Implementation of plonk in python

- Author: Ying Yue(Yolanda)

- Github repo: https://github.com/YolaYing/zk-toolkit

Preliminary

- **Two Forms of Polynomial Representations**
  - Coefficient Form
  - Evaluation Form

- **Convertion between Two Forms**
  - Fourier Transform
  - Inverse Fourier Transform

Component of a Proof System

- Computation Trace

- Interactive Oracle Proof(IOP)

Cryptography Compiler

- Proof System

1 Preliminary

- **Two Forms of Polynomial Representations**
  - Coefficient Form
  - Evaluation Form
- **Convertion between Two Forms**
  - Fourier Transform
  - Inverse Fourier Transform

Component of a Proof System

- Computation Trace ← Regard as given
- Interactive Oracle Proof(IOP) ← Polynomial IOP
- 2 **Cryptography Compiler** ← KZG
- 3 **Proof System** ← Plonk

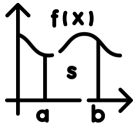
1 Preliminary

Two Forms of Polynomial

- Coefficient Form $\Phi(x) = 4 * x^3 + 5 * x^2 + 3 * x + 2$

- Evaluation Form 4 points: (1, 14), (4, 350), (16, 17714), (13, 9674)

Preliminary



Two Forms of Polynomial

$\boldsymbol{\Phi(x) \in F_q[x]}$

- Coefficient Form

$$\Phi(x) = \boxed{4} * x^3 + \boxed{5} * x^2 + \boxed{3} * x + \boxed{2}$$

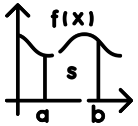
Every coefficient here need to 'mod q'

- Evaluation Form

4 points: ($\boxed{1}$, $\boxed{14}$), ($\boxed{4}$, $\boxed{350}$), ($\boxed{16}$, $\boxed{17714}$), ($\boxed{13}$, $\boxed{9674}$)

Every values of point here need to 'mod q'

Preliminary



Two Forms of Polynomial

$\boldsymbol{\Phi(x) \in F_q[x]}$

assume q = 17

- Coefficient Form

$$\Phi(x) = 4 * x^3 + 5 * x^2 + 3 * x + 2$$

Every coefficient here need to 'mod 17'

- Evaluation Form

4 points: (1, 14), (4, 10), (16, 0), (13, 1)

Every values of point here need to 'mod 17'

1 Preliminary

Two Forms of Polynomial $\Phi(x) \in F_q[x]$

- Coefficient Form

$$\Phi(x) = 4 * x^3 + 5 * x^2 + 3 * x + 2$$

represented as a tuple of n coefficients: [2, 3, 5, 4]

- Evaluation Form

4 points: (1, 14), (4, 10), (16, 0), (13, 1)

represented as
- a tuple of n distinct evaluations: [14, 10, 0, 1]
- evaluation domain:[1, 4, 16, 13]

1 Preliminary

Two Forms of Polynomial $\Phi(x) \in F_q[x]$

- Coefficient Form $\Phi(x) = 4 * x^3 + 5 * x^2 + 3 * x + 2$ [2, 3, 5, 4]

Fourier Transform

Inverse Fourier Transform

- Evaluation Form 4 points: (1, 14), (4, 10), (16, 0), (13, 1)
  - evaluations: [14, 10, 0, 1]
  - evaluation domain:[1, 4, 16, 13]

1 Preliminary

Two Forms of Polynomial $\Phi(x) \in F_q[x]$

- Coefficient Form $\Phi(x) = 4 * x^3 + 5 * x^2 + 3 * x + 2$ [2, 3, 5, 4]

Fourier Transform

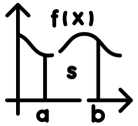
Inverse Fourier Transform

Naïve way: $\Phi(1), \Phi(4), \Phi(16), \Phi(13)$

- Evaluation Form 4 points: (1, 14), (4, 10), (16, 0), (13, 1)

$\left\{ \begin{array}{l} \text{evaluations: [14, 10, 0, 1]} \\ \text{evaluation domain:[1, 4, 16, 13]} \end{array} \right.$

1 Preliminary



Two Forms of Polynomial $\Phi(x) \in F_q[x]$

- Coefficient Form $\Phi(x) = 4 * x^3 + 5 * x^2 + 3 * x + 2$ [2, 3, 5, 4]

