Learnings in progress

Zero Knowledge Proofs

By Rudi Yardley

https://twitter.com/rudiyardley
https://github.com/ryardley
https://linkedin.com/in/rudiyardley

block hackers

Web3 Consultant

Solidity, Solana, Typescript,

Fullstack

DEXs, CEXs, NFT

22 years eng exp

Blockchain since 2018

NOT A CRYPTOGRAPHER!

LEARNING

WORK IN PROGRESS

ZERO MATH

I AM NOT A CRYPTOGRAPHER

Agenda

Zero Knowledge Primer

Mental Model

Deployment Architecture

Noir vs Circom

Demo

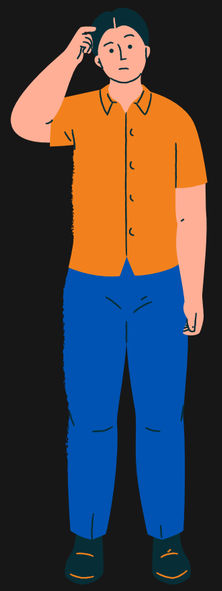
ZERO KNOWLEDGE

ZERO KNOWLEDGE

?

A zero-knowledge protocol is a method by which one party can prove to another party that a given statement is true whilst avoiding conveying any additional information apart from the fact that the statement is indeed true - Wikipedia

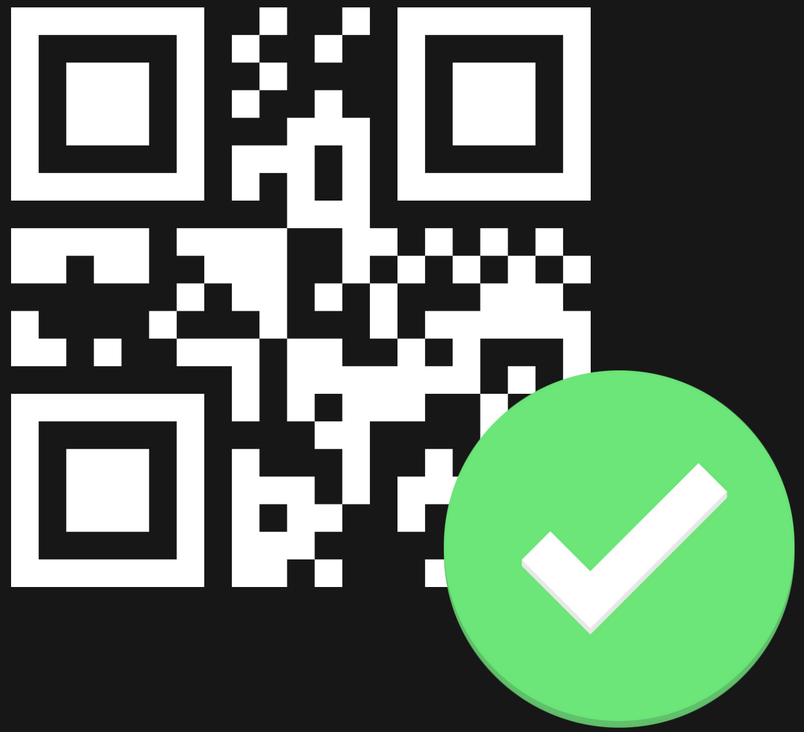
Tell me you're a _____



Without telling me you're a _____

DATA SOVEREIGNTY
AUTHENTICATION
DIGITAL IDS
DIGITAL PASSPORTS
BLOCKCHAIN PRIVACY
SCALABILITY
e-DEMOCRACY
and more...

ARE YOU OVER 18?



