eryx

zkCity

Día 5: Plonk

¿Qué vamos a ver?

- Repasito de KZG
- Fibonacci con KZG
- Plonk
- Optimizaciones de Plonk

Curvas elípticas

$$y^2 = x^3 + ax + b$$

G punto de curva, el subgrupo cíclico generado por G

$$\{G, 2G, 3G, ..., (q-1)G\}$$

Pairing: $$e(aP, bQ) = e(P, Q)^{ab}$$

KZG

Setup

- p primo
- Fp cuerpo
- E curva elíptica en Fp
- G punto de curva que genera un subgrupo cíclico
- e pairing

$$SRS = \{G, \tau G, \tau^2 G, ..., \tau^d G\}$$

con tau desconocido

KZG

Prover

Se compromete a un polinomio P

[P] = commit(P) = P(τ) G

(y,π) = open(P, ζ)
donde
y = p(ζ)
π = q(τ)G

q es el polinomio que cumple

$$P(x) - y = (x - \zeta)q(x)$$

manda (y,π)

Verifier

Recibe [P] = P(τ) G

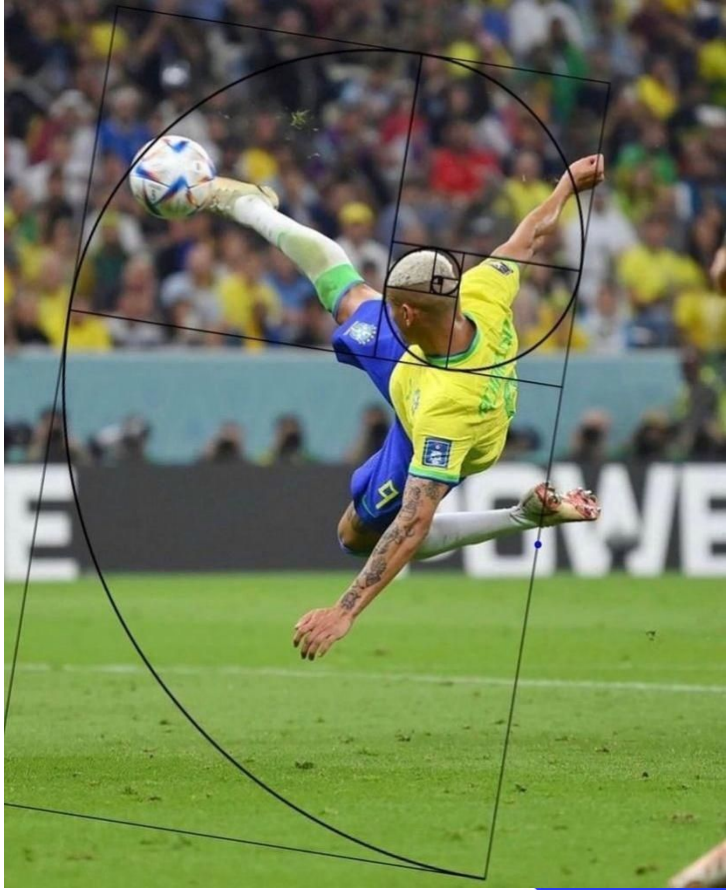
Sortea un challenge ζ al azar y lo manda

Verify([P], π, ζ, y).
Calcula

$$e(P(\tau)G - yG, G)$$
$$e(\tau G - \zeta G, q(\tau)G)$$

y los compara

Fibonacci

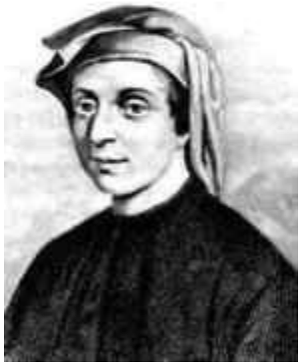


Fibonacci

$$f_0 = 1$$

$$f_1 = 1$$

$$f_{n+2} = f_{n+1} + f_n$$



Setup

- p un primo
- $\mathbb{F}_p$ un cuerpo finito
- $D = \{\omega^0, \omega^1, ..., \omega^{n-1}\}$ subgrupo cíclico.
- H(x,y,z) = x + y - z

Setup KZG

- E Curva elíptica
- G generador de un subgrupo de E
- Trusted Setup
  $SRS = \{G, \tau G, \tau^2 G, ..., \tau^d G\}$

P el prover (Pablo)
V la verifier (Veronica)