Fourier Transform ↓ ↑ Inverse Fourier Transform

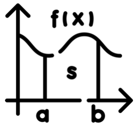
Naïve way: Lagrange Interpolation

$$\Phi(x) = \sum_{j=0}^{4} y_j l_j(x) \quad l_j(x) = \prod_{\substack{0 \le m \le 4 \\ m \neq j}} \frac{x - x_m}{x_j - x_m}$$

- Evaluation Form 4 points: (1, 14), (4, 10), (16, 0), (13, 1)

$\begin{cases} \text{evaluations: [14, 10, 0, 1]} \\ \text{evaluation domain:[1, 4, 16, 13]} \end{cases}$

Preliminary



Two Forms of Polynomial $\Phi(x) \in F_q[x]$

- Coefficient Form $\Phi(x) = 4 * x^3 + 5 * x^2 + 3 * x + 2$ [2, 3, 5, 4]

Fourier Transform

Inverse Fourier Transform

Optimized way: Fast Fourier Algorithm(FFT)

Inverse Fast Fourier Algorithm(IFFT)

- Evaluation Form 4 points: (1, 14), (4, 10), (16, 0), (13, 1)

$\begin{cases} \text{evaluations: [14, 10, 0, 1]} \\ \text{evaluation domain:[1, 4, 16, 13]} \end{cases}$

If you are not familiar with FFT, a good reference here: https://www.youtube.com/watch?v=h7apO7q16V0

2 Commitment Scheme: KZG

The KZG Commitment Scheme is a commitment scheme that allows **to commit** to a polynomial $\Phi(x)$, where $\Phi(x) \in F_q[x]$

to commit means proving that you know the polynomial $\Phi(x)$ without revealing it

2 Commitment Scheme: KZG

The KZG commitment scheme consists of 4 steps:

Step 1: Setup

Step 2: Commit to Polynomials

Step 3: Prove an Evaluation

Step 4: Verify an Evaluation Proof

Commitment Scheme: KZG

1: Setup

2: Commit to Polynomials

3: Prove an Evaluation

4: Verify an Evaluation Proof

Main purpose: $Setup(1^\lambda) \rightarrow (pk, vk)$

$G_1 \quad G_2$: pairing-friendly elliptic curve groups, determined by curve BLS12-381

g_1 : a generator of G_1

g_2 : a generator of G_2

F_p : finite field with order p

l : the maximum degree of the polynomials we want to commit to $(l < p)$

τ : secret parameter, a randomly picked field element[1] $\tau \in F_p$

Compute public parameters(pk & vk): $pk = (g_1, g_1^{\tau}, g_1^{\tau^2}, \ldots, g_1^{\tau^l}) \qquad vk = g_2^{\tau}$

1. usually done by MPC, to simplify, we just randomly choose one here

2 Commitment Scheme: KZG

- 1: Setup
- 2: Commit to Polynomials
- 3: Prove an Evaluation
- 4: Verify an Evaluation Proof

Main purpose: $Setup(1^{\lambda}) \rightarrow (pk, vk)$

$G_1 \quad G_2$: pairing-friendly elliptic curve groups, de

g_1 : a generator of G_1

g_2 : a generator of G_2

F_p : finite field with order p

l : the maximum degree of the polynomials we

τ : secret parameter, a randomly picked field element[1] $\tau \in F_p$

IMPORTANT

Do not forget to delete τ

Compute public parameters(pk & vk): $pk = (g_1, g_1^{\tau}, g_1^{\tau^2}, \dots, g_1^{\tau^l}) \qquad vk = g_2^{\tau}$

1. usually done by MPC, to simplify, we just randomly choose one here

2 Commitment Scheme: KZG

1: Setup

2: Commit to Polynomials

3: Prove an Evaluation

4: Verify an Evaluation Proof

Main purpose: $commit(pk, \Phi) \rightarrow com_{\Phi}\ \ where\ \ com_{\Phi} = g_1^{\Phi(\tau)} \in G_1$

Given a polynimial $\Phi(x) = \phi_0 + \phi_1 x + \phi_2 x^2 + \cdots + \phi_d x^d, \quad d \leq l$

Compute commitment $com_{\Phi} = g_1^{\Phi(\tau)}$

so easy...

