CS251 Fall 2021

(cs251.stanford.edu)

Building a SNARK

Dan Boneh

Recap: zk-SNARK applications

Private Tx on a public blockchain:

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

Compliance:

- Proving solvency in zero-knowledge
- Zero-knowledge taxes

Scalability: privacy in zk-SNARK Rollup (next week)

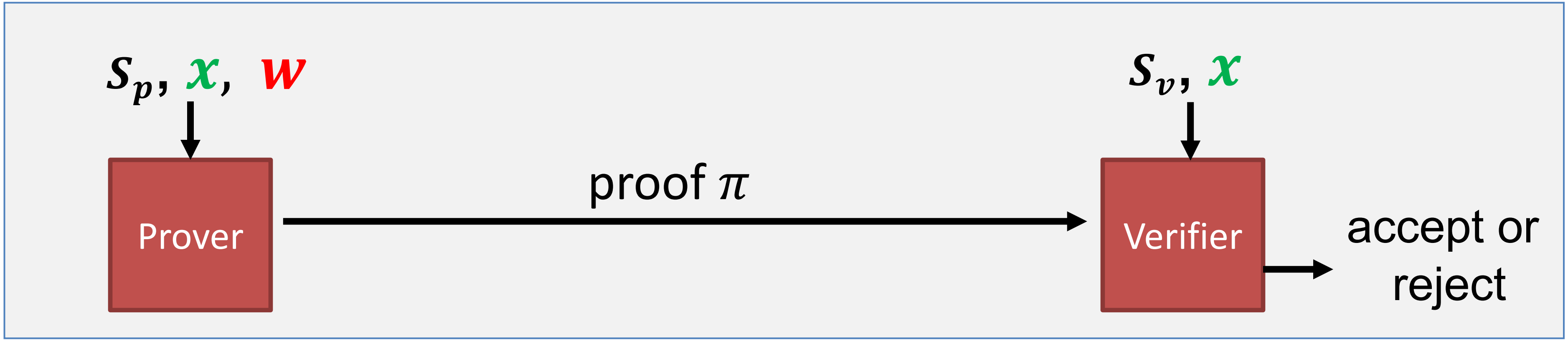
(non-interactive) Preprocessing argument systems

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

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

secret witness in $\mathbb{F}^m$

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



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

Requirements (informal)

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

accept or reject

Complete: $\forall x, w$: $C(\boldsymbol{x}, \boldsymbol{w}) = 0 \quad \Rightarrow \quad$ 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. $C(\boldsymbol{x}, \boldsymbol{w}) = 0$

example: P "knows" $\boldsymbol{w}$ s.t. $[\ H(\boldsymbol{w}) = \boldsymbol{x}$ and $0 \leq \boldsymbol{w} \leq 2^{128}\ ]$

Optional: **Zero knowledge**: $(S_v, \boldsymbol{x}, \pi)$ "reveals 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 ; $\boxed{|\pi| = O(\, \mathbf{log}(|\boldsymbol{C}|)\,,\ \lambda)}$
- **V**$(S_v, \boldsymbol{x}, \boldsymbol{\pi})$ **fast to verify** ; $\boxed{\text{time(V)} = O(|x|,\ \mathbf{log}(|\boldsymbol{C}|)\,,\ \lambda)}$

short "summary" of circuit

λ = security parameter = 128

A simple PCP-based SNARK

[Kilian’92, Micali’94]

A simple construction: PCP-based SNARK

<u>The PCP theorem</u>: Let $C(x, w)$ be an arithmetic circuit.

