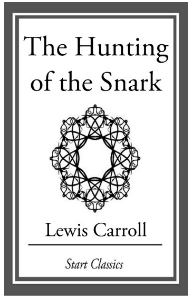
What is a SNARK?

(no, it is not an imaginary animal)

Dan Boneh
Stanford University



What is a SNARK ? (intuition)

SNARK: a <u>succinct</u> proof that a certain statement is true

Example statement: "I know an m such that SHA256$(m) = 0$"

- **SNARK**: the proof is **"short"** and **"fast"** to verify

  [if m is 1GB then the trivial proof (the message m) is neither]

- **zk-SNARK**: the proof "reveals nothing" about m

zk-SNARK: many blockchain applications

Private Tx on a public blockchain:

- Tornado cash, Zcash, IronFish
- Private Dapps: Aleo

Compliance:

- Private proofs of solvency and compliance
- Zero-knowledge taxes

Scalability: Rollup systems with validity proofs

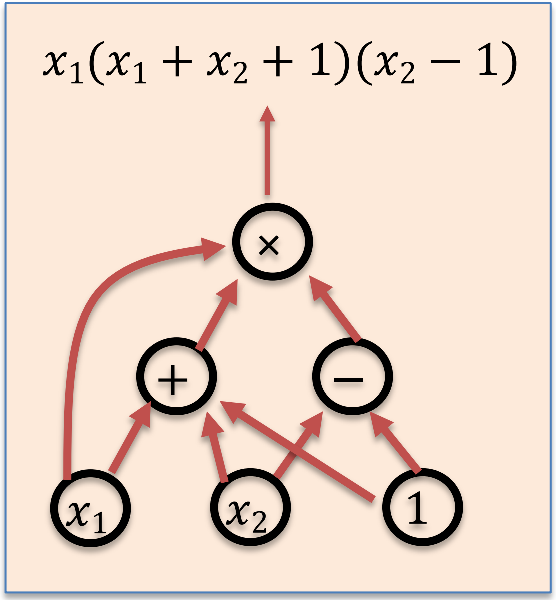
Cryptographic Background

(1) arithmetic circuits

- Fix a finite field $\mathbb{F} := \{0, \ldots, p-1\}$ for some prime p>2.

- **Arithmetic circuit:** $C\colon \mathbb{F}^n \to \mathbb{F}$
  - directed acyclic graph (DAG) where
    internal nodes are labeled +, −, or ×
    inputs are labeled $1, x_1, \ldots, x_n$
  - defines an n-variate polynomial with an evaluation recipe

$|C| = \#$ gates in C



Interesting arithmetic circuits

Examples:

- $C_{hash}(h, \mathbf{m})$: outputs 0 if $SHA256(\mathbf{m}) = h$, and $\neq 0$ otherwise

  $C_{hash}(h, \mathbf{m}) := (h - SHA256(\mathbf{m}))$, $|C_{hash}| \approx$ 20K gates

