Plonk: a full proof system

Pairing friendly curve $\mathbb{F}_r, \mathbb{G}_1, \mathbb{G}_2$ gps of order r, pairing

Trusted setup: $g_1 \in \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$ public generators.

~~SRS~~ ~~my Kzg~~ set $s \in \mathbb{F}_r$ is a "shared" random secret (toxic)

A circuit w/ n gates requires $n+5$ elements,

$$1 g_1, s g_1, s^2 g_1, \ldots, s^{n+2} g_1$$

and

$$1 g_2, s g_2$$

- paper (but hard to parse)
- plonk by hand by me/Kobi
- vitalik's understanding plonk

A proof system is really a collection of arguments

proofs ~~that~~ may have been composed -

multiple args

eg "commit + gates sat + permutation"

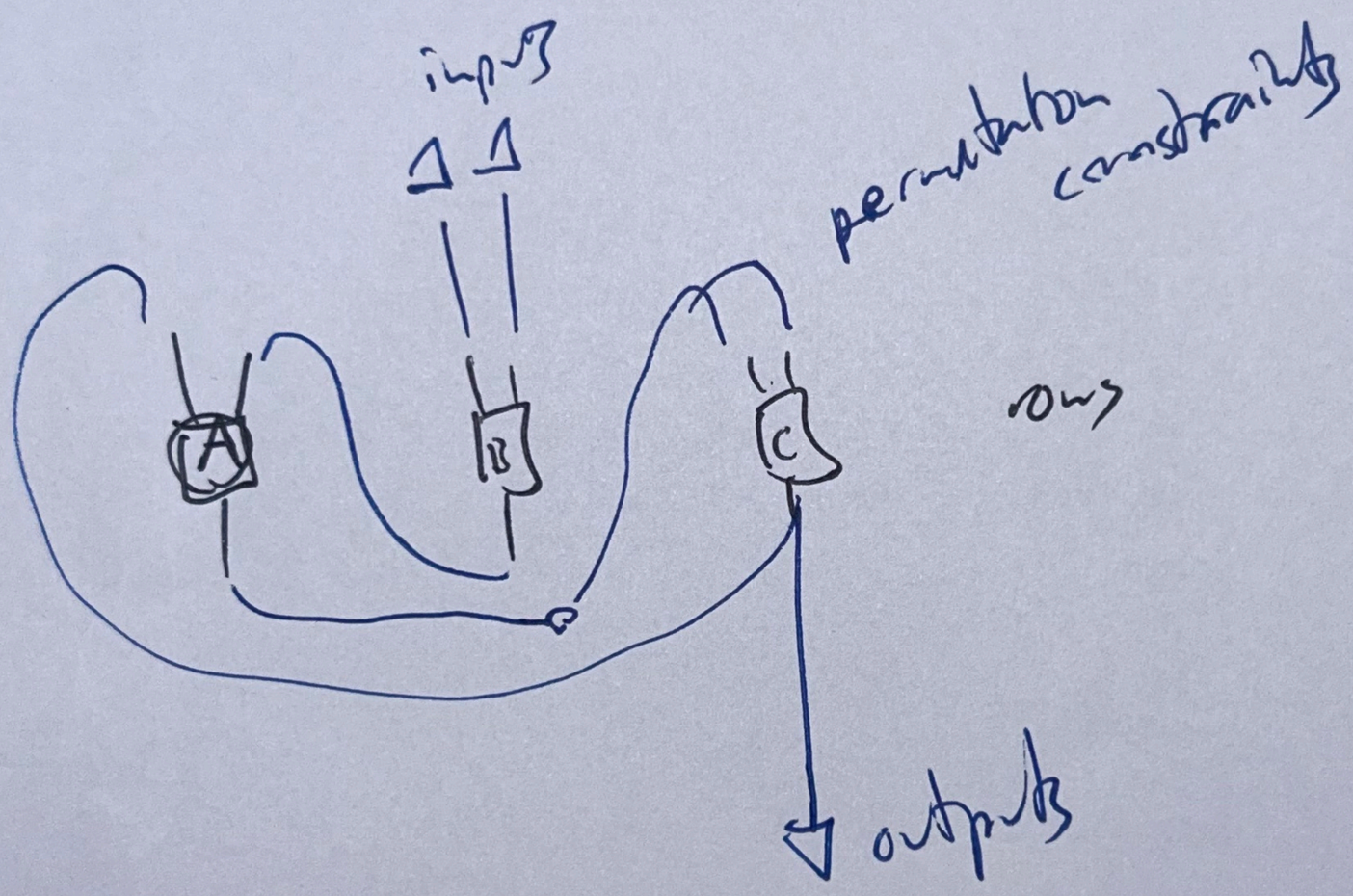
Gates, ~~and~~ circuits, ~~and~~ monoidal categories, complexity (1 1/2)

Fan-in, swap, AND, OR, ..

universal sets of gates (classical logic (Post proof)

Emil Post Letter

Quantum



Plonk Gates

Last time YT discussed RICS and other gates: polynomials satisfied on each line of the matrix.

