Building a SNARK, Part II

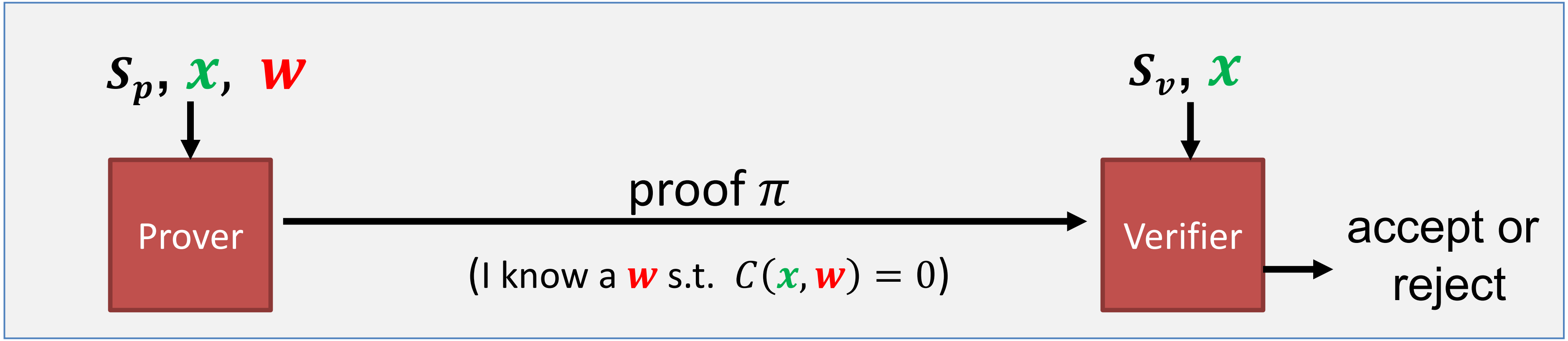
Dan Boneh
Stanford University

Review: Preprocessing argument systems

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

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

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



A SNARK paradigm: two steps

(1) A functional commitment scheme

(2) A compatible interactive oracle proof (IOP)

(zk)SNARK for general circuits

In this segment …

(1) A polynomial commitment scheme (PCS)

(2) A polynomial interactive oracle proof (**poly-IOP**)

(zk)SNARK for general circuits

(1) Polynomial Commitment Scheme (PCS)

A functional commitment for the family $\mathcal{F} = \mathbb{F}_p^{(\leq d)}[X]$

$\Longrightarrow$ prover commits to a univariate polynomial f in $\mathbb{F}_p^{(\leq d)}[X]$, later can prove that $v = f(u)$ for public $u, v \in \mathbb{F}_p$

Key point: **proof size** and **verifier time** are $O_\lambda(\boldsymbol{\log d})$

We saw several PCS constructions in the previous segment

(2) Polynomial IOP: prove $\exists w: C(\boldsymbol{x}, \boldsymbol{w}) = 0$

Setup(C) → public parameters $\boldsymbol{S_p}$ and $\boldsymbol{S_v}$ = ($\boxed{f_0}$, $\boxed{f_{-1}}$, … , $\boxed{f_{-s}}$)

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

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

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

$r_1 \leftarrow \mathbb{F}_p$

← r_1

⋮

← 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 point in $\mathbb{F}_p$ (outputs yes/no)

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

In this segment …

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

[Gabizon – Williamson – Ciobotaru]

Plonk + PCS $\Rightarrow$ SNARK

(and also a zk-SNARK)

First, a useful observation

A key fact: for $0 \neq f \in \mathbb{F}_p^{(\leq d)}[X]$

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

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

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

$\Rightarrow$ a simple zero test for a committed polynomial

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

A related observation

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

$$f(r) - g(r) = 0 \quad \Rightarrow \quad f - g = 0 \text{ w.h.p}$$

$\Rightarrow$ 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, ..., $\omega^{k\text{-}1}$ })

Prover P(f, ⊥)

Verifier V($\boxed{f}$)