Commitment Scheme: KZG

1: Setup

2: Commit to Polynomials

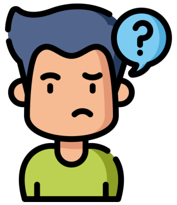
3: Prove an Evaluation

4: Verify an Evaluation Proof

Main purpose: $commit(pk, \Phi) \rightarrow com_{\Phi}$ $where$ $com_{\Phi} = g_1^{\Phi(\tau)} \in G_1$

Given a polynimial $\Phi(x) = \phi_0 + \phi_1 x + \phi_2 x^2 + \cdots + \phi_d x^d, \quad d \leq l$

Compute commitment $com_{\Phi} = g_1^{\Phi(\tau)}$



Wait! τ has been discarded already, right?

How can we compute $\Phi(\tau)$ directly?

Commitment Scheme: KZG

1: Setup

2: Commit to Polynomials

3: Prove an Evaluation

4: Verify an Evaluation Proof

Main purpose: $commit(pk, \Phi) \rightarrow com_\Phi \;\; where \;\; com_\Phi = g_1^{\Phi(\tau)} \in G_1$

Given a polynimial $\Phi(x) = \phi_0 + \phi_1 x + \phi_2 x^2 + \cdots + \phi_d x^d, \qquad d \leq l$

Compute commitment

$$
\begin{aligned}
com_\Phi &= g_1^{\Phi(\tau)} \\
&= g_1^{\phi_0 + \phi_1 \tau + \phi_2 \tau^2 + \cdots + \phi_d \tau^d} \\
&= (g_1)^{\phi_0} \cdot (g_1^{\tau})^{\phi_1} \cdot \left(g_1^{\tau^2}\right)^{\phi_2} \cdot \cdots \cdot \left(g_1^{\tau^d}\right)^{\phi_d}
\end{aligned}
$$

$pk = (g_1, g_1^{\tau}, g_1^{\tau^2}, \ldots, g_1^{\tau^l})$

2 Commitment Scheme: KZG

1: Setup

2: Commit to Polynomials

3: Prove an Evaluation

4: Verify an Evaluation Proof

Main purpose: Verifier gives a random value $a, a \in F_p$

Prover opens $\Phi(x)$ at a, and prove $\Phi(a) = b$

Prover

Randomly choose $a, a \in F_p$

Calculate $\Phi(a) = b$, and a proof π

Verifier

Goal: to prove $\Phi(a) = b$

$$\Phi(a) = b \Leftrightarrow a \ is \ a \ root \ of \ \Phi(x) - b$$

$$\Leftrightarrow (x - a) \ divides \ \Phi(x) - b$$

$$\Leftrightarrow \exists \ q \in F_p[X], s.t. \ q(x)(x - a) = \Phi(x) - b$$

2 Commitment Scheme: KZG

1: Setup

2: Commit to Polynomials

3: Prove an Evaluation

4: Verify an Evaluation Proof

Main purpose: Verifier gives a random value $a, a \in F_p$

Prover opens $\Phi(x)\ at\ a$, and prove $\Phi(a) = b$

Prover

Randomly choose $a, a \in F_p$

Calculate $\Phi(a) = b$, and a proof π

Verifier

Compute $q(x)$ $$q(x) \coloneqq \frac{\Phi(x) - b}{x - a}$$

Calculate commitment of com_q as proof π $$\pi = com_q = g_1^{q(\tau)}$$

2 Commitment Scheme: KZG

1: Setup

2: Commit to Polynomials

3: Prove an Evaluation

4: Verify an Evaluation Proof

Main purpose: Verifier accept if $q(x)(x-a) = \Phi(x) - b \; holds \; for \; x = \tau$

Verifier

All the stuff I have known:

- commitment com_Φ, $\quad com_\Phi = g_1^{\Phi(\tau)} \in G_1$
- evaluation result (a, b), $\quad \Phi(a) = b$
- proof π, $\quad \pi = com_q = g_1^{q(\tau)}$

And my purpose of verification: $q(\tau)(\tau - a) = \Phi(\tau) - b$

τ appears again!

2 Commitment Scheme: KZG

1: Setup

2: Commit to Polynomials

3: Prove an Evaluation

4: Verify an Evaluation Proof

Main purpose: Verifier accept if $q(x)(x-a) = \Phi(x) - b \ holds \ for \ x = \tau$

Verifier

All the stuff I have known:

- commitment com_Φ, $com_\Phi = g_1^{\Phi(\tau)} \in G_1$
- evaluation result (a, b), $\Phi(a) = b$
- proof π, $\pi = com_q = g_1^{q(\tau)}$

And my purpose of verification: $q(\tau)(\tau - a) = \Phi(\tau) - b$

τ appears again!