W polinomio del Witness

El prover toma la traza y el dominio D e interpola un polinomio W(x) que cumple

$$W(\omega^0) = 1$$
$$W(\omega^1) = 1$$
$$W(\omega^2) = 2$$

$$W(\omega^{n-1}) = f_{n-1}$$

Commitment

El prover le manda un ~~oráculo~~ al verifier.

$$[W] = W(\tau)G$$

Polinomios H y f

$$H(x, y, z) = x + y - z$$

$$f(x) = H(W(x), W(\omega x), W(\omega^2 x))$$

- La traza es válida < = > $f(d) = 0 \quad \forall\, d \in D$
- El prover calcula

$$t(x) = \frac{f(x)}{x^n - 1}$$

- Manda un commitment $[t] = t(\tau)G$

Verifier

El verifier tiene:

- Commitments [W] y [t].

- Toda la info pública.

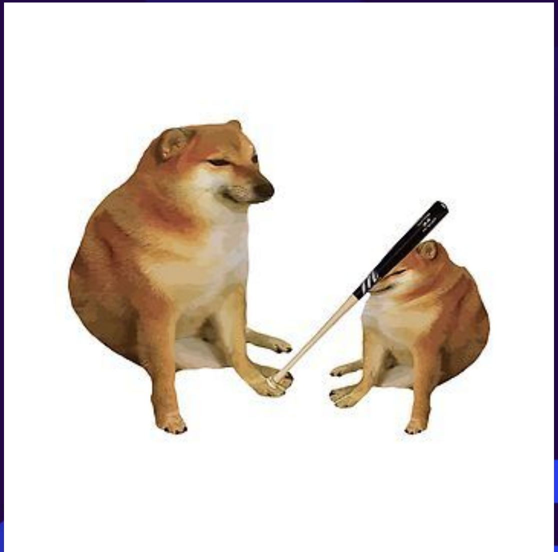
Sortea un elemento ζ al azar en Fp
Le pide al prover los puntos $W(\zeta), W(\omega\zeta), W(\omega^2\zeta), t(\zeta)$

Verifica las evaluaciones siguiendo KZG.

Chequea la igualdad $f(\zeta) = (\zeta^n - 1)t(\zeta)$

eryx

PLONK



XOR à la Plonk

Programa

Q_L	Q_R	Q_M	Q_O	Q_C
1	0	-1	0	0
1	0	-1	0	0
1	0	-1	0	0
1	1	-2	-1	0

Máscara

0	0	-
1	1	-
2	2	-
0	1	2

Ejecución

A	B	C
1	1	-
1	1	-
0	0	-
1	1	0

Setup

- p un primo
- $\mathbb{F}_p$ un cuerpo finito
- $D = \{\omega^0, \omega^1, ..., \omega^{n-1}\}$
  subgrupo cíclico.
- q_L, q_R, q_M, q_O, q_C
- Permutación σ

Setup KZG

- E Curva elíptica
- G generador de un subgrupo de E
- Trusted Setup
  $SRS = \{G, \tau G, \tau^2 G, ..., \tau^d G\}$
- e Pairing

Equation Satisfiability

Programa

Q_L	Q_R	Q_M	Q_O	Q_C
1	0	-1	0	0
1	0	-1	0	0
1	0	-1	0	0
1	1	-2	-1	0

Ejecución

A	B	C
1	1	-
1	1	-
0	0	-
1	1	0

La traza cumple las restricciones de la matriz Q fila a fila?

Equation Satisfiability

Prover

Interpola a(x), b(x), c(x) en D
Manda commits [a], [b], [c]

Calcula

$$f(x) = q_L(x)a(x) + q_R(x)b(x) + q_M(x)a(x)b(x) + q_O(x)c(x) + q_C(x)$$

$$t(x) = \frac{f(x)}{x^n - 1}$$

Manda el commit de [t]

Calcula a(ζ), b(ζ), c(ζ) y t(ζ) y los manda junto a sus pruebas

Verifier

Recibe commits de [a], [b] y [c]

Recibe commits de [t]

Sortea ζ en Fp y lo manda

Verifica las pruebas de a(ζ), b(ζ), c(ζ) y t(ζ)

Verifica

$$f(\zeta) = (\zeta^n - 1)t(\zeta)$$

Wirings

Máscara

0	0	-
1	1	-
2	2	-
0	1	2

Ejecución

A	B	C
1	1	-
1	1	-
0	0	-
1	1	0

La traza cumple las restricciones de la máscara?