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

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

$r \leftarrow \mathbb{F}_p$

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

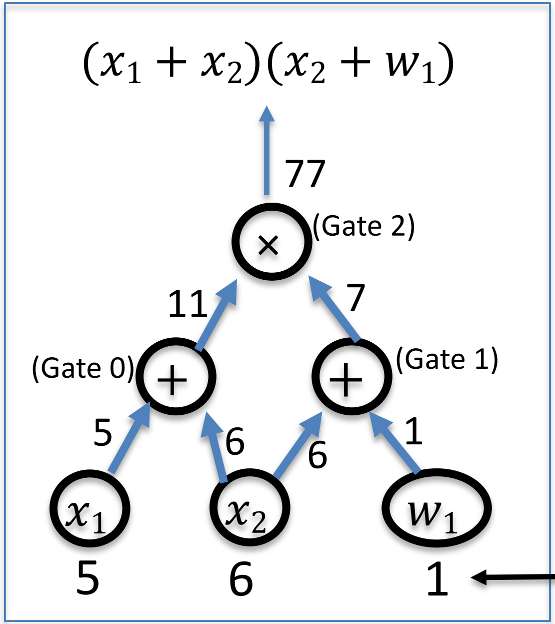
<u>Thm</u>: this protocol is complete and sound, assuming d/p is negligible.

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

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,	$\boxed{77}$

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

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

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

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

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

- $P(\omega^{3l})$: 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, Prover interpolates $P(X)$ such that:

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:	$P(\omega^{6}) = 11,$	$P(\omega^{7}) = 7,$	$P(\omega^{8}) = 77$

degree(P) = 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 P in time $d \log_2 d$

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] $\xrightarrow[\text{(commitment)}]{\boxed{P}}$

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 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 $P(\omega^{3|C|-1}) = 0$

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:

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

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

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

$\Rightarrow$ verifier has time do this

Proving (1): 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:

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

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)

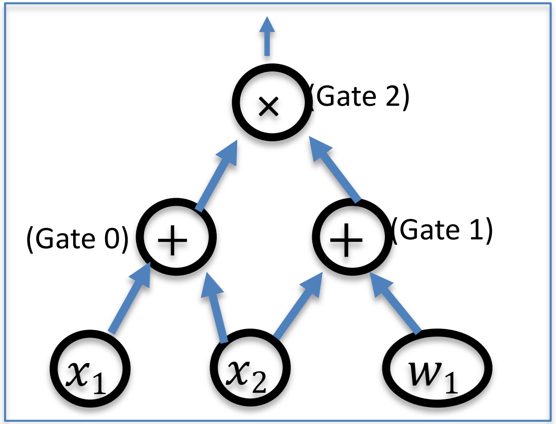
define S(X) $\in \mathbb{F}_p^{(\leq d)}$[X] such that $\forall\, l = 0, \ldots, |C| - 1$:

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

$$S(y) \cdot \underbrace{[P(y)}_{\text{left input}} + \underbrace{P(\omega y)]}_{\text{right input}} + (1 - S(y)) \cdot \underbrace{P(y)}_{\text{left input}} \cdot \underbrace{P(\omega y)}_{\text{right input}} = \underbrace{P(\omega^2 y)}_{\text{output}}$$

Proving (2): every gate is evaluated correctly

$\text{Setup}(C) \rightarrow \quad S_p := S$ and $S_v := (\boxed{S})$

<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 the set 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
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 $W: H \to H$ that implements a rotation:

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

Lemma: $\forall\, y \in H$: $P(y) = P(W(y))$ $\Rightarrow$ 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 $x \in H$

(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 $x \in H$, where

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

The final Plonk Poly-IOP (and SNARK)

Setup(C) → S_p := (S,W) and S_v := ($\boxed{S}$ and $\boxed{W}$) (untrusted)

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

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)

The final Plonk Poly-IOP (and SNARK)

Setup(C) → S_p := (S,W) and S_v := ($\boxed{S}$ and $\boxed{W}$)

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

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

$\boxed{P}$

(commitment)

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

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 (≈400 bytes), fast verifier (≈6ms)

- Can handle circuits with more general gates than $+$ and $\times$
  - PLOOKUP: efficient Poly-IOP for circuits with lookup tables

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

Main research effort: SNARKs with faster prover time

END OF LECTURE