A short introduction To Noir

By **Neiman**


Noir

noir

Programming language for cryptography

noir

Programming language for zk-snarks

What is zk-snarks?

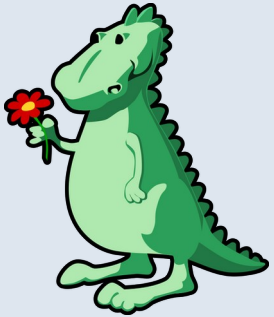
protocol

protocol

prover

protocol

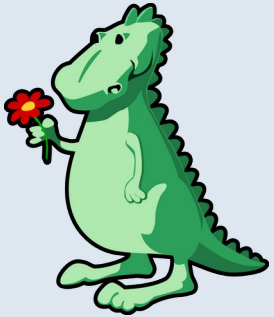
prover



verifier

secret,
data

protocol



prover

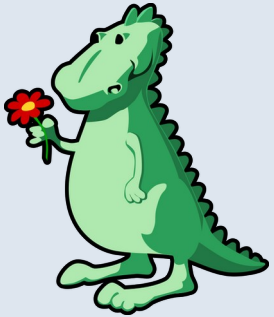
verifier

protocol

secret,
data

proof

prover

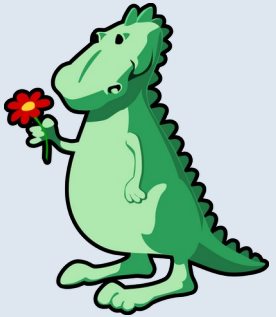


verifier

protocol

secret,
data

proof



proof

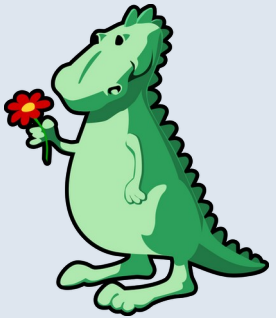
prover

verifier

Protocol 2

secret, data

proof



proof

prover

verifier

3 years of development

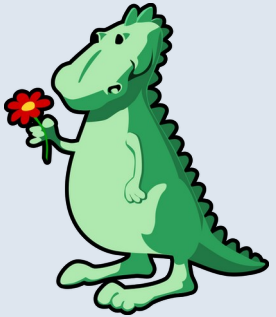
PROGRAM

secret,
data

protocol

