CS251 Fall 2022

(cs251.stanford.edu)

Building a SNARK

Dan Boneh

Recap: zk-SNARK applications

Private Tx on a public blockchain: Zcash, IronFish

Compliance:

- Proving that a private Tx are in compliance with banking laws
- Proving solvency in zero-knowledge

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

Bridging between blockchains: zkBridge

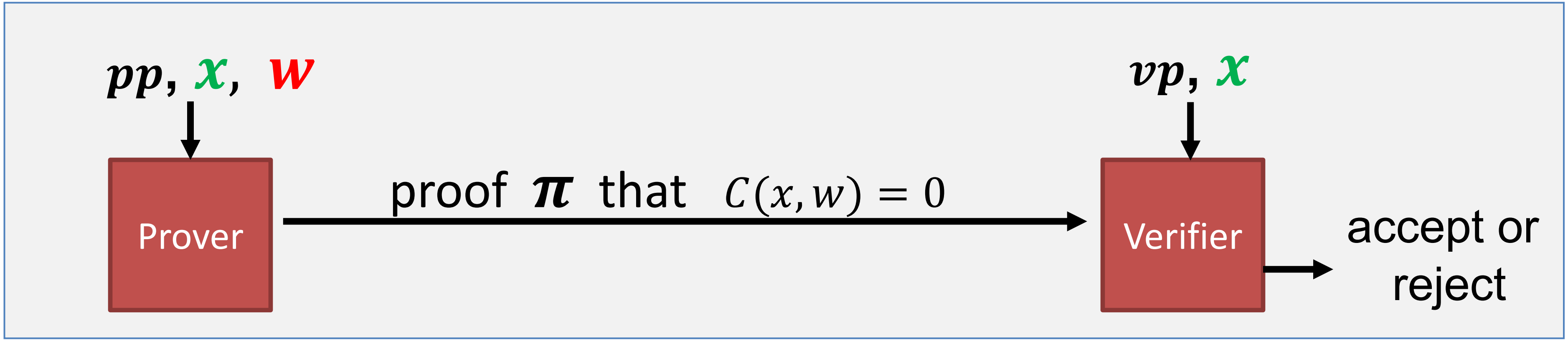
(preprocessing) NARK: Non-interactive ARgument of Knowledge

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

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

secret witness in $\mathbb{F}^m$ → $\boldsymbol{w}$

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



NARK: requirements (informal)

Prover P(pp, $\mathbf{x}$, $\mathbf{w}$)   Verifier V (vp, $\mathbf{x}$, $\boldsymbol{\pi}$)

proof π →

accept or reject

Complete: $\forall x, w$: $C(\mathbf{x}, \mathbf{w}) = 0 \Rightarrow$ Pr[V(vp, x, P(pp, $\mathbf{x}$, $\mathbf{w}$)) = accept] = 1

Adaptively **knowledge sound**: V accepts $\Rightarrow$ P "knows" $\mathbf{w}$ s.t. $C(\mathbf{x}, \mathbf{w}) = 0$

(an extractor E can extract a valid $\mathbf{w}$ from P)

Optional: **Zero knowledge**: $(C, pp, vp, \mathbf{x}, \pi)$ "reveal nothing new" about $\mathbf{w}$

SNARK: a Succinct ARgument of Knowledge

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

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

short "summary" of circuit

A simple PCP-based SNARK

[Kilian’92, Micali’94]

A simple construction: PCP-based SNARK

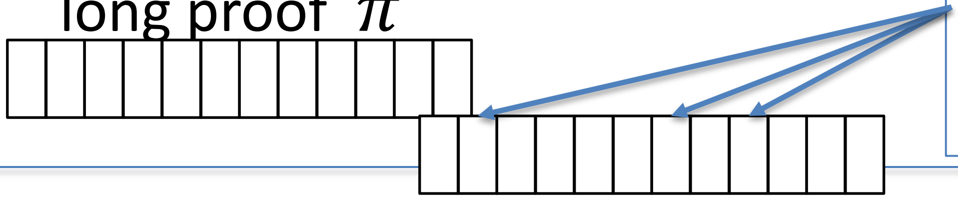
The PCP theorem: Let $C(x, w)$ be an arithmetic circuit.

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

Prover P(pp, x, w)

Verifier V(vp, x)

long proof π



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

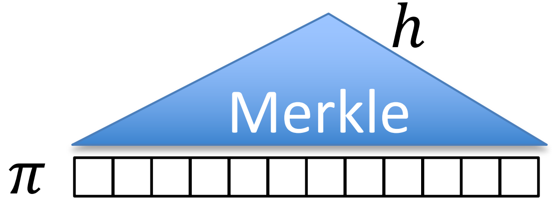
V always accepts valid proof. If no w, then V rejects with high prob.