PRIVATE PAYMENTS

LOOKING INTO ZERO KNOWLEDGE

LOOKING INTO ZERO KNOWLEDGE

SNARKS

STARKS

LOOKING INTO ZERO KNOWLEDGE

SNARKS

Lagrange Interpolation

Elliptic Curve Cryptography

LOOKING INTO ZERO KNOWLEDGE

STARKS

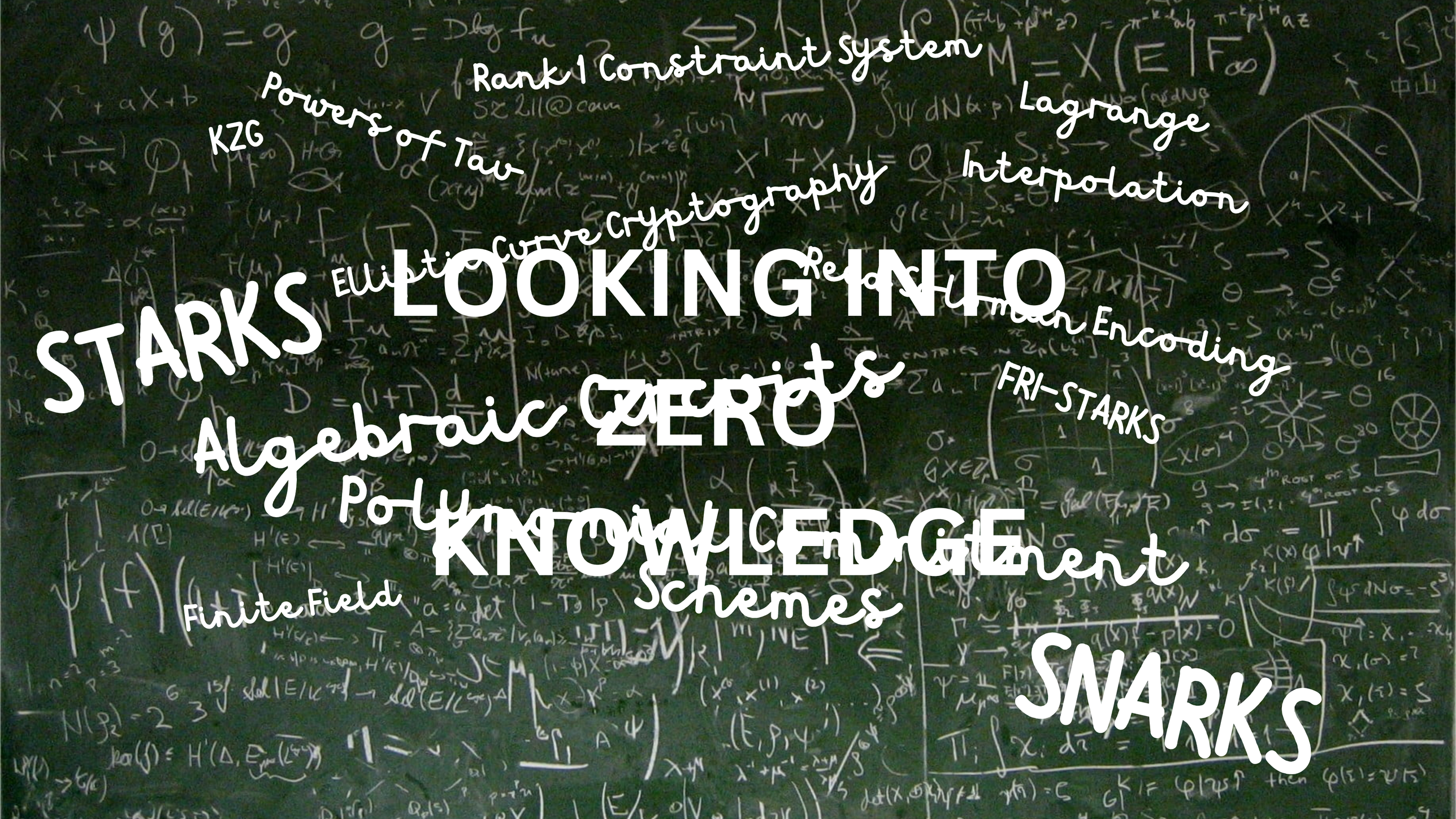
Algebraic

FRI-STARKS

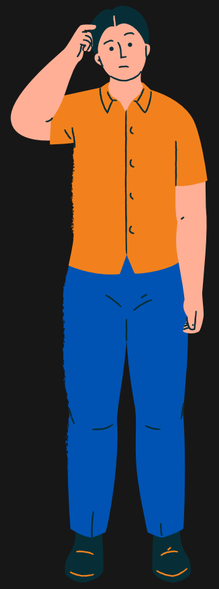
Schemes

SNARKS


Rank 1 Constraint System
Powers of Tau
KZG
Lagrange Interpolation
Elliptic Curve Cryptography
Reed-Solomon Encoding
STARKS