proof

proof



prover

verifier

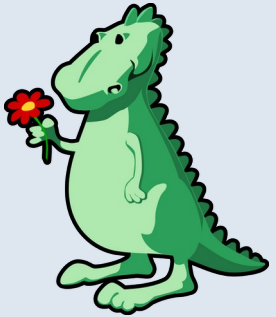
noir

PROGRAM 1

secret,
data

Protocol 1

proof



proof

prover

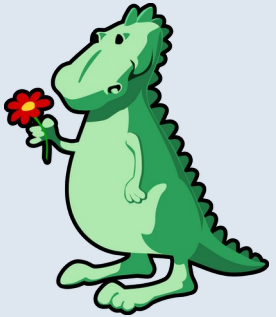
verifier

PROGRAM 2

secret,
data

Protocol 2

proof

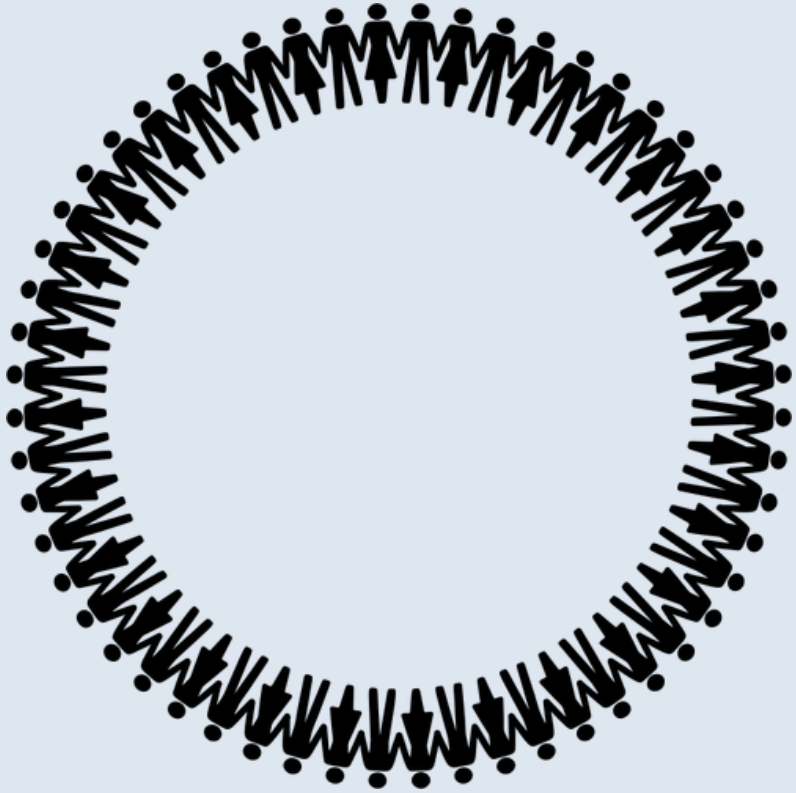


proof

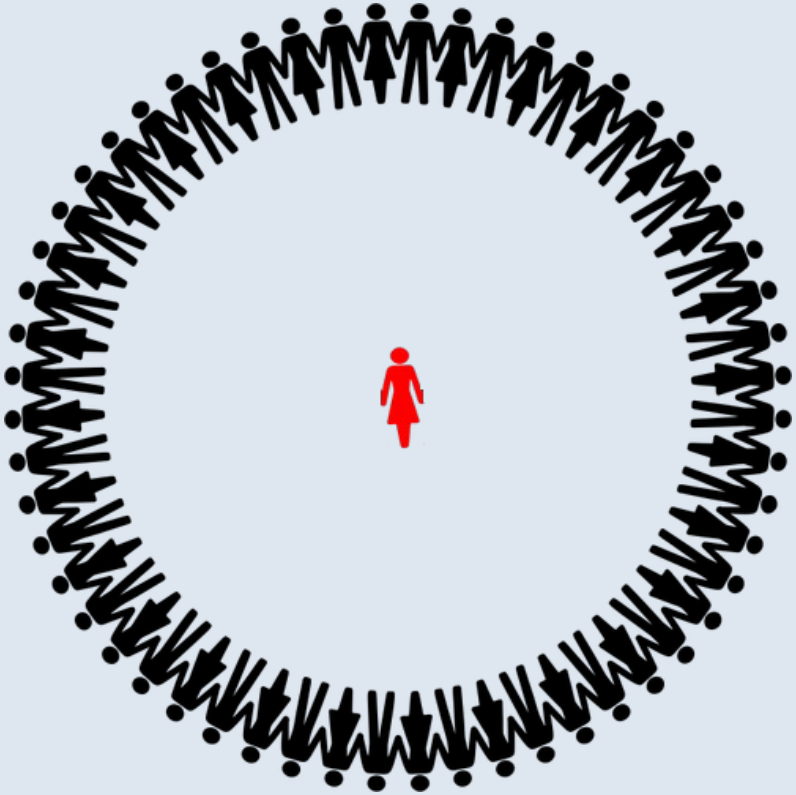
prover

verifier

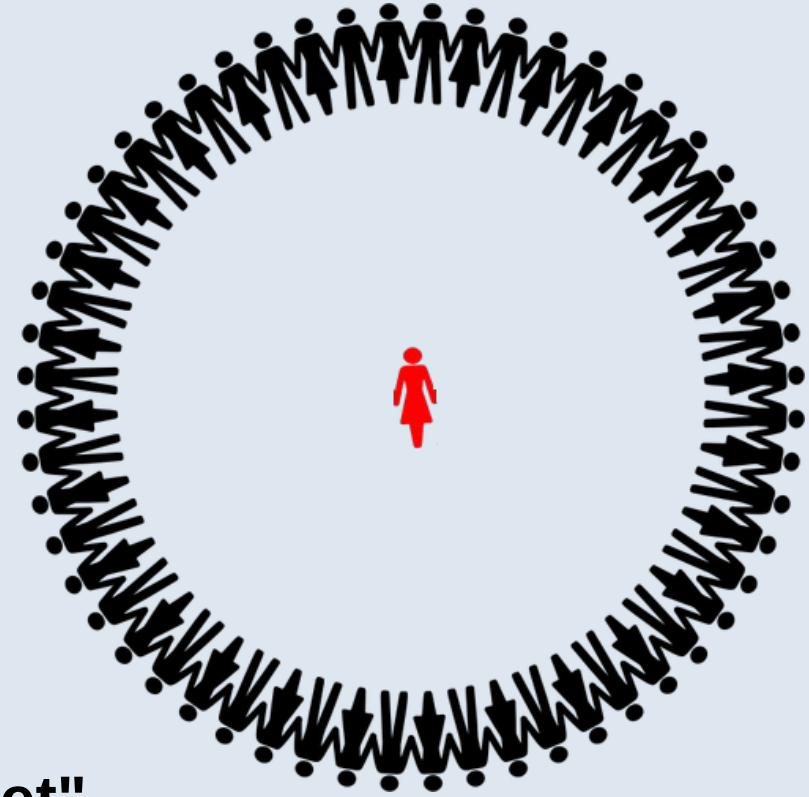
Ring Signatures



Ring Signatures



Ring Signatures



"How to Leak a Secret"

```
fn main(value: pub Field,
        ring2: pub Ring2,
        signature: Signature,
        public_key: PublicKey);
```

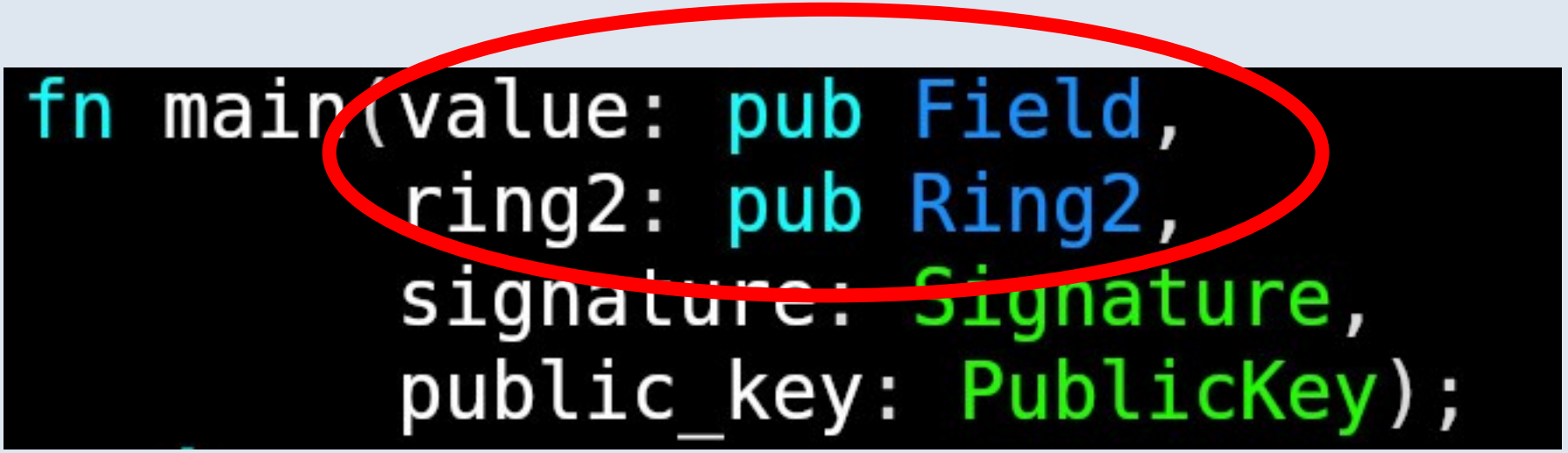
Syntax – Rust like

```
struct Ring2 {
    members: [PublicKey; 2],
}

▶ Compile | Info | Execute | Debug
fn main(value: pub Field, ring2: pub Ring2, public_key: PublicKey, signature: Signature) {
    let mut in_ring: bool = false;
    if (equal_public_keys(pub_key1: public_key, pub_key2: ring2.members[0])) {
        in_ring = true;
    } else if (equal_public_keys(pub_key1: public_key, pub_key2: ring2.members[1])) {
        in_ring = true;
    }

    // Verify signature
    let verify_signature: bool = eddsa::eddsa_verify::<PoseidonHasher>(
        public_key.x,
        public_key.y,
        signature.s,
        signature.rx,
        signature.ry,
        value,
    );

    assert(in_ring, true);
    assert(verify_signature, true);
}
```


fn main(value: pub Field,
ring2: pub Ring2,
signature: Signature,
public_key: PublicKey);