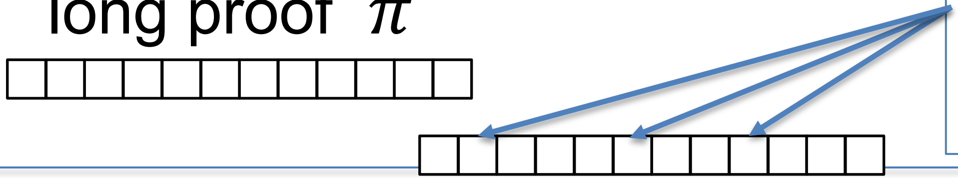
there is a proof system that for every x proves $\exists w: C(x, w) = 0$ as follows:

<u>Prover P($\boldsymbol{S_p}$, $\boldsymbol{x}$, $\boldsymbol{w}$)</u>

long proof π

<u>Verifier V($\boldsymbol{S_v}$, $\boldsymbol{x}$)</u>

read only $O(\lambda)$ bits of π, output accept or reject



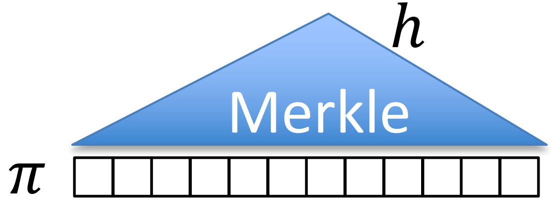
V always accepts valid proof. If no w, then V rejects with high prob.

size of proof π is $poly(|C|)$. (not succinct)

Converting a PCP proof to a SNARK

Prover P(S_p, x, w)

Verifier V(S_v, x)



Merkle root h

1 hash

open k positions of π $(k = O(\lambda))$

$O(k \log |C|)$ hashes

k opening and Merkle proofs

output accept or reject

Verifier sees $O(\lambda \log |C|)$ data $\Rightarrow$ succinct proof.

Problem: **interactive**

Making the proof non-interactive

The **Fiat-Shamir transform**:

- public-coin interactive protocol $\Rightarrow$ non-interactive protocol

public coin: all verifier randomness is public (no secrets)

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

msg1 $\longrightarrow$

$\longleftarrow$ r choose random bits r

msg2 $\longrightarrow$

accept or reject

Making the proof non-interactive

Fiat-Shamir transform: $H: M \to R$ a cryptographic hash function

- idea: prover generates random bits on its own (!)

Prover P($\boldsymbol{S_p}$, $\boldsymbol{x}$, $\boldsymbol{w}$)

Verifier V($\boldsymbol{S_v}$, $\boldsymbol{x}$)

generate msg1

r ⟵ H($\boldsymbol{x}$, msg1)

generate msg2

π = (msg1, msg2) ⟶

$|\pi| = O(\lambda \log |C|)$

r ⟵ H($\boldsymbol{x}$, msg1)

accept or reject

Fiat-Shamir: certain secure interactive protocols $\Longrightarrow$ non-interactive

Why is this an argument of knowledge? (can skip)

Let's build an extractor E for the interactive protocol:

- After prover commits to Merkle root of proof

  E asks prover to open many batches of $k = O(\lambda)$ positions of π
  (by rewinding prover)

- E fails to extract cell #j of π if

  (1) prover produces a false Merkle proofs (efficient prover cannot), or

  (2) prover fails (i.e., verifier rejects) whenever j is in batch to open:

  $$\Pr[\text{prover fails}] \geq \Pr[\text{ j in batch }] = 1 - (1 - 1/|\pi|)^k .$$

  so: this cannot happen if k is sufficiently large

$\Rightarrow$ E extracts entire proof π. Once π is known, E can obtain $\boldsymbol{w}$ from π.

Are we done?

Simple transparent SNARK from the PCP theorem

- Use Fiat-Shamir transform to make non-interactive
- We will apply Fiat-Shamir in many other settings

The bad news: an impractical SNARK --- Prover time too high

Better SNARKs: Goal: Time(Prover) = $\tilde{O}(|C|)$

Building an efficient SNARK

General paradigm

Many SNARKs are built in two steps:

polynomial interactive oracle proofs (poly-IOP)

polynomial commitment scheme

(zk)SNARK for general circuits

Recall: commitments

Two algorithms:

- *commit*(m, r) → ***com*** (r chose at random)
- *verify*$(m,$ ***com***$, r)$ → accept or reject

Properties:

- binding: cannot produce two valid openings for ***com***.
- hiding: ***com*** reveals nothing about committed data

(1) Polynomial commitment schemes

Notation:

Fix a finite field: $\mathbb{F}_p = \{0,1, \ldots, p-1\}$

$\mathbb{F}_p^{(\leq d)}[X]$: all polynomials in $\mathbb{F}_p[\mathrm{X}]$ of degree ≤ d.