LOOKING INTO
ZERO
KNOWLEDGE
Algebraic
FRI-STARKS
Finite Field
Schemes
SNARKS


LOOKING INTO
ZERO
KNOWLEDGE
STARKS
SNARKS
Rank 1 Constraint System
Powers of Tau
KZG
Elliptic Curve Cryptography
Lagrange Interpolation
Reed-Solomon Encoding
FRI-STARKS
Algebraic Circuits
Polynomial Commitments
Finite Field
Schemes
Element

Tell me you're a _____



Without telling me you're a _____

FUNCTIONS

with

RECEIPTS

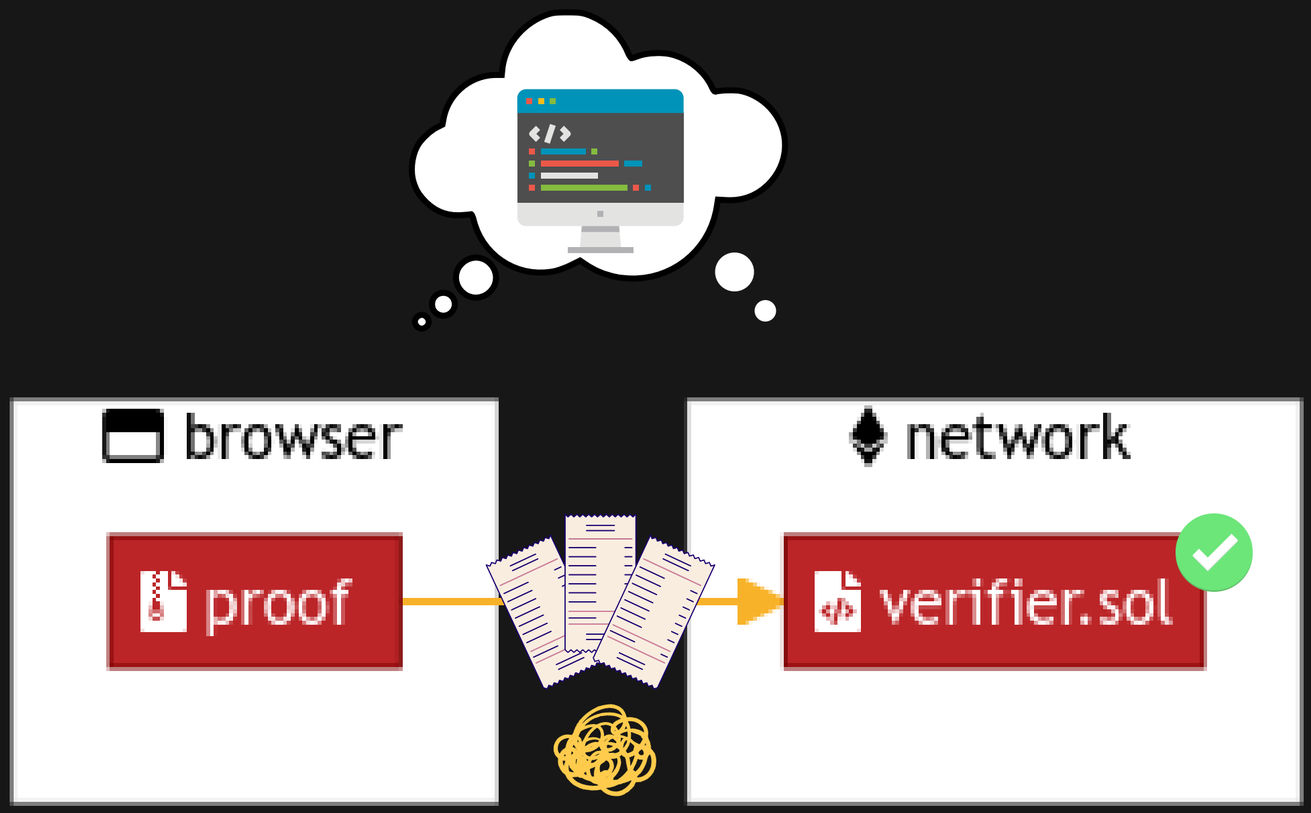
 = hash()