We use the bilinear property of ”pairing”:
for any points P, Q on the curve, and scalars a, b: $e\left(P^a, Q^b\right) = e(P,Q)^{ab}$

Commitment Scheme: KZG

- 1: Setup
- 2: Commit to Polynomials
- 3: Prove an Evaluation
- **4: Verify an Evaluation Proof**

Main purpose: Verifier accept if $q(x)(x-a) = \Phi(x) - b \text{ holds for } x = \tau$

All the stuff I have known:

- commitment com_Φ, $\quad com_\Phi = g_1^{\Phi(\tau)} \in G_1$
- evaluation result (a, b), $\quad \Phi(a) = b$
- proof π, $\quad \pi = com_q = g_1^{q(\tau)}$

And my purpose of verification: $q(\tau)(\tau - a) = \Phi(\tau) - b$

$$q(\tau)(\tau - a) = \Phi(\tau) - b \Leftrightarrow e(g_1, g_2)^{q(\tau)(\tau - a)} = e(g_1, g_2)^{\Phi(\tau) - b}$$

$$\Leftrightarrow e(\underbrace{\boxed{g_1^{q(\tau)}}}_{\pi}, \underbrace{\boxed{g_2^{\tau}}}_{vk}{}^{-a}) = e(\underbrace{\boxed{g_1^{\Phi(\tau)}}}_{com_\Phi}{}^{-b}, g_2)$$

$$\Leftrightarrow e(\pi, vk - g_2^{a}) = e(com_\Phi - g_1^{b}, g_2)$$

2 Commitment Scheme: KZG review

1: Setup

$pk = (g_1, g_1^{\tau}, g_1^{\tau^2}, \dots, g_1^{\tau^l})$ $\quad vk = g_2^{\tau}$, delete τ

2: Commit to Polynomials

Prover $\xrightarrow{com_{\Phi} = g_1^{\Phi(\tau)}}$ Verifier

3: Prove an Evaluation

Verifier → Prover: Randomly choose $a, a \in F_p$

Prover → Verifier: Calculate $\Phi(a) = b$, and a proof π

4: Verify an Evaluation Proof

Verifier: accept if $q(x)(x - a) = \Phi(x) - b \; holds \; for \; x = \tau$

3 Prove System: Plonk

Let's take **Square-Fibonacci** as an example to demonstrate the process of proof generation

Defination of **Square-Fibonacci** problem

- Let $f_0 = 1, f_1 = 1$
- For $i \geq 2$, define $f_i := (f_{i-2})^2 + (f_{i-1})^2 \; mod \; q$, q is a large prime number

n : a large number

k : n^{th} Square-Fibonacci number

Our Goal: Generate an efficiently-verifiable proof π , to prove $f_n = k$

This section is inspired by this [wonderful tutorial] from Scroll

3 Prove System: Plonk

The Plonk-based proof generation consists of 3 steps:

Step 1: Filling in the trace table

Step 2: Committing to the trace table

Step 3: Proving the trace table's correctness

Prove System: Plonk

1: Fill in the trace table

2: Commit to the trace table

3: Prove the correctness

A	B	C	S	P
f_0	f_1	f_2	1	f_0
f_1	f_2	f_3	1	f_1
f_2	f_3	f_4	1	k
…	…	…	…	…
f_{n-3}	f_{n-2}	f_{n-1}	1	
f_{n-2}	f_{n-1}	f_n	1	
			0	

element of F_q

A, B, C : witness data, each row lists 3 sequential Square-Fibonacci numbers

S : selector column, indicating a certain mathematical relation should hold over the element of the row

P : public inputs, inputs to the circuit that are public known

3 Prove System: Plonk

1: Fill in the trace table

2: Commit to the trace table

3: Prove the correctness

A	B	C	S	P
f_0	f_1	f_2	1	f_0
f_1	f_2	f_3	1	f_1
f_2	f_3	f_4	1	k
…	…	…	…	…
f_{n-3}	f_{n-2}	f_{n-1}	1	
f_{n-2}	f_{n-1}	f_n	1	
			0	

$n = 8$

A	B	C	S	P
1	1	2	1	1
1	2	5	1	1
2	5	29	1	317754178345286893212434
…	…	…	…	…
866	750797	563696885165	1	
750797	563696885165	317754178345286893212434	1	
			0	