(1) Polynomial commitment schemes

- *setup*$(d) \rightarrow pp$,   public parameters for polynomials of degree $\leq d$
- *commit*$(pp, \text{f}, r) \rightarrow \boldsymbol{com_f}$ commitment to $\text{f} \in \mathbb{F}_p^{(\leq d)}[X]$
- *eval*: goal: for a given $\boldsymbol{com_f}$ and $\text{x}, \text{y} \in \mathbb{F}_p$, prove that $\text{f(x)} = \text{y}$.

Formally: *eval* = (P, V) is a SNARK for:

statement $st = (pp, \boldsymbol{com_f}, \text{x}, \text{y})$ with witness = $w = (\text{f}, r)$

where $C(st, w) = 0$ iff

$$\left[\ \text{f(x)} = \text{y} \quad \textit{and} \quad \text{f} \in \mathbb{F}_p^{(\leq d)}[X] \quad \textit{and} \quad \textit{commit}(pp, \text{f}, r) = \boldsymbol{com}_{\text{f}}\ \right]$$

(1) Polynomial commitment schemes

Properties:

- Binding: cannot produce two valid openings (f_1, r_1), (f_2, r_2) for $\boldsymbol{com_f}$.
- eval is knowledge sounds (can extract (f, r) from a successful prover)
- optional:
  - commitment is hiding
  - eval is zero knowledge

Constructing polynomial commitments

Not today …   (see readings or CS355)

simple construction without this requirement

Properties of the best ones:

- transparent setup: no secret randomness in setup
- ***com***$_f$ is constant size (a single group element)
- eval proof size for f ∈ $\mathbb{F}_p^{(\leq d)}[X]$ is $\boxed{\text{O}(\log d)}$ group elements
- eval verify time is $\boxed{\text{O}(\log d)}$ Prover time: $\boxed{\text{O}(d)}$

Component 2: Polynomial IOP

Goal: polynomial commitment scheme $\Rightarrow$
SNARK for a general circuit $C(x, w)$.

… done using a polynomial-IOP

Fix an arithmetic circuit $C(x, w)$. Let $x \in \mathbb{F}_p^n$.

<u>Poly-IOP</u>: a proof system that proves $\exists w: C(x, w) = 0$ as follows:

(2) Polynomial IOP

Prover $P(S_p, x, w)$

Verifier $V(S_v, x)$

commit $\boxed{f_1 \in \mathbb{F}_p^{(\leq d)}[X]}$

$r_1 \leftarrow \mathbb{F}_p$

r_1

commit $\boxed{f_2 \in \mathbb{F}_p^{(\leq d)}[X]}$

$r_2 \leftarrow \mathbb{F}_p$

r_2

⋮

r_{t-1}

$r_{t-1} \leftarrow \mathbb{F}_p$

commit $\boxed{f_t \in \mathbb{F}_p^{(\leq d)}[X]}$

fast verify that can evaluate f_i at any x in $\mathbb{F}_p$

$\text{verify}^{f_1, \ldots, f_t}(r_1, \ldots, r_{t-1})$

Properties

- Complete: if $\exists w: C(x,w) = 0$ then verifier always accepts

- Knowledge sound: (informal) Let $x \in \mathbb{F}_p^n$.