= hash()

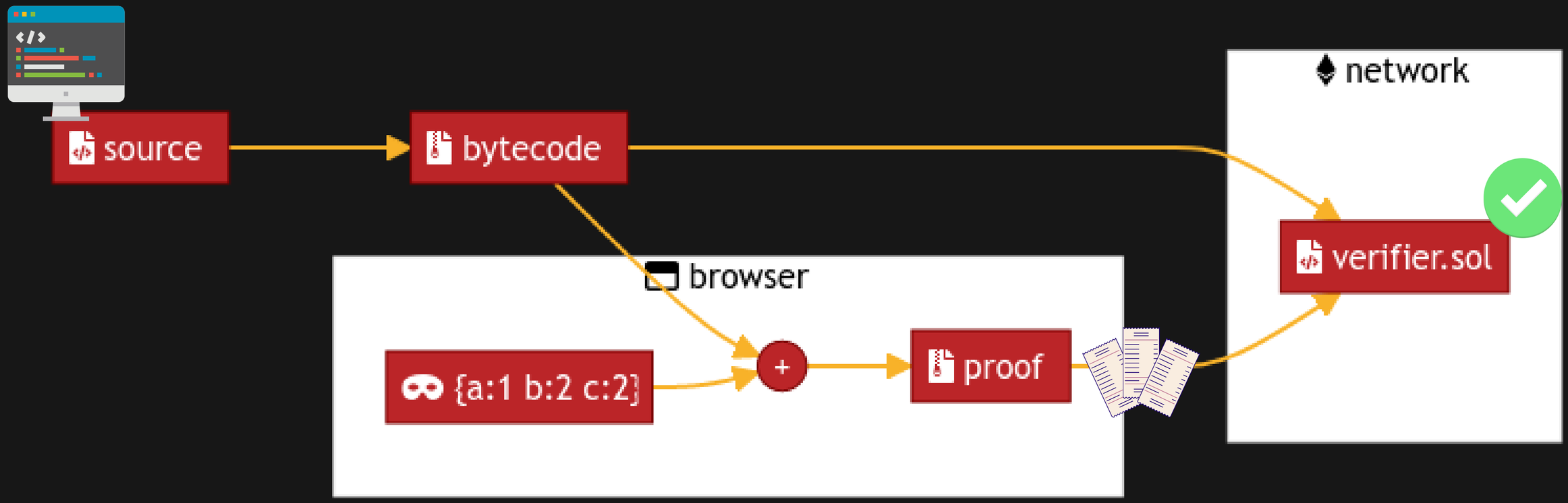
PRIVATE

PUBLIC

Implementation



Deployment


source
bytecode
network
verifier.sol
browser
{a:1 b:2 c:2}
+
proof

SNARKS

Succinct
Non-interactive
ARguments of
Knowledge

Trusted Setup

Cryptographic Randomness
Multiparty Computation Ceremony
1 must be honest