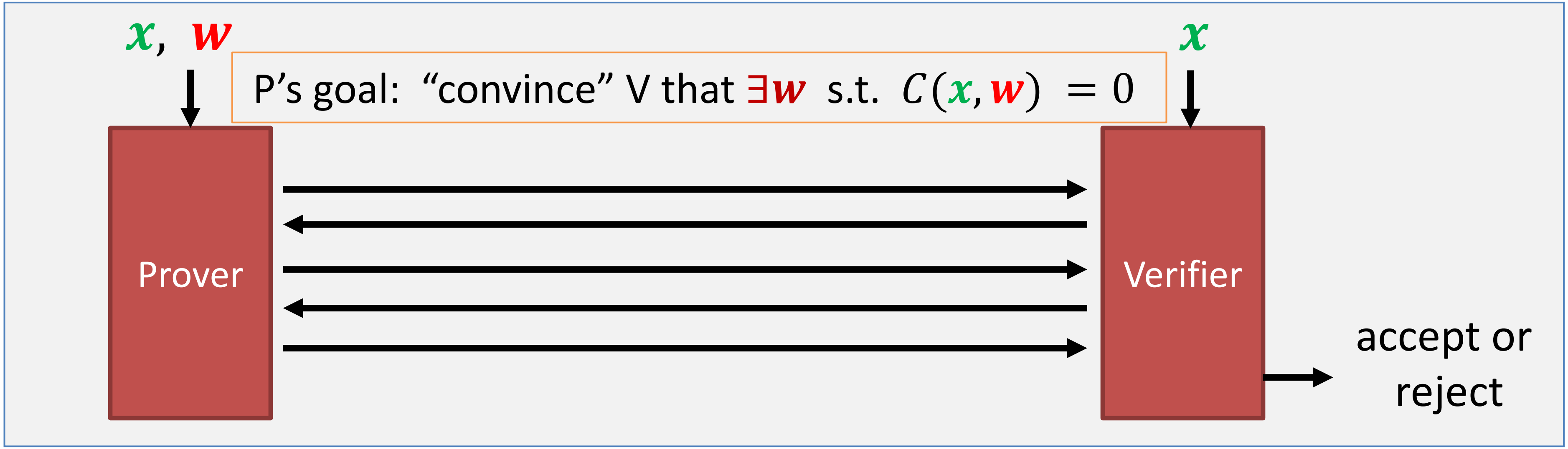
- $C_{sig}(pk, m, \sigma)$: outputs 0 if σ is a valid ECDSA signature on m with respect to pk

(2) Argument systems (for NP)

Public arithmetic circuit: $C(\,x,\ w\,) \rightarrow \mathbb{F}$

public statement in $\mathbb{F}^n$ → x

secret witness in $\mathbb{F}^m$ → w

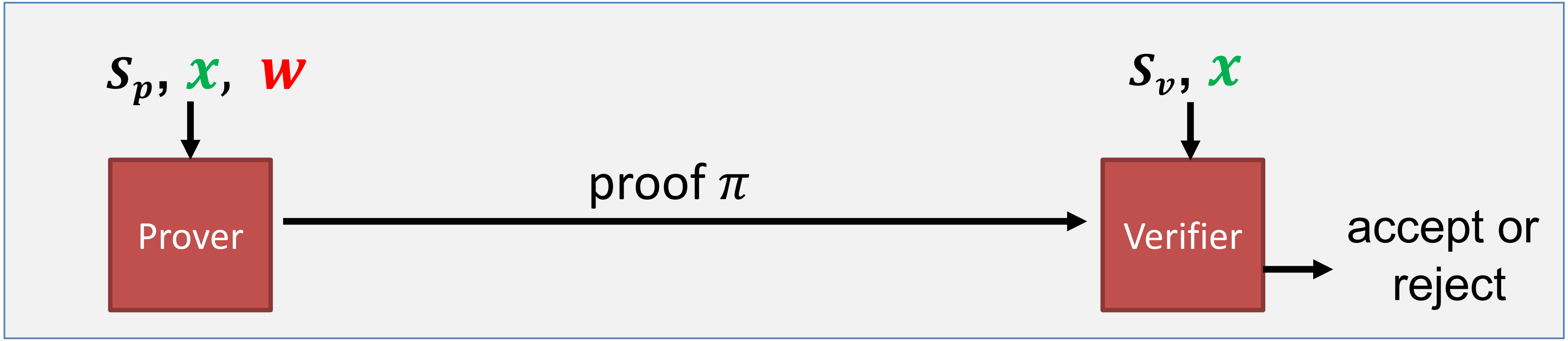


(non-interactive) Preprocessing argument systems

Public arithmetic circuit: $C(x, w) \rightarrow \mathbb{F}$

public statement in $\mathbb{F}^n$ secret witness in $\mathbb{F}^m$

Preprocessing (setup): $\mathbf{S}(C) \rightarrow$ public parameters (S_p, S_v)



Preprocessing argument System

A **preprocessing argument system** is a triple (S, P, V):

- **S**(C) → public parameters (S_p, S_v) for prover and verifier
- **P**$(S_p, \boldsymbol{x}, \boldsymbol{w})$ → proof π
- **V**$(S_v, \boldsymbol{x}, \boldsymbol{\pi})$ → accept or reject