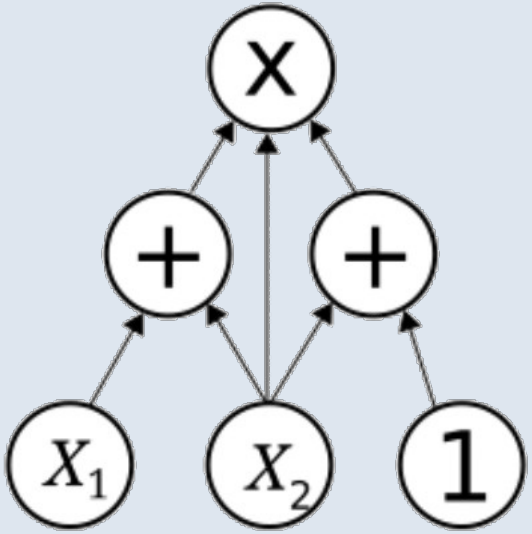
Ring2

```
fn main(value: pub Field,
        public_key: PublicKey,
        signature: Signature,
        ring2: pub Ring2);
```

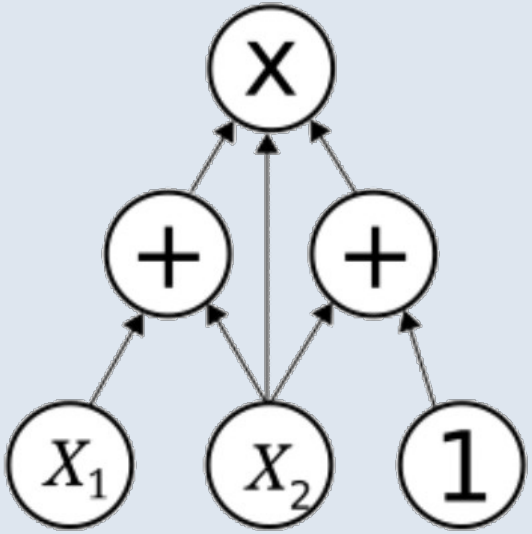
Ring2

```
struct Ring2 {
    members: [PublicKey; 2],
}
```

Circuit



ACIR



Proving backends

Proving Backends

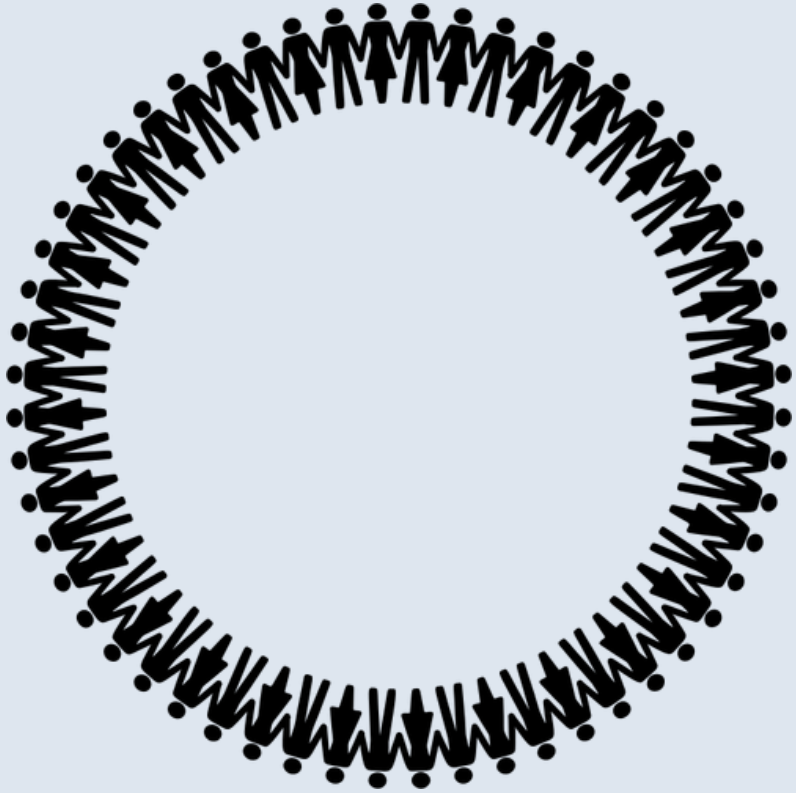
- Barretenberg (UltraHonk / MegaHonk) by Aztec Labs
- Plonky2 by Blocksense
- Plonky2 by Eryx
- Sonobe (Nova, HyperNova) by 0xPARC and PSE
- Plonky3 by Josef (needs updating)
- Halo2 by Ethan (needs updating)
- Groth16 (needs updating)
- Marlin (needs updating)