Trusted Setup

Proving Systems

Groth16

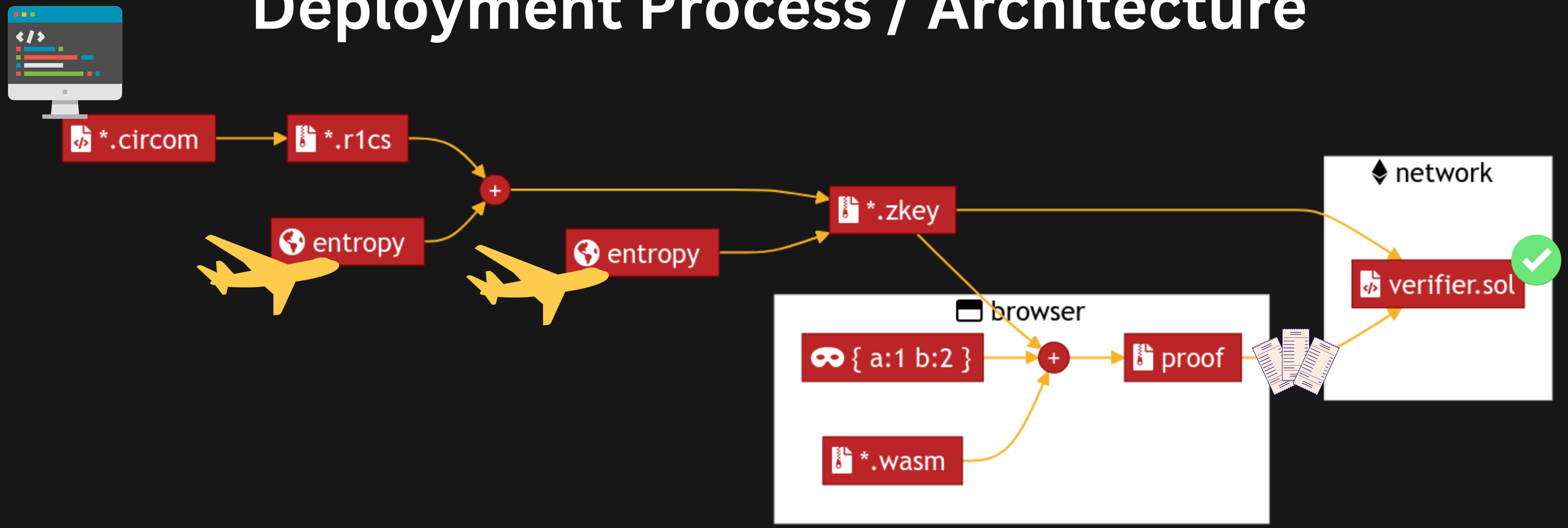
New TS per function ☹

PLONK

Can use a universal setup

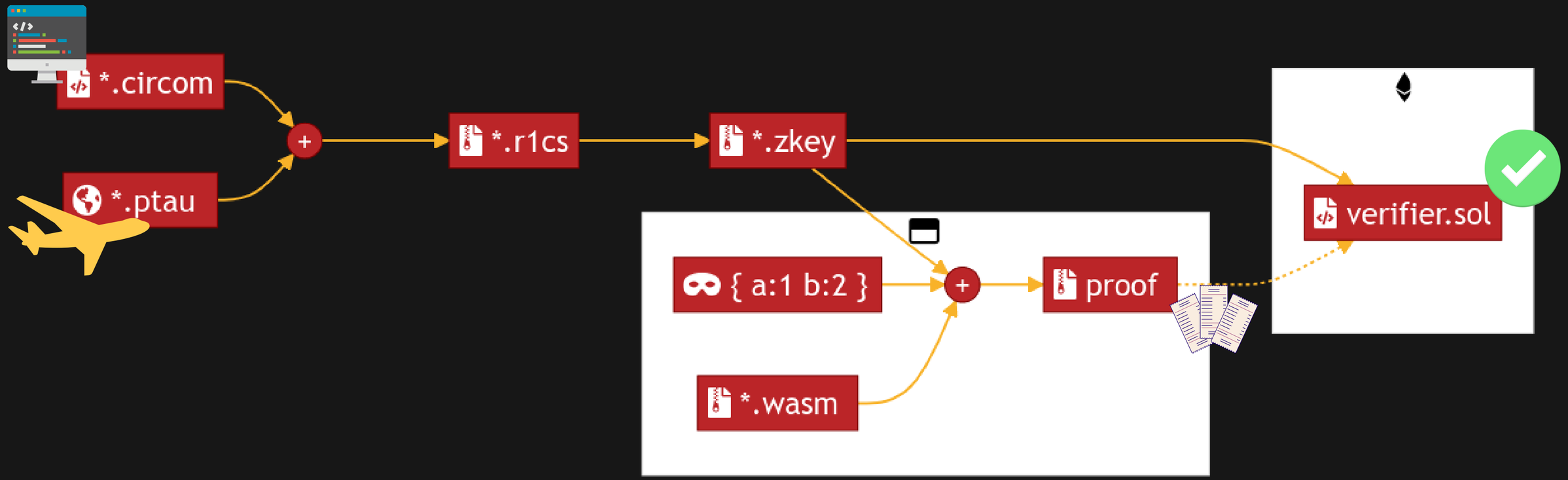
Groth16

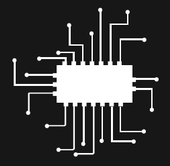
Deployment Process / Architecture



PLONK

Deployment Process / Architecture