Argument system: requirements (informal)

Prover P(S_p, $\boldsymbol{x}$, $\boldsymbol{w}$) Verifier V (S_v, $\boldsymbol{x}$, $\boldsymbol{\pi}$)

$\xrightarrow{\text{proof } \pi}$

accept or reject

Complete: $\forall x, w$: $C(\boldsymbol{x}, \boldsymbol{w}) = 0 \Rightarrow$ Pr[V(S_v, x, P(S_p, $\boldsymbol{x}$, $\boldsymbol{w}$)) = accept] = 1

Knowledge sound: V accepts $\Rightarrow$ P "knows" $\boldsymbol{w}$ s.t. $C(\boldsymbol{x}, \boldsymbol{w}) = 0$

P* does not "know" $\boldsymbol{w}$ $\Rightarrow$ Pr[V(S_v, x, π) = accept] < negligible

Optional: **Zero knowledge**: $(\mathrm{C}, \mathrm{S_p}, S_v, \boldsymbol{x}, \pi)$ "reveal nothing" about $\boldsymbol{w}$

SNARK: a Succinct ARgument of Knowledge

A **succinct preprocessing argument system** is a triple (S, P, V):

- **S**(C) → public parameters (S_p, S_v) for prover and verifier
- **P**$(S_p, \boldsymbol{x}, \boldsymbol{w})$ → **short** proof π ; $|\pi| = O(\, \mathbf{log}(|\boldsymbol{C}|)\,,\ \lambda)$
- **V**$(S_v, \boldsymbol{x}, \boldsymbol{\pi})$ **fast to verify** ; time(V) = $O(|x|,\ \mathbf{log}(|\boldsymbol{C}|)\,,\ \lambda)$

short "summary" of circuit

Why preprocess C??

SNARK: a Succinct ARgument of Knowledge

A **succinct preprocessing argument system** is a triple (S, P, V):

- **S**(C) → public parameters (S_p, S_v) for prover and verifier
- **P**(S_p, $\boldsymbol{x}$, $\boldsymbol{w}$) → **short** proof π ; $|\pi| = O(\log(|C|), \lambda)$
- **V**(S_v, $\boldsymbol{x}$, $\boldsymbol{\pi}$) **fast to verify** ; time(V) = $O(|x|, \log(|C|), \lambda)$

SNARK: (S, P, V) is **complete**, **knowledge sound**, and **succinct**

zk-SNARK: (S, P, V) is a SNARK and is **zero knowledge**

The trivial argument system

(a) Prover sends $\boldsymbol{w}$ to verifier,

(b) Verifier checks if $C(\boldsymbol{x}, \boldsymbol{w}) = 0$ and accepts if so.

Problems with this:

(1) $\boldsymbol{w}$ might be secret: prover does not want to reveal $\boldsymbol{w}$ to verifier