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

Converting a PCP proof to a SNARK

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

Verifier V(*vp*, $\boldsymbol{x}$)



Merkle root h → 1 hash

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

→ k opening and Merkle proofs — $O(k \log |C|)$ hashes

output accept or reject

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

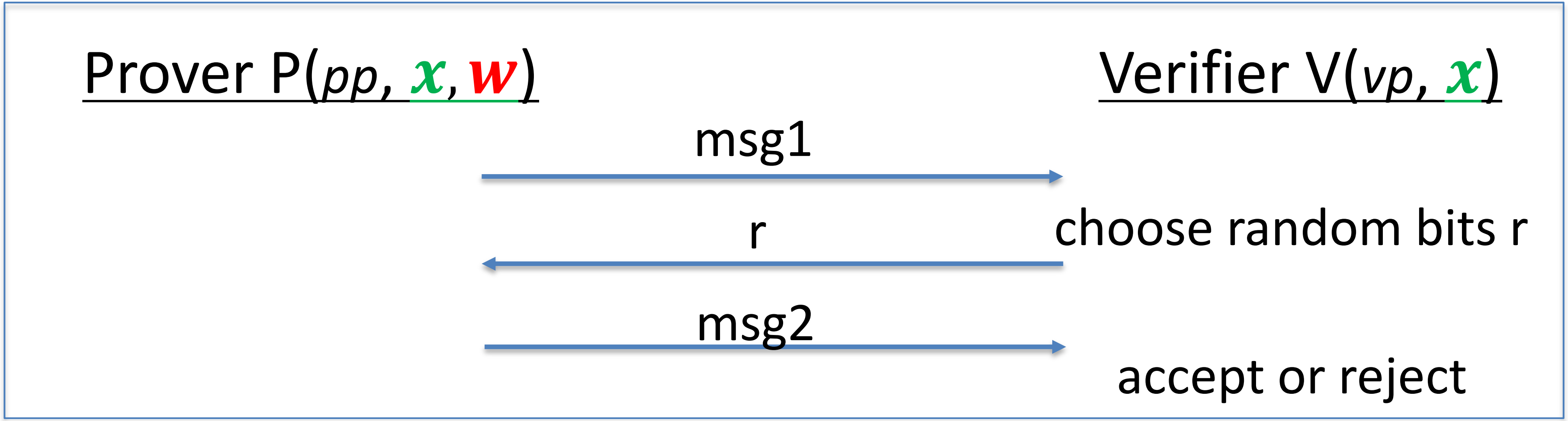
Problem: **interactive**

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)



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(pp, x, w)

generate msg1

r ← H(x, msg1)

generate msg2

π = (msg1, msg2) →

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

Verifier V(vp, x)

r ← H(x, msg1)

accept or reject

Fiat-Shamir: certain secure interactive protocols ⟹ non-interactive

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: two steps

(1) A polynomial commitment scheme

(cryptographic object)

(2) A polynomial interactive oracle proof (PIOP)

(info. theoretic object)

SNARK for general circuits

Let's explain each concept …

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 $\mathbb{F}_p[X]$ of degree ≤ d.

(1) Polynomial commitment schemes

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

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

statement *st* = (*pp*, ***com_f***, x, y) with witness = w = (f, r)

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

[f(x) = y *and* f ∈ $\mathbb{F}_p^{(\leq d)}[X]$ *and* *commit*(*pp*, f, r) = ***com_f***]

(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)

Properties of the most widely used in practice (called KZG) :

- trusted setup: secret randomness in setup. $|pp| = O_\lambda(d)$
- ***com***$_f$: constant size (one group element)
- eval proof size: constant size (one group element)
- eval verify time: constant time.   Prover time: $O_\lambda(d)$

General paradigm: two steps

(1) A polynomial commitment scheme

(2) A polynomial interactive oracle proof (PIOP)

SNARK for general circuits

What is a PIOP?