iden3

Circom

PLONK

Groth16

Noir

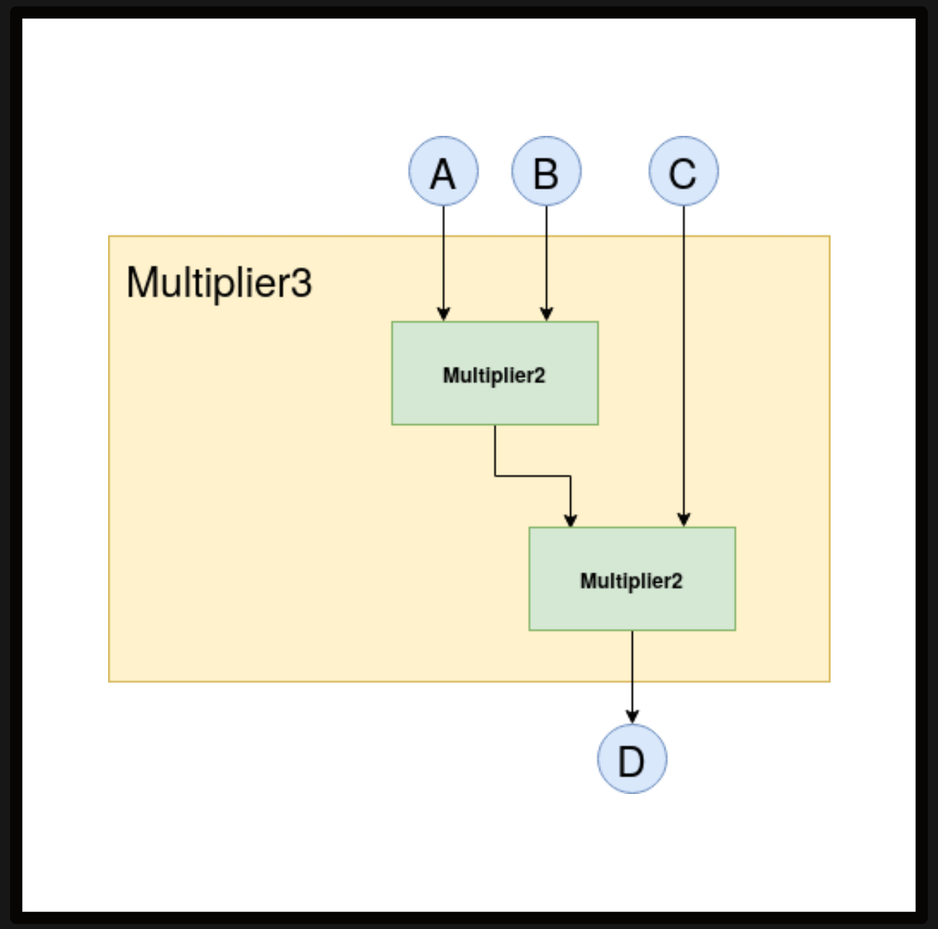
TurboPlonk

</>

Circom

```
pragma circom 2.0.0;

template Multiplier2() {

    // Declaration of signals.
    signal private input a;
    signal private input b;
    signal output c;

    // Constraints.
    c <== a * b;
}
```