3 Prove System: Plonk

1: Fill in the trace table

2: Commit to the trace table

3: Prove the correctness

A	B	C	S	P
f_0	f_1	f_2	1	f_0
f_1	f_2	f_3	1	f_1
f_2	f_3	f_4	1	k
…	…	…	…	…
f_{n-3}	f_{n-2}	f_{n-1}	1	
f_{n-2}	f_{n-1}	f_n	1	
			0	

regrad every column as polynomials **in point-value form**

IFFT

Convert to coefficient form $A(x)$

Reuse code in KZG

Commit to each of them $com_A = g_1^{A(\tau)}$

3 Prove System: Plonk

1: Fill in the trace table

2: Commit to the trace table

3: Prove the correctness

To prove the whole computation is valid, we need to fulfill 2 kinds of contraints:

- In vanilla plonk:

Gate Constraints

Make sure all the gates are **correctly computed**

Wiring Constraints

Make sure all the gates are **correctly connected**

- With some variation:

Custom Constraints

- Square-Fibonacci constraints

Wiring Constraints

Public input Constraints

3 Prove System: Plonk

1: Fill in the trace table

2: Commit to the trace table

3: Prove the correctness

To prove the whole computation is valid, we need to fulfill 2 kinds of contraints:

- In vanilla plonk:

will be explained with plonk paper

Gate Constraints

Make sure all the gates are **correctly computed**

Wiring Constraints

Make sure all the gates are **correctly connected**

- With some variation:

Custom Constraints

- Square-Fibonacci constraints

Wiring Constraints

Public input Constraints

Prove System: Plonk

1: Fill in the trace table

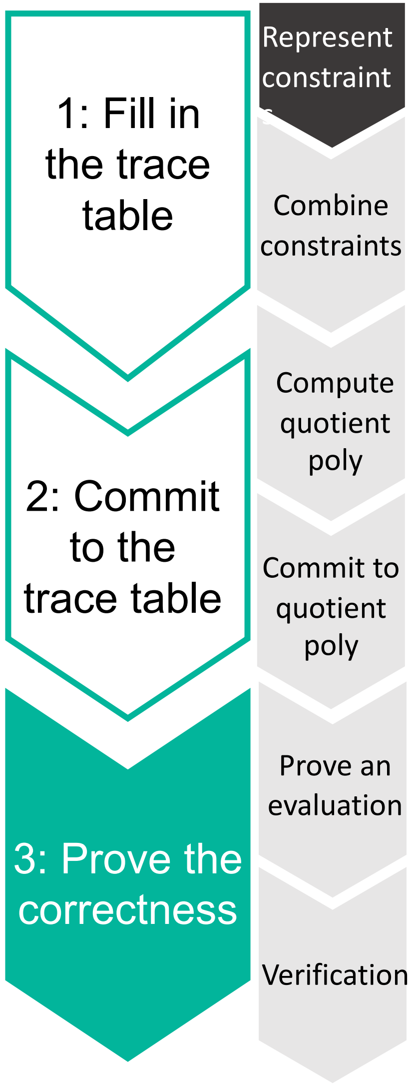
2: Commit to the trace table

3: Prove the correctness

To prove the correctness, we have the following 6 steps:

- Use column polynomials to represent all the constraints
- Combine all the contraints together
- Compute quotient polynomial
- Commit to quotient polynimial
- Prove an evaluation
- Conduct verification

3 Prove System: Plonk



- **Square-Fibonacci constraints**
  - For each line i: the first 3 elements (a, b, c) must satisfy $a_i^2 + b_i^2 = c_i \bmod q$
- **Wiring constraints**
  - For consecutive rows with value $[a_i, b_i, c_i]$ and $[a_{i+1}, b_{i+1}, c_{i+1}]$ must satisfy $a_{i+1} = b_i, b_{i+1} = c_i$
- **Public input constraints**
  - In the puclic inputs column, we require $a_0 = p_0, b_0 = p_1, c_{n-2} = p_2$

3 Prove System: Plonk

1: Fill in the trace table

2: Commit to the trace table

3: Prove the correctness

Represent constraints

Combine constraints

Compute quotient poly

Commit to quotient poly

Prove an evaluation

Verification

- **Square-Fibonacci constraints**
  - For each line i: the first 3 elements (a, b, c) must satisfy $a_i^2 + b_i^2 = c_i \bmod q$

$$S(x) \cdot \left(A(x)^2 + B(x)^2 - C(x)\right) = 0, \qquad \forall x \in \{\omega^0, \omega^1, \dots \omega^{n-1}\}$$

