



eryx

# zkCity 2024

Día 2: Fibonacci y aritmetización de Plonk

# ¿Qué vamos a ver hoy?

- Algunas preliminares
- Protocolo de prueba y verificación de un programa sencillo (Fibonacci)
- Plonk (versión básica)
- ZK Adventures 2




# 01

## Preliminares

# Polinomios en $\mathbb{F}_p$

$$P = a_0 + a_1x + a_2x^2 + \dots + a_nx^n$$

Grado 

- Evaluar P:  $P(2) = a_0 + a_12 + a_22^2 + \dots + a_n2^n$
- $P(r) = 0 \Rightarrow r$  es una raíz de P.
- Si  $P \neq 0$ , tiene a lo sumo tantas raíces como su grado.
- Si  $r$  es raíz, entonces  $P = (x - r)H(x)$

# Interpolación de Polinomios

Dados puntos  $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$

Existe un único polinomio  $P$  de grado  $n-1$  que cumple

$$P(x_1) = y_1 \quad \dots \quad P(x_n) = y_n$$

- Hay algoritmos para calcular el polinomio interpolador.

# Lema de Schwartz-Zippel

Dados

- $P$  polinomio de  $F_p$
- $e$  un elemento elegido al azar (uniformemente)

Que pasa con  $P(e)$ ?

Si  $P(e) = 0$ , con alta probabilidad  $P=0$

$$\mathbb{P}(P(e) = 0) < \frac{n}{p}$$

Dados  $P$  y  $Q$  polinomios, puedo usar el lema con  $P-Q$

# Oráculos de Polinomios

- Un oráculo es una caja negra que puede resolver problemas en una única operación.
- El oráculo de un polinomio  $P$  es una caja negra que le podemos preguntar cuánto vale  $P$  en un punto  $x$ .
- No conocemos nada sobre  $P$ , ni su grado, ni sus coeficientes.
- Bola de cristal.



# **Protocollo ZK para Fibonacci**



# Fibonacci

$$f_0 = 1$$

$$f_1 = 1$$

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



# Fibonacci

1 1 2 3 5 8 13 21 34 55

1 1 2 4 5 12 17 29 34 55

- A esta tira de elementos le decimos el Witness o la Traza del programa
- Hay trazas válidas y trazas no válidas.

# Setup

- $p$  un primo
- $\mathbb{F}_p$  un cuerpo finito
- $D = \{\omega^0, \omega^1, \dots, \omega^{n-1}\}$  subgrupo cíclico.

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

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

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

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

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

El prover le manda un **oráculo** [W] al verifier.

# Polinomios H y f

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

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

- Que pasa cuando evalúo f en elementos del dominio omega?  
Se anula!

- La traza es válida  $\iff f(d) = 0 \quad \forall d \in D$

- Comentario: H es público.

# Polinomio $f$

- $f$  tiene raíces en todos los puntos del dominio  $D$ .

$$f(x) = (x - 1)(x - \omega)(x - \omega^2) \dots (x - \omega^{n-1})t(x)$$

para algún  $t$

- El prover calcula  $t$  y manda su oráculo al verifier
- Finaliza la parte del prover.
- La prueba consiste de  $[W]$  y  $[t]$

# Verifier

El verifier tiene:

- oráculos  $[W]$  y  $[t]$ .
- Toda la info pública.

Sortea un elemento  $e$  al azar en  $F_p$

le pide a los oráculos los puntos  $W(e), W(\omega e), W(\omega^2 e)$   $t(e)$

Chequea la igualdad

$$f(e) = (e^n - 1)t(e)$$

# Comentarios

- Vimos un esqueleto de un protocolo de prueba ZK
- No se puede probar una traza no válida
- Inputs y output:  
El verifier puede pedir  $W(1)$ ,  $W(\omega)$  o  $W(\omega^n - 1)$



# Ejercicios

- Modelar un bit con restricciones polinomiales
- Hacer un sistema de restricciones que haga un AND.
- Hacer un sistema de restricciones que haga un NOT.
- Hacer un sistema de restricciones que haga un XOR.

# Intervalo