Gates in a plonkish system can be arbitrary (up to a degree bound) and switched on/off by selectors.

This is visible in the original plonk; it has gates of the form

$$q_L a + q_r b + q_o c + q_M ab + q_c = 0$$

a, b, c are left, right, output wires of the gate.

Note: RICS and Plonk have fixed gates that are used to assemble the circuit. If we allow user-defined gates we say the system supports "custom gates". An interesting golf is to choose a single custom gate that supports a circuit efficiently.

Plonk gates are specialized

by choosing q_L, q_R, q_O, q_M, q_C

on a per-row basis.

These vectors $\vec{q}, \vec{q}, \ldots$ are "preprocessed" and considered constant (fixed at setup).

Q Are they committed to?

The vectors $\vec{a}, \vec{b}, \vec{c}$ are the witness (chosen at proving time)

eg $a+b=c$ set $q_L = q_R = 1$, $q_O = -1$ rest $\emptyset$

$a \cdot b = c$ $q_O = -1$, $q_M = 1$, rest 0

Bind public value $q_R = q_O = q_M = \emptyset$, $q_L = 1$, $q_C =$ value

Example

Prove we know a Pythagorean triple,
ie α, β, γ public such that $\alpha^2+\beta^2=\gamma^2$
we can't directly square, so we do something like this

$x_1 \cdot x_1 = x_2$
$x_3 \cdot x_3 = x_4$
$x_5 \cdot x_5 = x_6$
$x_2 + x_4 = x_6$
$x_1 = \alpha$
$x_3 = \beta$
$x_5 = \gamma$

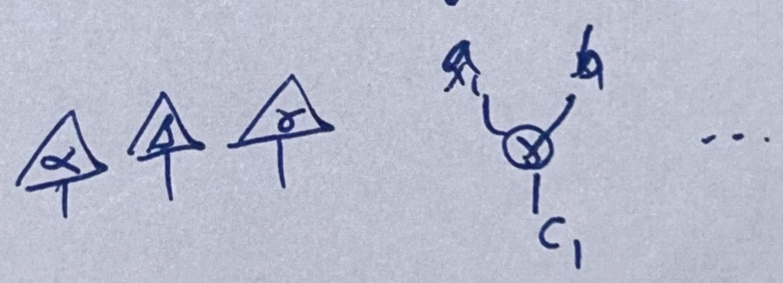
$a_1 \cdot b_1 = c_1$
$a_2 \cdot b_2 = c_2$
$a_3 \cdot b_3 = c_3$
$a_4 + b_4 = c_4$
$a_5 = \alpha$
$a_6 = \beta$
$a_7 = \beta$

as Plonk gate q-settings

$0a_1 + 0b_1 + (-1)c_1 + 1a_1b_1 + 0 = 0$
$0a_2 + 0b_2 + (-1)c_2 + 1a_2b_2 + 0$
$0a_3 + 0b_3 + (-1)c_3 + 1a_3b_3 + 0$
$1 \cdot a_4 + 1 \cdot b_4 + (-1)c_4 + 0 \cdot a_4b_4 + 0$
$-1 \cdot a_5 + 0 \cdot b_4 + (\alpha)a_4 + 0a_4b_4 + \alpha + \beta + \gamma$
$-1\,a_6$
$-1\,a_7$

q_L q_R q_O q_M q_C

This is a "sea of gates"



but they aren't connected,
we introduce new variables
each row

Forgetting copy constraints is a real source of bugs- if your witness generator enforces all the copy equalities, you'll never notice!

Suppose a_1, a_2, a_3, a_4 are witness values in $\mathbb{F}$, and we want to encode them in a polynomial (which we use to commit)

we can label the rows

1	a_1
2	a_2
3	a_3
4	a_4

and set our ~~witness~~ poly $\underline{q}$ such that

$\underline{q}(1) = a_1$
$\underline{q}(2) = a_2$
$\underline{q}(3) = a_3$
$\underline{q}(4) = a_4$

degree-3 polynomial can be interpolated.