Wirings

W la traza aplanada
El prover quiere probar que los siguientes vectores son uno la permutación del otro

$$((0, w_0),\ (1, w_1),\ (2, w_2),\ \ldots,\ (n, w_n))$$

$$((\sigma(0), w_0),\ (\sigma(1), w_1),\ (\sigma(2), w_2),\ \ldots,\ (\sigma(n), w_n))$$

Recibe un challenge β y construye

$$(\beta\omega^0 + w_0, \beta\omega^1 + w_1, \beta\omega^2 + w_2, \ldots, \beta\omega^{n-1} + w_{n-1})$$
$$(\beta\omega^{\sigma(0)} + w_0, \beta\omega^{\sigma(1)} + w_1, \beta\omega^{\sigma(2)} + w_2, \ldots, \beta\omega^{\sigma(n-1)} + w_{n-1})$$

Wirings

Valia que para dos vectores V y W, El vector W es una permutación del vector V con alta probabilidad si: para un α aleatorio, existe un polinomio Z tal que:

1. $Z(\omega^i) \cdot (v_i + \alpha) = Z(\omega^{i+1}) \cdot (w_i + \alpha)$ para todo $i = 0, \ldots, N-1$.
2. $Z(1) \neq 0$.

El prover recibe un challenge α y construye Z y F

F se anula en el dominio < = > Un vector es permutación del otro.

Prover

Interpola W sobre D
Obtiene *w*

Envía un commit [*w*] = w(τ)G

Construye $V_1 = \beta D + W$
$V_2 = \beta\sigma(D) + W$

Construye Z usando α

Calcula

$$t = \frac{Z(X)(v_1+\alpha) - Z(\omega X)(v_2+\alpha)}{X^N - 1}$$

Envía commits [Z] y [t]

Hace el open de Z y t y manda los valores junto a las pruebas

Verifier

Recibe [*w*]

Sortea coeficientes α y β
Envía α y β al Prover

Recibe [*Z*] y [t].
Elige ζ random.
Verifica las pruebas de Z(ζ) y t(ζ)

Calcula

$$\begin{aligned} a &:= Z(\zeta) \\ b &:= v_1(\zeta) + \alpha \\ c &:= Z(\zeta\omega) \\ d &:= v_2(\zeta) + \alpha \\ e &:= t(\zeta)\cdot(\zeta^N - 1) \end{aligned}$$

Verifica ab -cd = e
Verifica Z(1) != 0

eryx

Cositas extra

Que onda el primer commit?

- Para equations satisfiability: [a], [b] y [c]
- Para wirings: Achatamiento [w]
- Esto está bien?
- Necesito relacionar w con a, b y c.

One commit to rule them all

- Proponemos interpolar W en un dominio

  $$D = \{\omega^0, \omega^1, ..., \omega^{3n-1}\}$$

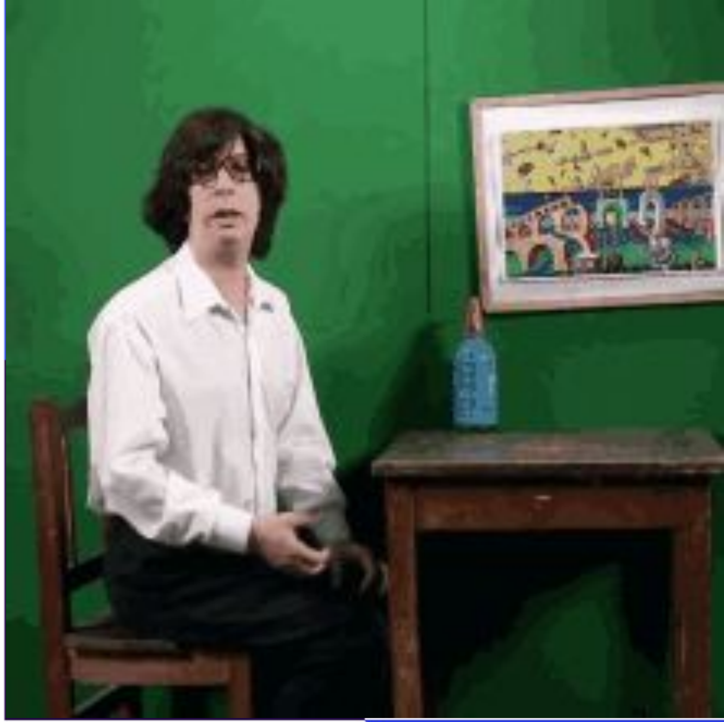
  y hacer un único commit [w]

- Podemos recuperar los polinomios a b y c

$$a(x) = w(x^3)$$
$$b(x) = w(\omega x^3)$$
$$c(x) = w(\omega^2 x^3)$$

¿Y los inputs?

¿Dónde están los inputs?



Preguntas?

Intervalo