Combine Circuits



Circom represents "Algebraic Circuits"

```
pragma circom 2.0.0;

template Multiplier2() {

    // Declaration of signals.
    signal private input a;
    signal private input b;
    signal output c;

    // Constraints.
    c <== a * b;
}
```

```
//This circuit multiplies in1, in2, and in3.
template Multiplier3 () {
    //Declaration of signals and components.
    signal input in1;
    signal input in2;
    signal input in3;
    signal output out;
    component mult1 = Multiplier2();
    component mult2 = Multiplier2();

    //Statements.
    mult1.in1 <== in1;
    mult1.in2 <== in2;
    mult2.in1 <== mult1.out;
    mult2.in2 <== in3;
    out <== mult2.out;
}

component main = Multiplier3();
```

Algebraic Circuits

~~Algebraic Circuits~~

Functions

Multiplier With Functions

```
fn mult(a: Field, b: Field) -> pub Field {
    a * b
}

fn main(a: Field, b: Field, c: Field) -> pub Field {
    mult(mult(a, b), c)
}
```

```
pragma circom 2.0.0;

template Multiplier2() {

    // Declaration of signals.
    signal private input a;
    signal private input b;
    signal output c;

    // Constraints.
    c <== a * b;
}
```

```
//This circuit multiplies in1, in2, and in3.
template Multiplier3 () {
    //Declaration of signals and components.
    signal input in1;
    signal input in2;
    signal input in3;
    signal output out;
    component mult1 = Multiplier2();
    component mult2 = Multiplier2();

    //Statements.
    mult1.in1 <== in1;
    mult1.in2 <== in2;
    mult2.in1 <== mult1.out;
    mult2.in2 <== in3;
    out <== mult2.out;
}

component main = Multiplier3();
```

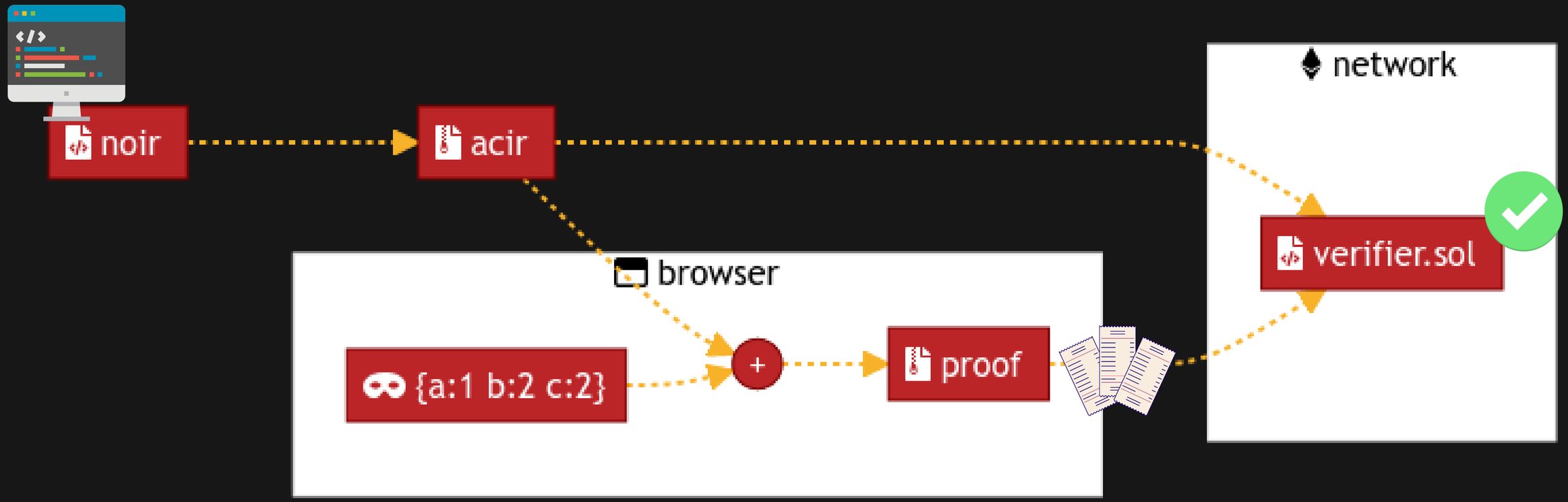
Multiplier With Functions

```
fn mult(a: Field, b: Field) -> pub Field {
    a * b
}

fn main(a: Field, b: Field, c: Field) -> pub Field {
    mult(mult(a, b), c)
}
```