Component 2: Polynomial IOP

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

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

Setup(C) → public parameters $\boldsymbol{pp}$ and $\boldsymbol{vp}$ = ($\boxed{f_0}$, $\boxed{f_{-1}}$, ... , $\boxed{f_{-s}}$)

Polynomial IOP

Prover P(pp, $\mathbf{x}$, $\mathbf{w}$) — Verifier V(vp, $\mathbf{x}$)

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

$r_1 \leftarrow \mathbb{F}_p$

r_1

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

$r_2 \leftarrow \mathbb{F}_p$

r_2

⋮

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

r_{t-1}

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

fast verify that can evaluate f_i at any point in $\mathbb{F}_p$ (outputs yes/no)

$$\text{verify}^{f_{-s},\ldots,f_t}(\mathbf{x}, r_1, , \ldots, r_{t-1})$$

The Plonk poly-IOP

<u>Goal</u>: construct a poly-IOP called ***Plonk*** (eprint/2019/953)

[Gabizon – Williamson – Ciobotaru]

$$\textit{Plonk} + \text{PCS} \Rightarrow \text{SNARK}$$

(and also a zk-SNARK)

[PCS = Polynomial Commitment Scheme]

First, a useful observation

A key fact: for non-zero $f \in \mathbb{F}_p^{(\leq d)}[X]$

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

⇒ 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

⇒ a simple zero test for a committed polynomial

SZDL lemma: $(*)$ also holds for **multivariate** polynomials (where d is total degree of f)

First, a useful observeration

Suppose $p \approx 2^{256}$ and $d \leq 2^{40}$ so that d/p is negligible

Let $f, g \in \mathbb{F}_p^{(\leq d)}[X]$.

For $r \leftarrow \mathbb{F}_p$, if $f(r) = g(r)$ then $f = g$ w.h.p

$f(r) - g(r) = 0 \Rightarrow f - g = 0$ w.h.p

$\Rightarrow$ a simple equality test for two committed polynomials

Useful proof gadgets

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

Set $\mathrm{H} := \{ 1, \omega, \omega^2, \ldots, \omega^{k-1} \} \subseteq \mathbb{F}_p$

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

There are efficient poly-IOPs for the following tasks:

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

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

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

Zero-test on H (H = { 1, ω, ω^2, ..., ω^{k-1} })

Prover P(f, ⊥)

Verifier V(f)

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

$q \in \mathbb{F}_p^{(\leq d)}[X]$ →

$r \leftarrow \mathbb{F}_p$

eval $q(X)$ and $f(X)$ at r ↔

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

Lemma: f is zero on H if and only if $f(X)$ is divisible by $X^k - 1$

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)

Another useful tool: permutation check

$W: H \to H$ is a **permutation of H** if $\forall i \in [k]$: $W(\omega^i) = \omega^j$

ex: $W(\omega^1) = \omega^{17}$, $W(\omega^2) = \omega^5$, $W(\omega^3) = \omega^2$, ...

Let $f, g: H \to H$ be polynomials in $\mathbb{F}_p^{(\leq d)}[X]$

Goal: given commitments to f, g, W prover want to prove that

$$f(y) = g(W(y)) \quad \text{for all } y \in H$$

$\Rightarrow$ Proves that $g(H)$ is the same as $f(H)$, just permuted by W

Another useful tool: permutation check

How? Use our <u>zero-test</u> to prove $f(y) - g(W(y)) = 0$ on H

<u>The problem</u>: the polynomial $f(y) - g(W(y))$ has degree k^2

⇒ prover would need to manipulate polynomials of degree k^2

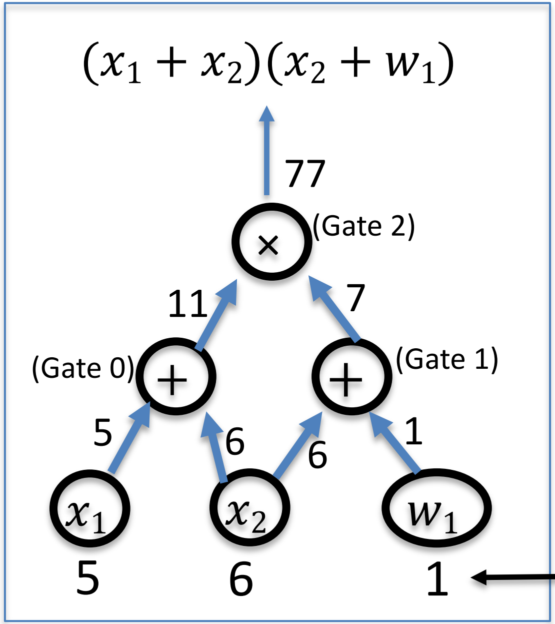
⇒ quadratic time prover !! (goal: linear time prover)

Cute trick: reduce this to a prod-check
on a polynomial of degree $2k$ (not k^2)

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:

	left inputs	right inputs	outputs
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

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

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

The plan: prover interpolates a polynomial

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

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,	77

Encoding the trace as a polynomial

The plan:

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

(1) ***T* encodes all inputs:** $T(\omega^{-j}) = \text{input } \#j$ for $j = 1, \ldots, |I|$

(2) ***T* encodes all wires:** $\forall\, l = 0, \ldots, |C| - 1$:

- $T(\omega^{3l})$: left input to gate $\#l$
- $T(\omega^{3l+1})$: right input to gate $\#l$
- $T(\omega^{3l+2})$: 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, Prover interpolates $T(X)$ such that:

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

degree(T) = 11

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

Prover uses FFT to compute the coefficients of T in time $d \log_2 d$

Step 2: proving validity of P

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

build T(X) ∈ $\mathbb{F}_p^{(\leq d)}[X]$

$\boxed{T}$ → (commitment)

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

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

(1) T encodes the correct inputs,

(2) every gate is evaluated correctly,

(3) the wiring is implemented correctly,

(4) the output of last gate is 0

(wiring constraints)

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

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

Proving (1): T 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:

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

In our example: $v(\omega^{-1}) = 5$, $v(\omega^{-2}) = 6$, $v(\omega^{-3}) = 1$. (v is quadratic)

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

$\Rightarrow$ verifier has time do this

Proving (1): T 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:

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

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

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

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

Proving (2): every gate is evaluated correctly

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

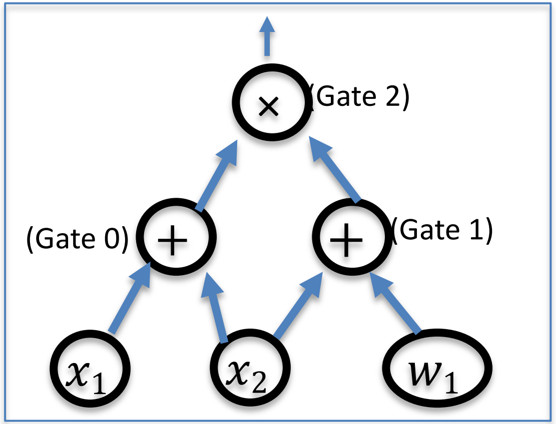
define S(X) $\in \mathbb{F}_p^{(\leq d)}[X]$ such that $\forall\, l = 0, \ldots, |C| - 1$:

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(ω^0) $= 1$, S(ω^3) $= 1$, S(ω^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(ω^{3l}) = 1 if gate #l is an addition gate

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

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

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

left input, right input, left input, right input, output

Proving (2): every gate is evaluated correctly

Setup(C) → pp:=S and vp:= (S)

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

build T(X) ∈ $\mathbb{F}_p^{(\leq d)}[X]$

T → (commitment)

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

Prover uses zero-test on the set H_{gates} to prove that ∀ y ∈ H_{gates}

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

Proving (3): the wiring is correct

Step 4: encode the wires of C:

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

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

	$\omega^{-1}, \omega^{-2}, \omega^{-3}$:	**5, 6, 1**
0:	$\omega^{0}, \omega^{1}, \omega^{2}$:	**5, 6, 11**
1:	$\omega^{3}, \omega^{4}, \omega^{5}$:	**6, 1, 7**
2:	$\omega^{6}, \omega^{7}, \omega^{8}$:	**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>: $\forall\, y \in H$: $T(y) = T(W(y))$ ⇒ wire constraints are satisfied

Proving (3): the wiring is correct

Step 4: encode the wires of C:

$$\begin{cases} T(\omega^{-2}) = T(\omega^{1}) = T(\omega^{3}) \\ T(\omega^{-1}) \ldots \\ T(\omega^{2}) \ldots \\ T(\omega^{-3}) \ldots \end{cases}$$

example: $x_1=5, x_2=6, w_1=1$

$\omega^{-1}, \omega^{-2}, \omega^{-3}$: **5, 6, 1**

…, **6, 11**

…, **1, 7**

…1, **7, 77**

Proved using a permutation check

Define a polynomia… H that implements a rotation:

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

Lemma: $\forall\, y \in H$: $T(y) = T(W(y)) \Rightarrow$ wire constraints are satisfied

The final Plonk Poly-IOP (and SNARK)

Setup(C) → pp := (S,W) and vp := (S and W) (untrusted)

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

build T(X) ∈ $\mathbb{F}_p^{(\leq d)}$[X]

→ T (commitment) →

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

build $v(X) \in \mathbb{F}_p^{(\leq |I_x|)}$ [X]

Prover proves:

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

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

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

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

The final Plonk Poly-IOP (and SNARK)

Setup(C) → pp := (S,W) and vp := (S and W) (untrusted)

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

build T(X) ∈ $\mathbb{F}_p^{(\leq d)}[X]$

T → (commitment)

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

build $v(X) \in \mathbb{F}_p^{(\leq |I_x|)}[X]$

Thm: The Plonk Poly-IOP is complete and knowledge sound

(eprint/2019/953)

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