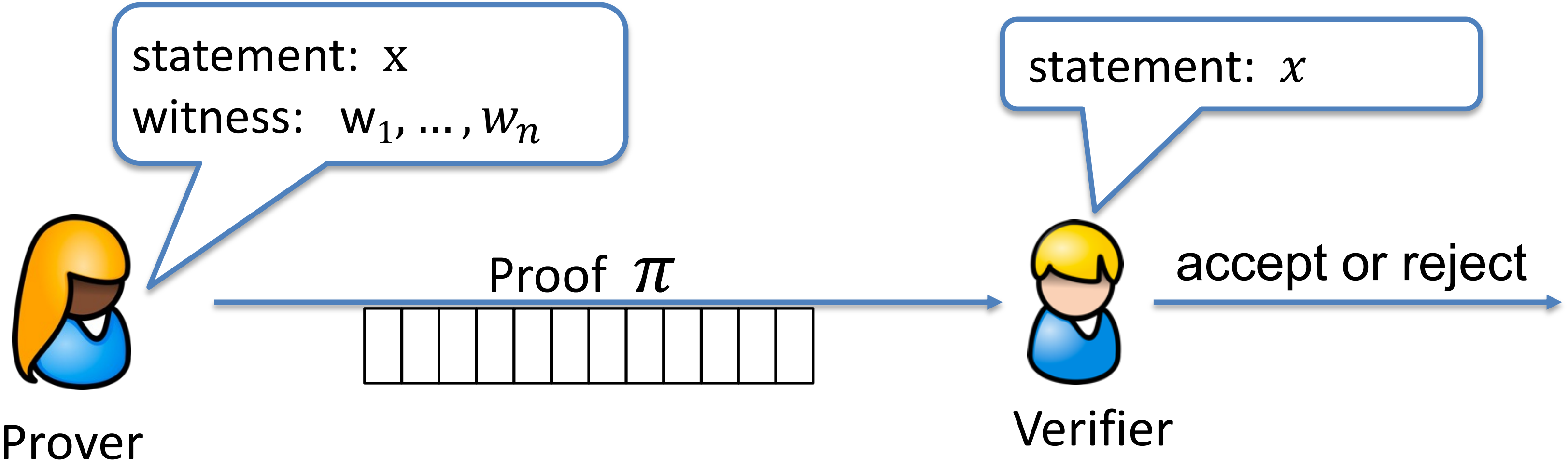
(2) $\boldsymbol{w}$ might be long: we want a "short" proof

(3) computing $C(\boldsymbol{x}, \boldsymbol{w})$ may be hard: we want a "fast" verifier

Back to our first example ...

Prover: I know $(w_1, \ldots, w_n)$ such that $H(w_1, \ldots, w_n) = x$

SNARK: $\text{size}(\pi)$ and $\text{VerifyTime}(\pi)$ is $O(\log n)$!!



Types of preprocessing Setup

Recall setup for circuit C: **S**$(C;r) \rightarrow$ public parameters (S_p, S_v)

random bits

Types of setup:

better

trusted setup per circuit: **S**$(C;r)$ random r must be kept secret from prover

prover learns $r \Rightarrow$ can prove false statements

trusted but universal (updatable) setup: secret r is independent of C

$\boldsymbol{S} = (S_{init}, S_{index})$: $\underbrace{S_{init}(\lambda;r) \rightarrow pp,}_{\text{one-time}}$ $\underbrace{S_{index}(pp, C) \rightarrow (S_p, S_v)}_{\text{no secret data from prover}}$

transparent setup: **S**(C) does not use secret data (no trusted setup)

Significant progress in recent years (partial list)

	size of proof π	size of S_p (beyond C)	verifier time (for a common task)	trusted setup?
Groth'16	≈ 200 Bytes $O_\lambda(1)$	$O_\lambda(\lvert C \rvert)$	≈ 3 ms $O_\lambda(1)$	yes/ per circuit
Plonk/Marlin	≈ 400 Bytes $O_\lambda(1)$	$O_\lambda(\lvert C \rvert)$	≈ 6 ms $O_\lambda(1)$	yes/**universal**

Significant progress in recent years (partial list)

	size of proof π	size of S_p (beyond C)	verifier time (for a common task)	trusted setup?
Groth'16	≈ 200 Bytes $O_\lambda(1)$	$O_\lambda(\lvert C\rvert)$	$\approx$ 3 ms $O_\lambda(1)$	yes/ per circuit
Plonk/Marlin	≈ 400 Bytes $O_\lambda(1)$	$O_\lambda(\lvert C\rvert)$	$\approx$ 6 ms $O_\lambda(1)$	yes/**universal**
Bulletproofs	$\approx$ 1.5 KB $O_\lambda(\log \lvert C\rvert)$	$O_\lambda(1)$	$\approx$ 1.5 sec $O_\lambda(\lvert C\rvert)$	no
STARK	$\approx$ 80 KB $O_\lambda(\log^2 \lvert C\rvert)$	$O_\lambda(1)$	$\approx$ 10 ms $O_\lambda(\log \lvert C\rvert)$	no
DARK	$\approx$ 10 KB $O_\lambda(\log \lvert C\rvert)$	$O_\lambda(1)$	$O_\lambda(\log \lvert C\rvert)$	no
⋮	⋮			⋮