As discussed Fri & Monday, if we want to make shifting polynomials easy, and multiply & a multiplicative subgroup is ~~roots~~ 2^kth roots of unity. it is better to use

So our vertical polynomials will be degree $2^k - 1$, rows labelled by the powers of a primitive root of unity

row 1	ω^0
2	ω^1
row 3	ω^2
4	ω^3

$\underline{q}(\omega^0) = a_1$
$\vdots$
$\underline{q}(\omega^{2^k-1}) = a_{2^k-1}$

and interpolation/evaluation is the DFT, V_ω or $\frac{1}{n} V_{\omega^{-1}}$, $O(n \log n)$ $n = 2^k$

Vitalik copy constraints

we need $x_1 \cdot x_1 = x_2$ ⋮ $x_2 + x_4 = 6$

$a_1 = x_1, b_1 = x_1$ ⋮ $a_4 = x_2$

etc.

Vitalik: "coordinate pair accumulator polynomial"

$X, Y \in \mathbb{F}[x]$ interpreted represent x&y coordinates of a set of pts $\in \mathbb{F}^2$.

Let $p \in \mathbb{F}[x]$ represent all the points up to a pos for

$p(0) = 1$

$p(1)$: "first point" represents first pt

$p(2)$ represents second pt

random challenges v_1, v_2, let

$p(0) = 1$

$p(x+1) = p(x) \cdot (\overbrace{v_1 + X(x) + v_2 Y(x)}^{\text{this one}})$ on the domain

$\vdots$

$p(4)$

Claim: If you permute the values of X on the domain, $p(4)$ is unchanged since p is just a product

Prove copy constraints in one column.

two "coordinate accum polys"

P: $X(x) = x$ $\quad Y(x) = a(x)$
identity permut

P': $X'(x)$ $\quad$ swap a_1 & a_3 to enforce $a_1 = a_3$
permutation

P ~~compress~~ $\quad (1, a(\omega^0)), (1, a(\omega^1)), (2, a(\omega^2)), (3, a(\omega^3))$

$\quad (v_1 + X(z) + v_2 Y(z)) \cdots$

P' $\quad 3, a(\omega^1) \qquad\qquad 1, a(\omega^3)$

products are same only if
locks or $a_1 = a_3$

Now for a, b, c

we accumulate points from all three polys.

Stack the vectors

X_a $0, \ldots, n-1$

X_b $n, \ldots 2n-1$

X_c $2n, \ldots, 3n-1$

Then do prove $a_2 = b_4$ w/ $n = 5$

in X' permute those

	X'	
	0	0
	1	1
	2	(2)
	3	3
	4	4
n	5	5
	6	6
	7	7
	8	8
$3n-1$	9	(2)

X'_a, X'_b, X'_c

also called

$\sigma'_a, \sigma_b, \sigma_c$

and then we check the product of the accumulators

$$p_a(n)\ p_b(n)\ p_c(n) = p'_a(n)\ p'_b(n)\ p'_c(n)$$

By accumulating all the pairs, allows for set-crossing permutations

In that discussion we used

the domain 1, 2, 3, ...

rather than $\omega^0, \omega^1, \ldots$

Instead of 3 columns, ~~stacked~~ augmented,

~~stacked~~ and ~~labelled~~

and w/ domain $1, 2, \ldots 3n-1$

labelling entries by $a \in 0, \ldots n-1$, $b \in n, \ldots 2n-1$,

$c \in 2n \ldots$

we can use cosets

$\underline{a}$ $\quad$ $\underline{b}$ $\quad$ $\underline{c}$

$H \quad k_1 H \quad k_2 H$

to 'label' the entries

So how do we enforce the copy constraints?

Coset Idea (Plonk by Hand)

roots of unity are arranged in a multiplicative subgroup H

(eg: in $\mathbb{F}_{17}$, $2^2 | 17-1$ so fourth primitive root of unity exists, 4 is one (a fourth root of unity))

17
34
68

$4^0 = 1 \quad 4^1 = 4 \quad 4^2 = 16 = -1 \quad 4^4 = 1, \quad 4^3 = 64 = -4 = 13$

$H = \{1, 4, 16, 13\}$

$\omega^0, \omega^1, \omega^2, \omega^3$

cosets: if $k_1 \notin H$, $k_1 H$ is a coset (set of the form ah, $h \in H$)