Noir

```
fn mult(a: Field, b: Field) -> pub Field {
    a * b
}

fn main(a: Field, b: Field, c: Field) -> pub Field {
    mult(mult(a, b), c)
}
```

Noir Deployment

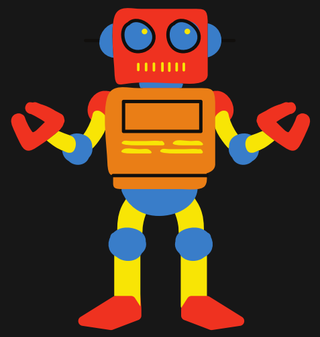


```
use dep::std;

global CANDIDATE_COUNT = 10;

fn check_within_budget(token_budget: u32, votes: [u32; CANDIDATE_COUNT]) {
    let mut tokens_spent: u32 = 0;
    for i in 0..CANDIDATE_COUNT {
        let vote = votes[i];
        tokens_spent = tokens_spent + (vote * vote);
    };
    constrain tokens_spent <= token_budget;
}
```

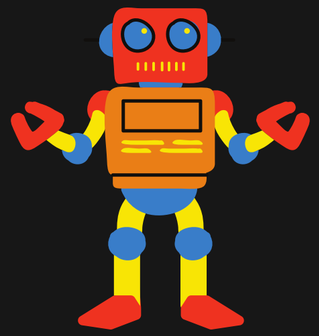
```
fn calculate_ballot_commitment(
    secret: Field,
    votes: [u32; CANDIDATE_COUNT]
) -> Field {
    let mut transcript = [0; CANDIDATE_COUNT + 1];
    transcript[0] = secret;
    for i in 0..CANDIDATE_COUNT {
        transcript[i + 1] = votes[i] as Field;
    };
    std::hash::pedersen(transcript)[0]
}
```

```
fn main(
    token_budget: pub u32,
    votes: [u32; CANDIDATE_COUNT],
    secret: Field
) -> pub Field {
    check_within_budget(token_budget, votes);
    calculate_ballot_commitment(secret, votes)
}
```

SO I TRIED TO BUILD SOMETHING...



WARNING



This is all pretty alpha

Mismatched deployments

Integration with FE is still poor

Be prepared to patch packages and polyfill

- Fullstack application
- Various simple examples
- Compare CIRCOM with Noir
- Monorepo
- Patched Noir WASM where appropriate
- Pollyfilled Node
- Run in web workers

DEMO

TAKE AWAYS

ZK will be transformational

ZKPs are hard!

Functions with Receipts

Noir is 

Simpler Deployment

Lots of this is Alpha

THANK YOU

Twitter https://twitter.com/rudiyardley
Linkedin https://linkedin.com/in/rudiyardley
ZK Sampler https://github.com/blockhackers/zksampler

Slides https://www.canva.com/design/DAFZZEi8nN4/hWLlNhm9usZRLi4sLi_R9g/view
Noir Docs https://noir-lang.github.io/book
Noir Repo https://github.com/noir-lang/noir
Noir Resources https://github.com/noir-lang/awesome-noir
Great Video on Circom https://youtu.be/Oaub9QwwgCA
Powers of Tau 41 https://youtu.be/I4cDAqeEmpU