Significant progress in recent years (partial list)

	size of proof π	size of S_p (beyond C)	verifier time (for a common task)	trusted setup?
Groth'16	$\approx$ 200 B		$\approx$ 3 ms	yes/
Plonk/Marlin				
Bulletproofs				
STARK				
DARK	$\approx$ 10 KB $O_\lambda(\log\lvert C\rvert)$	$O_\lambda(1)$	$O_\lambda(\log\lvert C\rvert)$	no
⋮	⋮			⋮

Prover time is almost linear in $|C|$

A SNARK software system

DSL <u>program</u>

Circom,
ZoKrates,
Leo,
Zinc,
Cairo,
Noir,
…

compiler

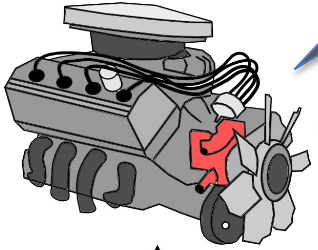
SNARK friendly <u>format</u>

R1CS,
AIR,
Plonk-CG,
PIL,
…

S_p, S_v

SNARK backend prover

heavy computation



π

x, witness

More can be built using the CirC framework

How to define “knowledge soundness” and “zero knowledge”?

Definitions: (1) knowledge sound

Goal: if V accepts then P "knows" $\boldsymbol{w}$ s.t. $C(\boldsymbol{x}, \boldsymbol{w}) = 0$

What does it mean to "know" $\boldsymbol{w}$??

informal def: P knows $\boldsymbol{w}$, if $\boldsymbol{w}$ can be "extracted" from P

P

Definitions: (1) knowledge sound

Formally: (S, P, V) is **knowledge sound** for a circuit C if

for every poly. time adversary $\mathrm{A} = (\mathrm{A}_0, \mathrm{A}_1)$ such that

$\mathrm{S}(C) \to (\mathrm{S_p}, \mathrm{S_v})$, $(x, \text{state}) \leftarrow \mathrm{A}_0(\mathrm{S_p})$, $\pi \leftarrow \mathrm{A}_1(\mathrm{S_p}, x, \text{state})$:

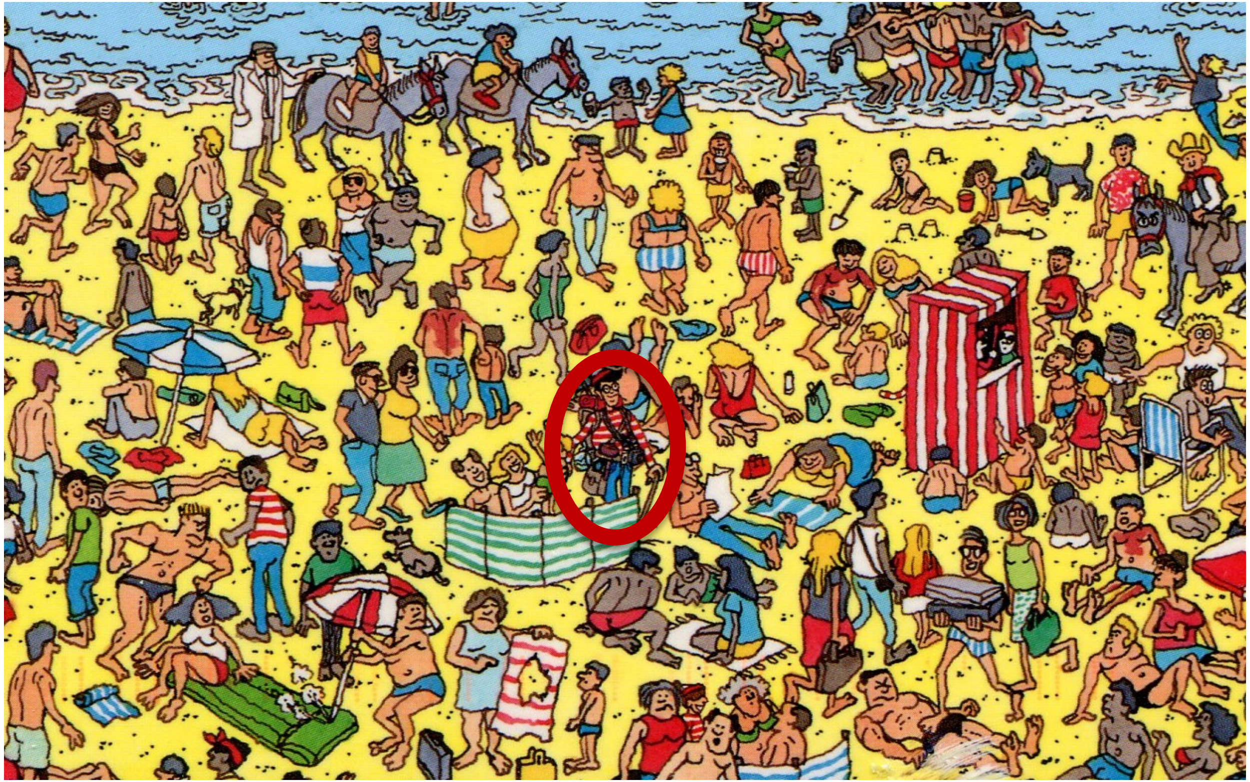
$\Pr[\mathrm{V}(\mathrm{S_v}, x, \pi) = \text{accept}] > 1/10^6$ (non-negligible)

