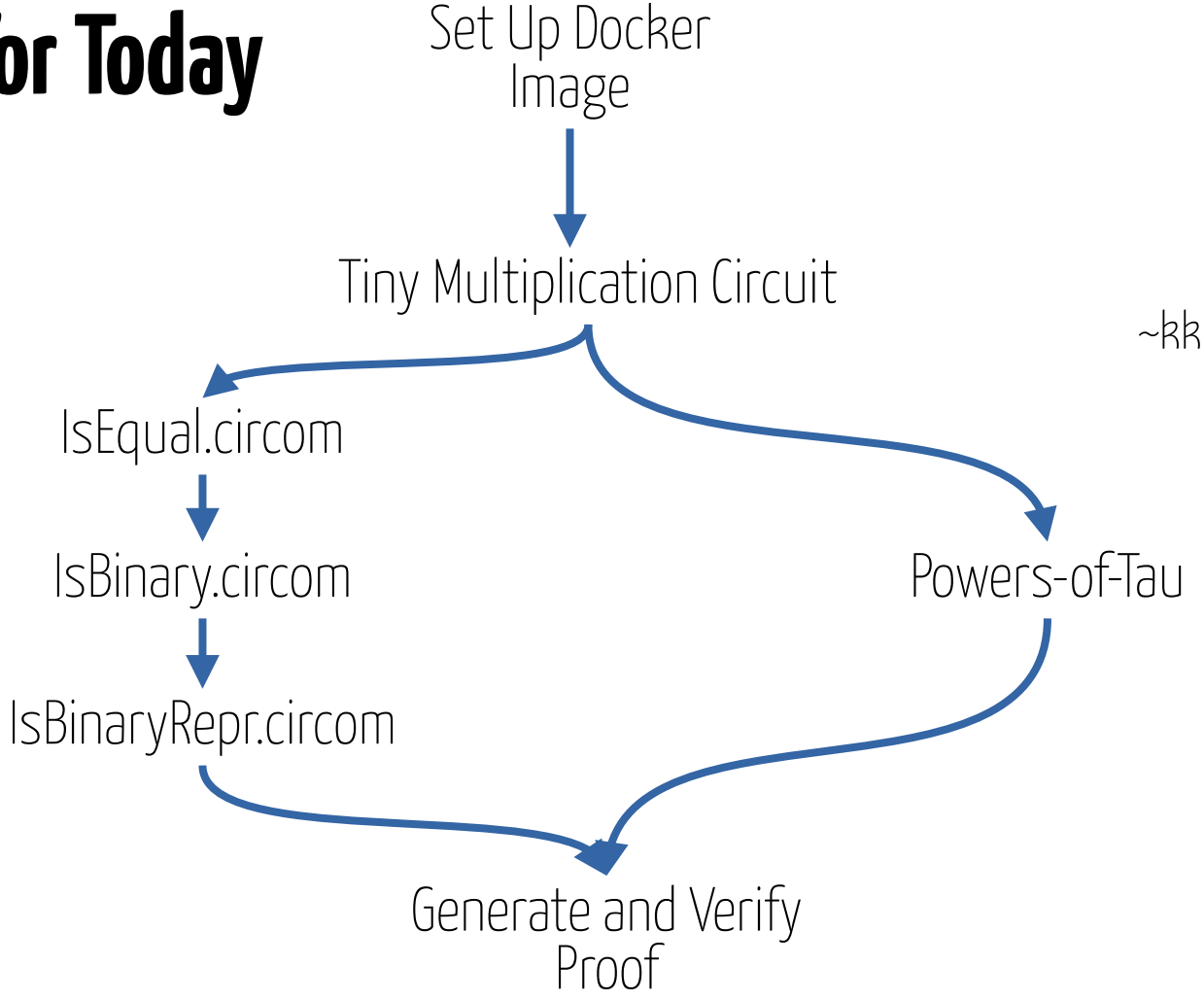
For Your Convenience

- Docker Image (snarkjs.1.1.0.dockerimage)
- compose.yml
- README.html



https://n.ethz.ch/~kklier/download/zkp/

Roadmap for Today



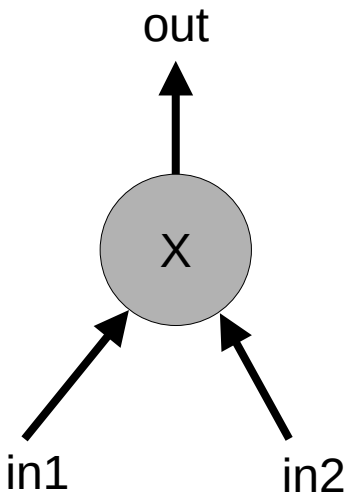
https://n.ethz.ch/~kklier/download/zkp/

Solutions

1. Multiplication

```
template Multiply() {
    signal input in1;
    signal input in2;
    signal output out;

    out <== in1*in2;
}
```