  P*: a prover that convinces the verifier with prob. $\geq 1/10^6$

  then there is an efficient extractor E s.t.

  $$\Pr\left[\ E(x, f_1, r_1, \dots, r_{t-1}, f_t) = w \ \text{ s.t. } C(x,w) = 0\ \right] \geq 1/10^6 - \varepsilon$$

- Optional: zero knowledge

The resulting SNARK

Poly-IOP params: t = #polynomials, q = # eval queries in verify

The SNARK:

- During interactive phase of poly-IOP: send t poly commitments
- During poly-IOP verify: run poly-commit eval protocol q times
- Use Fiat-Shamir to make the proof system non-interactive

Length of SNARK proof: t poly-commits + q eval proofs

SNARK verify time: q poly eval proof verifications + time(IOP-verify)

SNARK prover time: t poly commits + time(IOP-prover)

Constructing a Poly-IOP: t + q = 4

First some useful tricks …

The fundamental theorem of algebra: for $0 \neq f \in \mathbb{F}_p^{(\leq d)}[X]$

$$\boxed{\text{for } r \leftarrow \mathbb{F}_p : \qquad \Pr[\, f(r) = 0 \,] \leq d/p}$$

⇒ suppose $p \approx 2^{256}$ and $d \leq 2^{40}$ then d/p is negligible

⇒ for $r \leftarrow \mathbb{F}_p$, if $f(r) = 0$ then f is identically zero w.h.p

⇒ simple zero test for a committed polynomial

Some useful gadgets

Let $\omega \in \mathbb{F}_p$ be a primitive k-th root of unity $(\omega^k = 1)$

Set H := { 1, ω, ω^2, …, $\omega^{k\text{-}1}$ } $\subseteq \mathbb{F}_p$

Let $f \in \mathbb{F}_p^{(\leq d)}[X]$ and $b, c \in \mathbb{F}_p$. $(d \geq k)$

Want poly-IOPs for the following tasks:

Task 1 (**zero-test**): prove that f is identically zero on H

Tast 2 (**sum-check**): prove that $\sum_{a \in H} f(a) = b$

Task 3 (**prod-check**): prove that $\prod_{a \in \mathrm{H}} f(a) = c$

Zero test on H (H = { $1, \omega, \omega^2, \ldots, \omega^{k-1}$ })

Prover P(f, ⊥)

$q(X) \leftarrow f(X)/(X^k - 1)$

if f is zero on H then
$f(X)$ is divisible by $X^k - 1$

$\boxed{q \in \mathbb{F}_p^{(\leq d)}[X]}$ →

eval $q(X)$ and $f(X)$ at r ↔

Verifier V($\boxed{f}$)

$r \leftarrow \mathbb{F}_p$

learn $q(r)$, $f(r)$

accept if $f(r) \stackrel{?}{=} q(r) \cdot (r^k - 1)$

(implies that $f(X) = q(X)(X^k - 1)$)

Thm: this protocol is complete and sound, assuming d/p is negligible.

Verifier time: O(log k) and two eval verify (but can be done in one)

Product check on H: $\prod_{a \in \mathrm{H}} f(a) = 1$

Let $t \in \mathbb{F}_p^{(\leq k)}[X]$ be the degree-d polynomial:

$$t(1) = f(1), \qquad t(\omega^s) = \prod_{i=0}^{s} f(\omega^i) \quad \text{for } s = 1, \ldots, k-1$$

Then $\mathrm{t}(\omega^{k-1}) = \prod_{a \in \mathrm{H}} f(a) = 1$

and $t(\omega \cdot \mathrm{x}) = t(x) \cdot f(\omega \cdot \mathrm{x})$ for all $x \in H$ (including $x = \omega^{k-1}$)

Lemma: if (1) $\mathrm{t}(\omega^{k-1}) = 1$ and

(2) $t(\omega \cdot \mathrm{x}) - t(x) \cdot f(\omega \cdot \mathrm{x}) = 0$ for all $x \in H$

then $\prod_{a \in \mathrm{H}} f(a) = 1$

Product check on H (unoptimized)