$k_2 \notin H$ or $k_1 H$, $k_2 H$ a coset

a and b iff $a \in bH$ is an equivalence relation by partition the set

~~equivalence class mod k~~

$2H = \{2, 8, 15, 9\}$

$3H = \{3, 12, 14, 5\}$

Then the column locks like

for Montgomery

	row in 1...3n domain	row in H,2H,3H	copy constraint	1...3n sump	w sump		
a_1	1	1	① $a_1 = b_1$	$1 \mapsto 5$	$1 \mapsto 2$	σ_1	interp $\to S_1$
a_2	2	4	② $a_2 = b_2$	$2 \mapsto 6$	$4 \mapsto 8$		
a_3	3	16	③ $a_3 = b_3$	$3 \mapsto 7$	$16 \mapsto 15$		
a_4	4	13	④ $\to$ $a_4 = c_1$	$4 \mapsto 9$	$13 \to 3$		
b_1	5	2	① $b_1 = a_1$	$5 \mapsto 1$	$2 \mapsto 1$	σ_2	interp $\to S_2$
b_2	6	8	② $b_2 = a_2$	$6 \mapsto 2$	$8 \mapsto 4$		
b_3	7	15	③ $b_3 = a_3$	$7 \mapsto 3$	$15 \mapsto 16$		
b_4	8	9	⑤ $b_4 = c_2$	$8 \mapsto 10$	$9 \mapsto 12$		
c_1	9	3	④ $\to$ $c_1 = a_4$	$9 \mapsto 4$	$3 \mapsto 13$	σ_3	interp $\to S_3$
c_2	10	12	⑥ $c_2 = b_4$	$10 \to 8$	$12 \mapsto 9$		
c_3	11	14	⑦ $c_3 = c_4$	$11 \mapsto 12$	$14 \mapsto 5$		
c_4	12	5	⑧ $c_4 = c_3$	$12 \mapsto 11$	$5 \mapsto 14$		

S_i vs accumulator?

we can do the accumulator as follows.

$$\rho''' = \rho'' \cdot \rho \quad \text{or} \quad 1 = \frac{\rho'''}{\rho'' \cdot \rho}$$

Let S_i interp σ_i

get verifier randomness β, γ. Define acc vectors as

$$acc_0 = 1$$

$$acc_i = acc_{i-1} \left(\frac{(a_i + \beta\omega^{i-1} + \gamma)}{(a_i + \beta S_1(\omega^{i-1}) + \gamma)} \frac{(b_i + \beta k_1 \omega^{i-1} + \gamma)}{(b_i + \beta S_2(\omega^{i-1}) + \gamma)} \frac{(c_i + \beta k_2 \omega^{i-1} + \gamma)}{(c_i + \beta S_3(\omega^{i-1}) + \gamma)} \right)$$

many factors cancel

interpolate acc vector to acc poly $acc(x)$, save for later

Putting it together in a 5-round power protocol

Recap: Fiat-Shamir & transcript:

every time a value is fixed (becomes known over the course of the computation) we hash it into a cryptographic sponge construction

→ {} which causes a state transition in the sponge/PRNG

whenever we need randomness, (so not blinding randomness from the verifier)

we "squeeze" the sponge, which emits pseudorandomness & causes a state transition.

"hash into transcript"

If we forget to hash something in, it can cause soundness bugs. These are common, Yi found 3 already in V. talks @ zkplonk.

Round 1 Z_H = poly vanishing on all elts of H = mult. s/g $H = x^{2^k} - 1 = x^n - 1$

Blind & Commit to the witness

generate random $r_1, \ldots, r_6 \in \mathbb{F}$

a, b, c witness vectors

f_a, f_b, f_c poly encodings

define polys

$$a(x) = (r_1 x + r_2) Z_H(x) + f_a(x)$$

$$b(x) = (r_3 x + r_4) Z_H + f_b(x)$$

$$c(x) = (r_5 x + r_6) Z_H + f_c(x)$$

this blinds our commitment to the witness,

Then Now we compute the KZG commitment

$a(s)$, $b(s)$, $c(s)$ using secret s & CRS

Round 2

Commit to the ccm constraints

(in a single poly $p_{all}(w) p_a(w) p_b(w) p_c(w) = p_a'(w) p_b'(w) p_c'(w)$

~~(code~~ draw random $\alpha_1, \ldots \alpha_8 \in \mathbb{F}$

Squeeze challenges $\beta, \gamma \in \mathbb{F}$

Compute

$$z(x) = (\alpha_1 x^2 + \alpha_8 x + \alpha_9) Z_H + \underbrace{acc(x)}_{\text{from earlier}}$$