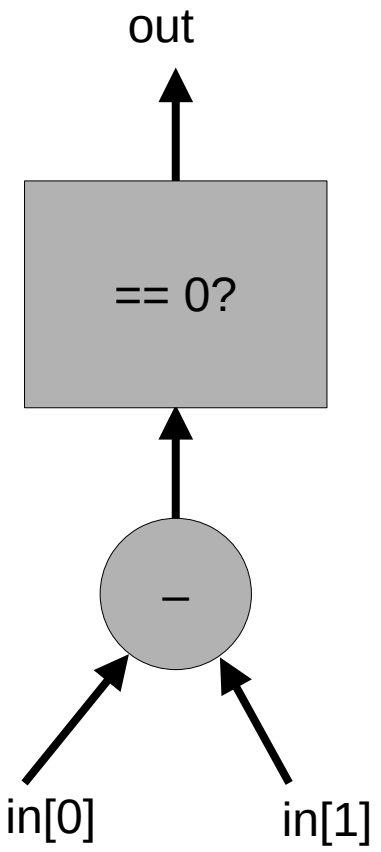


2. IsEqual

```
template IsEqual() {
    signal input in[2];
    signal output out;

    component iszero = IsZero();
    iszero.in <== in[0] - in[1];

    out <== iszero.out;
}
```



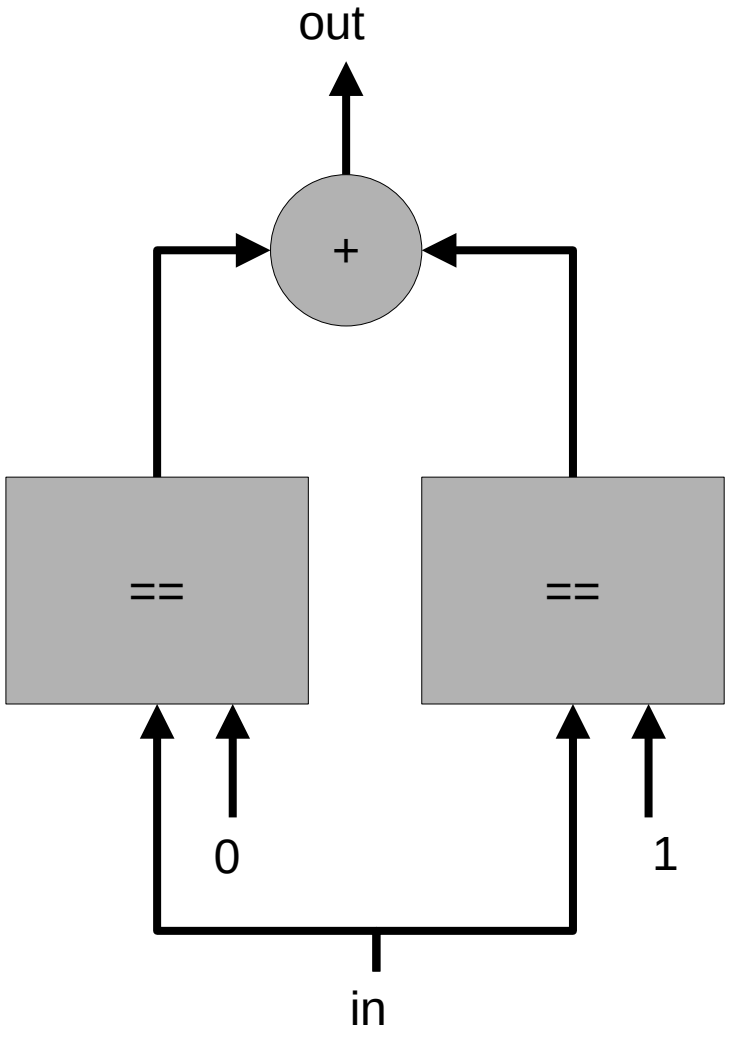
3. IsBinary

```
template IsBinary() {
    signal input in;
    signal output out;

    component is0 = IsEqual();
    is0.in[0] <== in;
    is0.in[1] <== 0;

    component is1 = IsEqual();
    is1.in[0] <== in;
    is1.in[1] <== 1;

    out <== is0.out + is1.out;
}
```



4. IsBinaryRepresentation

```
template IsBinaryRepr(len) {
    signal input in;
    signal input repr[len];
    signal output out;

    component isBinary[len];

    for (var i = 0; i < len; i += 1){
        isBinary[i] = IsBinary();
        isBinary[i].in <== repr[i];
        isBinary[i].out === 1;
    }

    signal intermediate[len];
    intermediate[0] <== repr[0];

    for (var i = 1; i < len; i += 1){
        intermediate[i] <== 2*intermediate[i-1] + repr[i];
    }

    component isEq = IsEqual();
    isEq.in[0] <== intermediate[len-1];
    isEq.in[1] <== in;

    out <== isEq.out;
}
```