Prover P$((f, c), \perp)$ Verifier V($\boxed{f}$)

construct $t(X) \in \mathbb{F}_p^{(\leq k)}$, $t_1(X) = t(\omega \cdot X) - t(X) \cdot f(\omega \cdot X)$

and $q(X) = t_1(X)/(X^k - 1) \in \mathbb{F}_p^{(\leq k)}$

$\boxed{q,\ t \in \mathbb{F}_p^{(\leq k)}[X]}$ $\longrightarrow$

$r \leftarrow \mathbb{F}_p$

$\longleftrightarrow$ eval $t(X)$ at ω^{k-1}, r, ωr

learn $t(\omega^{k-1})$, $t(r)$, $t(\omega r)$, $q(r)$, $f(\omega r)$

$\longleftrightarrow$ eval $q(X)$ at r, and $f(X)$ at ωr

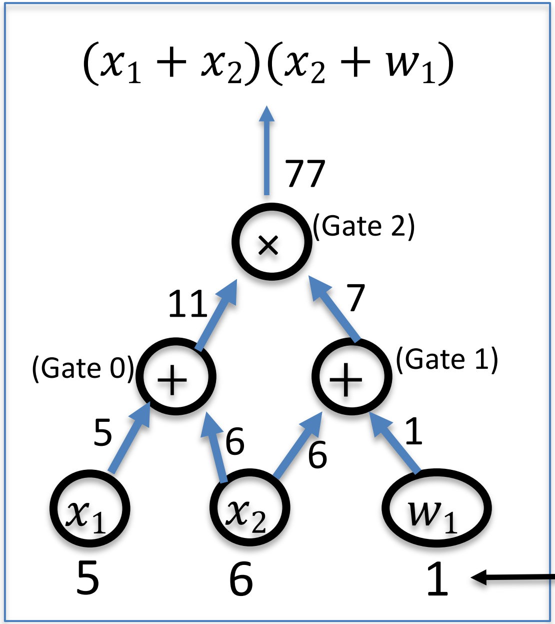
$t_1(H) = 0$:

accept if $t(\omega^{k-1}) \stackrel{?}{=} 1$ and
$t(\omega r) - t(r) f(\omega r) \stackrel{?}{=} q(r) \cdot (r^k - 1)$

PLONK: a poly-IOP for a general circuit

PLONK: a poly-IOP for a general circuit $C(x, w)$

Step 1: compile circuit to a computation trace (gate fan-in = 2)



example input

The computation trace:

inputs:	5,	6,	1
Gate 0:	5 ,	6 ,	11
Gate 1:	6 ,	1 ,	7
Gate 2:	11,	7,	77

left inputs | right inputs | outputs

Encoding the trace as a polynomial

$|C|$ = total # of gates in C , $|I| = |I_x| + |I_w|$ = # inputs to C

let $d = 3\,|C| + |I|$ (in example, $d = 12$) and H = { 1, ω, ω^2, …, ω^{d-1} }

The plan: prover interpolates a polynomial

$$P \in \mathbb{F}_p^{(\leq d)}[X]$$

that encodes the entire trace.

Let's see how …

inputs:	5,	6,	1
Gate 0:	5 ,	6 ,	11
Gate 1:	6 ,	1 ,	7
Gate 2:	11,	7,	$\boxed{77}$

Encoding the trace as a polynomial

The plan: (prover uses FFT to compute coefficients of P in time $d \log_2 d$)

Prover interpolates $P \in \mathbb{F}_p^{(\leq d)}[X]$ such that

(1) $P(\omega^{-j}) =$ input #j for j = 1, …, $|I|$ (all inputs), and

(2) $\forall\, l = 0, \dots, |C| - 1$:

- $P(\omega^{3l})$: left input to gate #l
- $P(\omega^{3l+1})$: right input to gate #l
- $P(\omega^{3l+2})$: output of gate #l

inputs:	5,	6,	1
Gate 0:	5 ,	6 ,	11
Gate 1:	6 ,	1 ,	7
Gate 2:	11,	7,	77

Encoding the trace as a polynomial

In our example:

inputs:	$P(\omega^{-1}) = 5,$	$P(\omega^{-2}) = 6,$	$P(\omega^{-3}) = 1,$
gate 0:	$P(\omega^{0}) = 5,$	$P(\omega^{1}) = 6,$	$P(\omega^{2}) = 11,$
gate 1:	$P(\omega^{3}) = 6,$	$P(\omega^{4}) = 1,$	$P(\omega^{5}) = 7,$
gate 2:	$P(\omega^{6}) = 11,$	$P(\omega^{7}) = 7,$	$P(\omega^{8}) = 77$

inputs:	5,	6,	1
Gate 0:	5 ,	6 ,	11
Gate 1:	6 ,	1 ,	7
Gate 2:	11,	7,	77

degree(P) = 11

Step 2: proving validity of P

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

build P(X) ∈ $\mathbb{F}_p^{(\leq d)}[X]$ $\xrightarrow{\boxed{P}}$ (commitment)