- **Wiring constraints**
  - For consecutive rows with value $[a_i, b_i, c_i]$ and $[a_{i+1}, b_{i+1}, c_{i+1}]$ must satisfy $a_{i+1} = b_i, b_{i+1} = c_i$

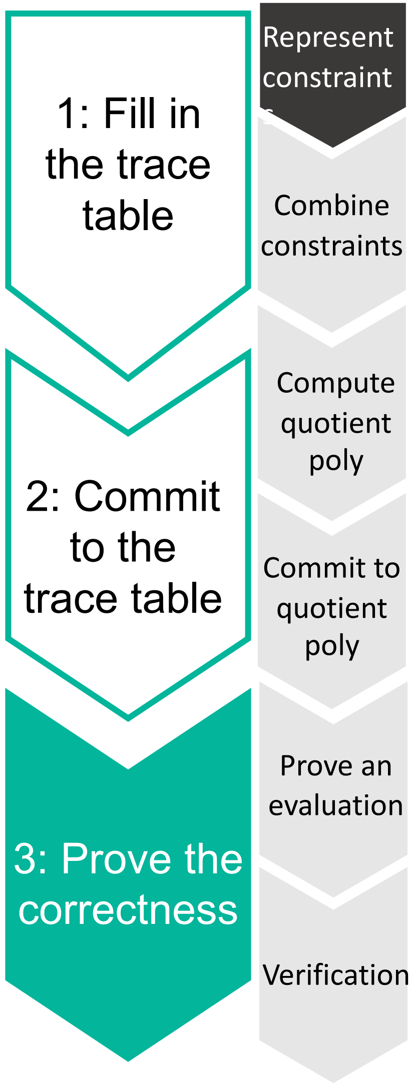
$$S'(x) \cdot \left(A(\omega x) - B(x)\right) = 0, S'(x) = S(\omega x), \qquad \forall x \in \{\omega^0, \omega^1, \dots \omega^{n-1}\}$$

$$S'(x) \cdot \left(B(\omega x) - C(x)\right) = 0, S'(x) = S(\omega x), \qquad \forall x \in \{\omega^0, \omega^1, \dots \omega^{n-1}\}$$

- **Public input constraints**
  - In the puclic inputs column, we require $a_0 = p_0, b_0 = p_1, c_{n-2} = p_2$

can be done by construct aux. selectors, but that's not the focus here…

Prove System: Plonk



For shorten, we label left-hand side:

- $\phi_0(x) \coloneqq S(x) \cdot \left(A(x)^2 + B(x)^2 - C(x)\right)$
- $\phi_1(x) \coloneqq S'(x) \cdot \left(A(\omega x) - B(x)\right),\ S'(x) = S(\omega x)$
- $\phi_2(x) \coloneqq S'(x) \cdot \left(B(\omega x) - C(x)\right),\ S'(x) = S(\omega x)$

…

All the contraints can be expressed as $\phi_i(x) = 0, \quad \forall x \in \{\omega^0, \omega^1, \dots \omega^{n-1}\}$

3 Prove System: Plonk

1: Fill in the trace table

2: Commit to the trace table

3: Prove the correctness

Represent constraints

Combine constraints

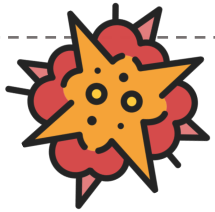
Compute quotient poly

Commit to quotient poly

Prove an evaluation

Verification

Now we have gotten m constraint polynomials $\phi_0(x), \phi_1(x), \dots, \phi_{m-1}(x)$



Committing to those polynomials one by one seems to be computational-intensive…

Why not batch them together?

Prove System: Plonk

1: Fill in the trace table

2: Commit to the trace table

3: Prove the correctness

Represent constraints

Combine constraints

Compute quotient poly

Commit to quotient poly

Prove an evaluation

Verification

Now we have gotten m constraint polynomials $\phi_0(x), \phi_1(x), \dots, \phi_{m-1}(x)$

Randomly sample a field element $\gamma \in F_q$, and then take a random linear combination of the individual constraints:

$$\Phi(x) \coloneqq \gamma_0 \cdot \phi_0(x) + \gamma_1 \cdot \phi_1(x) + \cdots + \gamma_{m-1} \cdot \phi_{m-1}(x)$$

satisfies at every row, that is $\Phi(\omega^i) = 0, \ \forall i \ 0 \leq i < n$

Now the task has become ‘prove $\Phi(x)$ holds for each row of the trace table’