there is an efficient **extractor** E (that uses A_1 as a black box) s.t.

$\mathrm{S}(C) \to (\mathrm{S_p}, \mathrm{S_v})$, $(x, \text{state}) \leftarrow \mathrm{A}_0(\mathrm{S_p})$, $\boxed{w \leftarrow E^{\mathrm{A}_1(\mathrm{S_p}, x, \text{state})}(\mathrm{S_p}, x):}$

$\Pr[\mathrm{C}(x, w) = 0] > 1/10^6 - \epsilon$ (for a negligible ϵ)

Definitions: (2) Zero knowledge



Where is Waldo?

Definitions: (2) Zero knowledge (simplified)

(S, P, V) is **zero knowledge** if for every $x \in \mathbb{F}^n$

proof π "reveals nothing" about $\boldsymbol{w}$, other than its existence

What does it mean to "reveal nothing" ??

Informal def: π "reveals nothing" about $\boldsymbol{w}$ if the verifier can generate π **by itself** $\Longrightarrow$ it learned nothing new from π

(S, P, V) is **zero knowledge** if there is an efficient alg. **Sim** s.t. (S_p, S_v, π) ⇽ ***Sim***(C, x) "look like" the real S_p, S_v and π.

Main point: ***Sim***$(C$,x) simulates π without knowledge of $\boldsymbol{w}$

Definitions: (2) Zero knowledge (simplified)

Formally: (S, P, V) is (honest verifier) **zero knowledge** for a circuit C

if there is an efficient simulator ***Sim*** such that

for all $x \in \mathbb{F}^n$ s.t. $\exists w: C(x, w) = 0$ the distribution:

$(C, \mathrm{S_p}, \mathrm{S_v}, x, \pi)$: where $(\mathrm{S_p}, \mathrm{S_v}) \leftarrow \mathrm{S}(C)$, $\pi \leftarrow \mathrm{P}(\mathrm{S_p}, x, \boldsymbol{w})$

is indistinguishable from the distribution:

$(C, \mathrm{S_p}, \mathrm{S_v}, x, \pi)$: where $(\mathrm{S_p}, \mathrm{S_v}, \pi) \leftarrow \textbf{\textit{Sim}}(C, x)$

Quick review

A zk-SNARK for a circuit C:

- For a public statement x, prover outputs a proof that "convinces" verifier that prover knows w s.t. $C(x, w) = 0$.
- Proof is <u>short</u> and <u>fast</u> to verify

What is it good for?

- Private payments and private Dapp logic (e.g., Aleo)
- Private compliance and L2 scalability

More to think about: private DAO? private governance?

How to build a zk-SNARK?

Next segment

END OF MODULE