Prover needs to prove that P is a correct computation trace:

(1) P encodes the correct inputs,

(2) every gate is evaluated correctly,

(3) the wiring is implemented correctly,

(4) the final output is 0

inputs:	5,	6,	1
Gate 0:	5 ,	6 ,	11
Gate 1:	6 ,	1 ,	7
Gate 2:	11,	7,	77

Proving (4) is easy: prove P($\omega^{3|C|-1}$) $= 0$

Proving (1): P encodes the correct inputs

Both <u>prover</u> and <u>verifier</u> interpolate a polynomial $v(X) \in \mathbb{F}_p^{(\leq |I_x|)}$[X] that encodes the x-inputs to the circuit:

for $j = 1, \ldots, |I_x|$: $v(\omega^{-j}) =$ input #j

In our example: $v(\omega^{-1}) = 5$, $v(\omega^{-2}) = 6$. (v is linear)

constructing $v(X)$ takes time proportional to the size of input x

Proving (1): P encodes the correct inputs

Both prover and verifier interpolate a polynomial $v(X) \in \mathbb{F}_p^{(\leq |I_x|)}[X]$ that encodes the x-inputs to the circuit:

$$\text{for } j = 1, \ldots, |I_x|: \qquad v(\omega^{-j}) = \text{input \#j}$$

Let $H_{inp} = \{ \omega^{-1}, \omega^{-2}, \ldots, \omega^{-|I_x|} \} \subseteq H$ (points encoding the input)

Prover proves (1) by using a zero-test on H_{inp} to prove that

$$\boxed{P(y) - v(y) = 0 \qquad \forall\, y \in H_{inp}}$$

Proving (2): every gate is evaluated correctly

Idea: encode gate types using a *selector* polynomial S(X)

define S(X) $\in \mathbb{F}_p^{(\leq d)}$[X] such that $\forall\, l = 0, \dots, |C| - 1$:

S(ω^{3l}) = 1 if gate #l is an addition gate

S(ω^{3l}) = 0 if gate #l is a multiplication gate

In our example S$(\omega^0) = 1$, S$(\omega^3) = 1$, S$(\omega^6) = 0$

(so that S is a quadratic polynomial)

Proving (2): every gate is evaluated correctly

Idea: encode gate types using a *selector* polynomial S(X)

define $S(X) \in \mathbb{F}_p^{(\leq d)}[X]$ such that $\forall\, l = 0, \dots, |C| - 1$:

$S(\omega^{3l}) = 1$ if gate $\#l$ is an addition gate

$S(\omega^{3l}) = 0$ if gate $\#l$ is a multiplication gate

Observe that, $\forall\, y \in H_{gates} = \{\, 1, \omega^3, \omega^6, \omega^9, \dots, \omega^{3(|C|-1)} \,\}$:

$$S(y)\cdot[P(y) + P(\omega y)] + (1 - S(y))\cdot P(y)\cdot P(\omega y) = P(\omega^2 y)$$

Proving (2): every gate is evaluated correctly

Setup(C): S_v = (poly commitment to S(X))

<u>Prover P(S_p, $\boldsymbol{x}$, $\mathbf{w}$)</u>

<u>Verifier V(S_v, $\boldsymbol{x}$)</u>

build $P(X) \in \mathbb{F}_p^{(\leq d)}[X]$ $\xrightarrow{\boxed{P}}$ (commitment)

Prover uses zero-test on H_{gates} to prove that $\forall\, y \in H_{gates}$

$$S(y)\cdot[P(y) + P(\omega y)] + \big(1 - S(y)\big)\cdot P(y)\cdot P(\omega y) - P(\omega^2 y) = 0$$