Prove System: Plonk

1: Fill in the trace table

2: Commit to the trace table

3: Prove the correctness

Represent constraints

Combine constraints

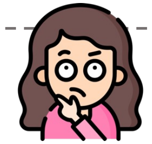
Compute quotient poly

Commit to quotient poly

Prove an evaluation

Verification

Prove $\Phi(x) = 0, \boxed{\forall x \in \{\omega^0, \omega^1, \ldots \omega^{n-1}\}}$



Due to the strange domain, it is not easy to prove directly…

Prove System: Plonk

1: Fill in the trace table

2: Commit to the trace table

3: Prove the correctness

Represent constraints

Combine constraints

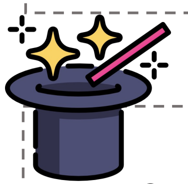
Compute quotient poly

Commit to quotient poly

Prove an evaluation

Verification

Prove $\Phi(x) = 0, \ \forall x \in \{\omega^0, \omega^1, \ldots \omega^{n-1}\}$



Some tricks here:

$x - \omega^i$ is the root of $\Phi(x)$

$$\Phi(x) = 0, \forall x \in \{\omega^0, \omega^1, \ldots, \omega^{n-1}\} \Leftrightarrow \boxed{(x - \omega^i)} | \Phi(x), \forall x \in \{\omega^0, \omega^1, \ldots, \omega^{n-1}\}$$

$$\Leftrightarrow \prod_{i=0}^{n-1} (x - \omega^i) | \Phi(x)$$

$$\Leftrightarrow \boxed{(x^n - 1)} | \Phi(x)$$

$$\Leftrightarrow \exists Q(x) \ s.t. \Phi(x) = Q(x) \cdot (x^n - 1)$$

by polynomial remainder theorem

$$\prod_{i=0}^{n-1} (x - \omega^i) = (x^n - 1)$$

Now the task becomes 'prove the existence of $Q(x)$'

3 Prove System: Plonk

1: Fill in the trace table

2: Commit to the trace table

3: Prove the correctness

Represent constraints

Combine constraints

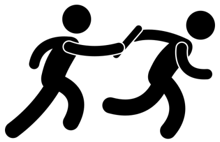
Compute quotient poly

Commit to quotient poly

Prove an evaluation

Verification

Now we have the quotient polynomial $Q(x) \coloneqq \frac{\Phi(x)}{x^n - 1} = \frac{\gamma_0 \cdot \phi_0(x) + \gamma_1 \cdot \phi_1(x) + \cdots + \gamma_{m-1} \cdot \phi_{m-1}(x)}{x^n - 1}$



Take $Q(x)$ as the input of KZG and everything is done?

$pk = (g_1, g_1^{\tau}, g_1^{\tau^2}, \ldots, g_1^{\tau^l})$

maxium degree = l

Degree of $Q(x)$:

- $\phi_0(x) \coloneqq S(x) \cdot \left(A(x)^2 + B(x)^2 - C(x)\right)$

  maxium degree = 3n-3

- $Q(x) \coloneqq \frac{\Phi(x)}{x^n - 1}$

  maxium degree = 2n-2

  → round to 2n

3 Prove System: Plonk

1: Fill in the trace table

2: Commit to the trace table

3: Prove the correctness

Represent constraints

Combine constraints

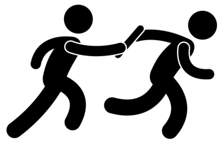
Compute quotient poly

Commit to quotient poly

Prove an evaluation

Verification

Now we have the quotient polynomial $Q(x) \coloneqq \frac{\Phi(x)}{x^n - 1} = \frac{\gamma_0 \cdot \phi_0(x) + \gamma_1 \cdot \phi_1(x) + \cdots + \gamma_{m-1} \cdot \phi_{m-1}(x)}{x^n - 1}$



Take $Q(x)$ as the input of KZG and everything is done?