output commitment $z(s)$

Round 3

Now we compute a big quotient polynomial that combines information

Inputs squeeze a challenge $\alpha \leftarrow \mathbb{F}$

$a(x), b(x), c(x)$ blinded interps

$a(x) = (r_1 x + r_2) Z_H + f_{a(x)}$ interpolation

preprocess/ circuit structure interps

q_R, q_L, q_M, q_O, q_C

$Z_H = (x^k - 1)$,

Permutation interps S_i

$PI(x) = \sum_{i \in [l]} -x_i \cdot L_i(x)$ will appear as $(q_C + PI)$ to bind public inputs

Lagrange basis polys L_i, $L_i(i) = 1$ o/w 0 on $0, 1, \ldots \omega^{2^k - 1}$

$Z(x)$ from blinded accumulator

$$t(x) = \Big(a(x)b(x)q_m(x) + a(x)q_c(x) + b(x)q_R(x) + c(x)q_0(x) + PI(x) + q_c(x) \Big) \frac{1}{Z_H}$$

$$+ \Big((a(x)+\beta x+\gamma)(b(x)+\beta k_1 x+\gamma)(c(x)+\beta k_2 x+\gamma) z(x) \Big] \frac{\alpha}{Z_H(x)}$$

$$- \Big((a(x)+\beta S_1(x)+\gamma)(b(x)+\beta S_2(x)+\gamma)(c(x)+\beta S_3(x)+\gamma) z(x\omega) \Big)$$

$$\therefore \frac{\alpha}{Z_H(x)}$$

$$+ \big(z(x) - 1 \big) L_1(x) \frac{\alpha^2}{Z_H(x)}$$

~~Degree~~ t_{hi} has degree $3n+5$, OOR for SRS,

split to 3 $n+2$ deg poly t_{low}, t_{mid}, t_{hi}

$$f(x) = t_{low}(x) + x^{n+2} t_{mid}(x) + x^{2n+4} t_{hi}(x)$$

output $t_{low}(s)$, $t_{mid}(s)$, $t_{hi}(s)$

Round 4

Grab a new challenge δ, etc

compute opening evals

$$a(\delta),\ b(\delta),\ c(\delta)$$
$$\bar{a} \qquad \bar{b} \qquad \bar{c}$$

$$\bar{s}_1 = s_1(\delta),\ \bar{s}_2 = s_2(\delta)$$

$$\bar{t} = t(\delta)$$

$$\bar{z}_\omega = z(\delta\omega)$$

and linearization poly

$$r(x) = \bar{a}\bar{b}\, q_M(x) + \bar{a}\, q_L(x) + \bar{b}\, q_R(x) + \bar{c}\, q_O(x) + q_C(x)$$

$$+ \Big((\bar{a} + \beta\delta + \gamma)(\bar{b} + \beta k_1 \delta + \gamma)(\bar{c} + \beta k_2 \delta + \gamma)\, z(x) \Big)\alpha$$

$$- \Big((\bar{a} + \beta\bar{s}_1 + \gamma)(\bar{b} + \beta\bar{s}_2 + \gamma)\beta\bar{z}_\omega\, s_3(x) \Big)\alpha$$

$$+ z(x)\, l_1(\delta)\, \alpha^2 +$$

and $\bar{r} = r(\delta)$

output $(\bar{a}, \bar{b}, \bar{c}, \bar{s}_1, \bar{s}_2, \bar{z}_\omega, \bar{t}, \bar{r})$

Round 5

Some more summarization and commitment.

- Grab another challenge ν and compute $W_\delta(x)$

$$W_\delta(x) = \frac{1}{x-\delta}\Big(t_{low}(x) + \delta^{n+2} t_{mid}(x) + \delta^{2n+4} t_{hi}(x) - E + \nu(r(x)-\bar{r}) + \nu^2(a(x)-\bar{a}) + \nu^3(b(x)-\bar{b}) + \nu^4(c(x)-\bar{c}) + \nu^5(\zeta_1(x)-\bar{\zeta}_1) + \nu^6(\zeta_2(x)-\bar{\zeta}_2)\Big)$$

and

$$W_{\delta\omega}(x) = \frac{z(x)-\bar{z}_\omega}{x-\delta\omega}$$

output commits $[W_\delta]$, $[W_{\delta\omega}]$

$W_\delta(s)$ $W_{\delta\omega}(s)$

So we get the proof

$$\pi_{SO} = \Big(\underbrace{[a], [b], [c], [z], [t_{low}], [t_{mid}], [t_{hi}],}_{\text{group elts}}$$