Barrentenberg

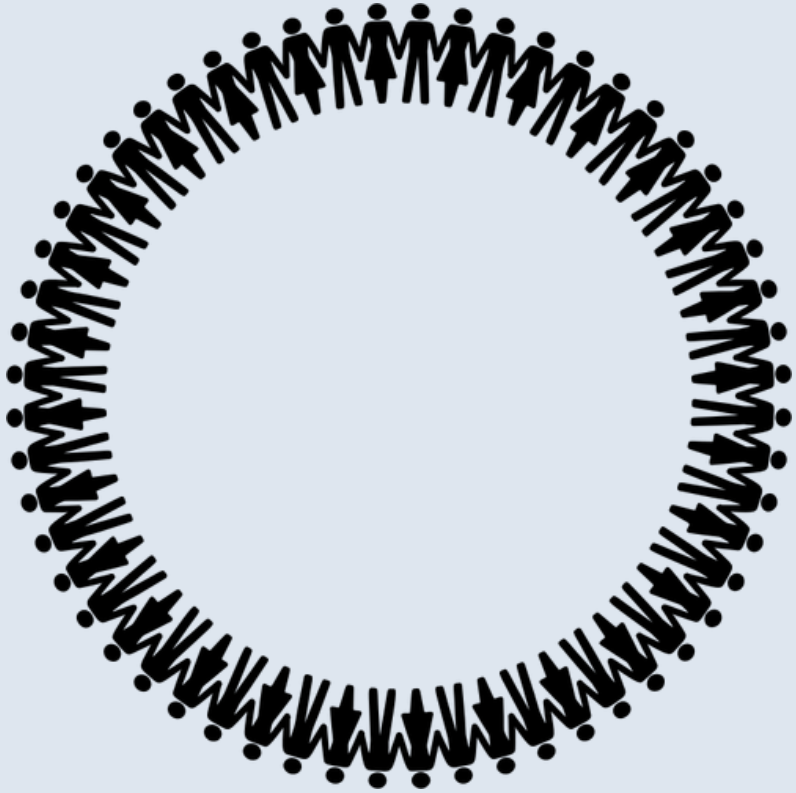
Proving Backends

- Barretenberg (UltraHonk / MegaHonk) by Aztec Labs
- Plonky2 by Blocksense
- Plonky2 by Eryx
- Sonobe (Nova, HyperNova) by 0xPARC and PSE
- Plonky3 by Josef (needs updating)
- Halo2 by Ethan (needs updating)
- Groth16 (needs updating)
- Marlin (needs updating)

Linkable Ring Signatures



Traceable Ring Signatures



- **zero-knowledge: 1985**

- **zero-knowledge: 1985**
- **snark: 1998 TRUSTED SETUP**

- zero-knowledge: 1985
- snark: 1998 TRUSTED SETUP

- zero-knowledge: 1985
- snark: 1998 TRUSTED SETUP
- Applied SNARK: 2012

- zero-knowledge: 1985
- snark: 1998 TRUSTED SETUP
- Applied SNARK: 2012
- STARK: 2018

Use case 1: blockchain

Use case 2:
ID

Alternatives: zk-vm

Alternatives: circom

Thank you!