Yes, but require a larger KZG setup

$pk = (g_1, g_1^{\tau}, g_1^{\tau^2}, \ldots, g_1^{\tau^l})$

maxium degree = l

Degree of $Q(x)$:

- $\phi_0(x) \coloneqq S(x) \cdot \left(A(x)^2 + B(x)^2 - C(x)\right)$

  maxium degree = 3n-3

- $Q(x) \coloneqq \frac{\Phi(x)}{x^n - 1}$

  maxium degree = 2n-2

  → round to 2n

3 Prove System: Plonk

1: Fill in the trace table

2: Commit to the trace table

3: Prove the correctness

Represent constraints

Combine constraints

Compute quotient poly

Commit to quotient poly

Prove an evaluation

Verification

Now we have the quotient polynomial $\mathrm{Q}(x) := \frac{\Phi(x)}{x^n-1} = \frac{\gamma_0 \cdot \phi_0(x) + \gamma_1 \cdot \phi_1(x) + \cdots + \gamma_{m-1} \cdot \phi_{m-1}(x)}{x^n-1}$

Expand setup of KZG

$$pk = \left(g_1, g_1^{\tau}, g_1^{\tau^2}, \dots, g_1^{\tau^l}\right) \rightarrow pk' = (g_1, g_1^{\tau}, g_1^{\tau^2}, \dots, g_1^{\tau^{2n}})$$

Commit to quotient polynomial $Q(x)$

$$com_{\mathrm{Q}} = g_1^{\mathrm{Q}(\tau)}$$

Prove System: Plonk

1: Fill in the trace table

2: Commit to the trace table

3: Prove the correctness

Represent constraints

Combine constraints

Compute quotient poly

Commit to quotient poly

Prove an evaluation

Verification

We have gotten all the column polynomial and quotient polynomial:

- Column Polynomials: $A(x), B(x), C(x), S(x), P(x)$
- Quotient Polynomial: $Q(x)$

and all the commitment of those polynomials :

$$com_A, com_B, com_C, com_S, com_P, com_Q$$

The last two steps are just following the logic we introduced in KZG

3 Prove System: Plonk

2 Commitment Scheme: KZG review

1: Setup

$pk = (g_1, g_1^{\tau}, g_1^{\tau^2}, \ldots, g_1^{\tau^l})$ $vk = g_2^{\tau}$, delete τ

2: Commit to Polynomials

$com_{\Phi} = g_1^{\Phi(\tau)}$

Prover

Verifier

3: Prove an Evaluation

Randomly choose $a, a \in F_p$

Prover

Calculate $\Phi(a) = b$, and a proof π

Verifier

4: Verify an Evaluation Proof

accept if $q(x)(x-a) = \Phi(x) - b$ $holds\ for\ x = \tau$

Verifier

olynomial:

$_P, com_Q$

correctness

Verification

The last two steps are just following the logic we introduced in KZG

3 Prove System: Plonk

1: Fill in the trace table

2: Commit to the trace table

3: Prove the correctness

Represent constraints

Combine constraints

Compute quotient poly

Commit to quotient poly

Prove an evaluation

Verification

We have gotten all the column polynomial and quotient polynomial:

- Column Polynomials: $A(x), B(x), C(x), S(x), P(x)$
- Quotient Polynomial: $Q(x)$

and all the commitment of those polynomials :

$$com_A, com_B, com_C, com_S, com_P, com_Q$$

Randomly choose $\alpha, \alpha \in F_p$

Prover

Calculate $\mathrm{A}(\alpha) = \beta_0, B(\alpha) = \beta_1, C(\alpha) = \beta_2, S(\alpha) = \beta_3, P(\alpha) = \beta_4,$ $Q(\alpha) = \beta_Q$, and corresponding proof $\pi_0, \pi_1, \pi_2, \pi_3, \pi_4, \pi_Q$

Verifier

3 Prove System: Plonk

1: Fill in the trace table

2: Commit to the trace table

3: Prove the correctness

Represent constraints

Combine constraints

Compute quotient poly

Commit to quotient poly

Prove an evaluation

Verification

Verifier

accept if

$$\begin{cases} q_A(x)(x-\alpha) = \mathrm{A}(x) - \beta_0 \ holds\ for\ x = \tau \\ q_B(x)(x-\alpha) = \mathrm{B}(x) - \beta_1 \ holds\ for\ x = \tau \\ q_C(x)(x-\alpha) = \mathrm{C}(x) - \beta_2 \ holds\ for\ x = \tau \\ q_S(x)(x-\alpha) = \mathrm{S}(x) - \beta_3 \ holds\ for\ x = \tau \\ q_P(x)(x-\alpha) = \mathrm{P}(x) - \beta_4 \ holds\ for\ x = \tau \\ q_Q(x)(x-\alpha) = \mathrm{Q}(x) - \beta_Q \ holds\ for\ x = \tau \end{cases}$$

Appendix: intro to vanilla plonk