Proving (3): the wiring is correct

Step 4: encode the wires of C:

$$\begin{cases} P(\omega^{-2}) = P(\omega^{1}) = P(\omega^{3}) \\ P(\omega^{-1}) = P(\omega^{0}) \\ P(\omega^{2}) = P(\omega^{6}) \\ P(\omega^{-3}) = P(\omega^{4}) \end{cases}$$

<u>example:</u> $x_1=5, x_2=6, w_1=1$

$\omega^{-1}, \omega^{-2}, \omega^{-3}$:	**5, 6, 1**
$\omega^{0}, \omega^{1}, \omega^{2}$:	**5, 6, 11**
$\omega^{3}, \omega^{4}, \omega^{5}$:	**6, 1, 7**
$\omega^{6}, \omega^{7}, \omega^{8}$:	**11, 7, 77**

Define a polynomial W: H → H that implements a rotation:

$$W(\omega^{-2}, \omega^{1}, \omega^{3}) = (\omega^{1}, \omega^{3}, \omega^{-2}), \quad W(\omega^{-1}, \omega^{0}) = (\omega^{0}, \omega^{-1}), \ \ldots$$

<u>**Lemma**</u>: $\forall\, y \in H$: $P(y) = P(W(y))$ ⇒ wire constraints are satisfied

Proving (3): encoding the circuit wiring

Problem: the constraint $P(y) = P(W(y))$ has degree d^2

$\Rightarrow$ prover would need to manipulate polynomials of degree d^2

$\Rightarrow$ quadratic time prover !! (goal: linear time prover)

Cute trick: use prod-check proof to reduce this to a constraint of linear degree

Reducing wiring check to a linear degree

Lemma: P(y) = P(W(y)) for all y∈H if and only if $L(Y,Z) \equiv 1$,

where $L(Y,Z) = \prod_{x \in H} \frac{P(x)+Y \cdot W(x)+Z}{P(x)+Y \cdot x+Z}$

To prove that $L(Y,Z) \equiv 1$ do:

(1) verifier chooses random $y, z \in \mathbb{F}_p$

(2) prover builds L_1(X) s.t. $L_1(x) = \frac{P(x)+y \cdot W(x)+z}{P(x)+y \cdot x+z}$ for all x∈H

(3) run prod-check to prove $\prod_{x \in H} L_1(x) = 1$

(4) validate L_1: run zero-test on H to prove $L_2(x)$ = 0 for all x∈H, where

$$L_2(\mathrm{x}) = (P(x) + y \cdot x + z)\, L_1(x) \; - \; (P(x) + y \cdot W(x) + z)$$

The final (S, P, V) SNARK

Setup(C): S_v = (poly commitment to S(X) and W(X))

<u>Prover P(S_p, $\boldsymbol{x}$, $\mathbf{w}$)</u>

<u>Verifier V(S_v, $\boldsymbol{x}$)</u>

build $P(X) \in \mathbb{F}_p^{(\leq d)}[X]$ $\xrightarrow[\text{(commitment)}]{\boxed{P}}$ build $v(X) \in \mathbb{F}_p^{(\leq |I_x|)}[X]$

Prover proves:

gates: (1) $S(y)\cdot[P(y) + P(\omega y)] + (1 - S(y))\cdot P(y)\cdot P(\omega y) - P(\omega^2 y) = 0$ $\forall\, y \in H_{gates}$

inputs: (2) $P(y) - v(y) = 0$ $\forall\, y \in H_{inp}$

wires: (3) $P(y) - P(W(y)) = 0$ $\forall\, y \in H$

output: (4) $P(\omega^{3|C|-1}) = 0$ (output of last gate = 0)

Many extensions …

- Plonk proof: a short proof (O(1) commitments), fast verifier

- Can handle circuits with more general gates than + and ×
  - PLOOKUP: efficient SNARK for circuits with lookup tables

- The SNARK can easily be made into a zk-SNARK

Main challenge: reduce prover time

END OF LECTURE

Next lecture: scaling the blockchain