$$\underbrace{[W_\sigma], [W_{\sigma\omega}], \bar{a}, \bar{b}, \bar{c}, \bar{\xi}, \bar{\xi}_0, \bar{r}, \bar{z}_\omega}_{\text{field elts}} \Big)$$

send π

and check it with a pairing

Verification

Circuit structure polynomials

$q_M, q_L, q_R, q_O, q_C, S_1, S_2, S_3$

Verifier can eval at s using SRS to get commitments
These, w/ the S_2 point and the nth root of unity are the VK
Verifier gets a random pt u.

11-step process

1) Check content points received are in G_1
2) Check field elts received are in F
3) The public inputs $\{w_i\}_{i \in [\ell]}$ are field elts
4) compute challenges from common inputs, public input, elts of π as per
5) compute $Z_H(\zeta) = Z_H(r) = \zeta^n - 1$
6) compute $L_1(\zeta) = \frac{\zeta^n - 1}{n(\zeta - 1)}$
7) compute P.I. poly

$$PI(\zeta) = \sum_{i \in \ell} w_i L_i(\zeta)$$

8) split r into constant terms

r_0 $r'(x)$

9 & 10) ~~compute~~ compute batched poly commitment

11) group-encoded batch evaluate

12) Batch validate all evaluations w/ pairing

Lookup (idea)

Halo2 simplified version for book

(1) Subset

table w/ 2^k rows $0, \dots, 2^k$

columns A, S

subset argument: every cell in A is equal to some cell in S (not a permutation, values can repeat, not surjective)

both can have duplicates,

~~Sfixed(k)~~

Think: A is a advice witness column

S is fixed at circuit creation ✓, ✓

or

also a witness (dynamic lookup)

(Lookup selector: replace A_i w/ S_0 if not selected)

have not this bug!

L_i Lagrange basis, $L_i(\omega^i) = 1$

0 o/w on domain

Now let prover supply permutations
of each column A and S, A' & S'.

augmenting with an accumulator column Z, require
w/ challenges β, γ

$$Z(\omega X)\left(A'(X)+\beta\right)\left(S'(X)+\gamma\right)$$
$$-Z(X)\left(A(X)+\beta\right)\left(S(X)+\gamma\right) = 0$$
$$\text{and } L_0(X)\left(1-Z(X)\right) = 0$$

ie writing $a - b \neq 0$ as $\frac{a}{b} \neq 1$ &accum,

$$Z_{i+1} = Z_i \frac{\left(A_i+\beta\right)\left(S_i+\gamma\right)}{\left(A'_i+\beta\right)\left(S'_i+\gamma\right)}$$

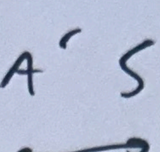
$$Z_{2^k} = Z_0 = 1$$

The prover arranges A' and S' such that

- like-valued cells of A' are vertically adjacent, and
- the first row of like values in A' has
the corresponding value in S'
of/in S' arbitrary permutation of S

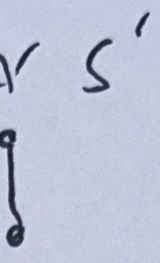
enforce either

$A_i' = S_i'$ match



or

$A_i' = A_{i-1}'$ match



using

$$(A'(x) - S'(x)) \cdot (A'(x) - A'(\omega^{-1} x)) = 0$$

and in addition, initial condition

$A_0' = S_0'$ using

$$\mathcal{L}_0(x) \cdot (A'(x) - S'(x)) = 0$$

the $A'(x) - A'(\omega^{-1}x)$ term of the first rule has the effect though $\omega^{-1}X$ wraps ℓ of second rule,

inductive forces every elt $\geq A'$ to equal at least one elt in S'.

Zero knowledge –
Hiding rows in cols.....

for multiple colums $A, S,$ (A', S' skip single cols)
combine cols using a random challenge

colums $f_1, \ldots, f_w \in \mathbb{F}[x]$, table $t^* \in (\mathbb{F}^w)^d$

check each row j $(f_1(g^j), \ldots f_w(g^j)) \in t^*$

~~verifier~~ include in ?vk? (preprocessed poly's)
$t_i \in \mathbb{F}[x]$ st $t_i(g^j) = t^*_{ij}$ $\forall j = 1, \ldots d$

verifier chooses rand $\alpha \in \mathbb{F}$

set $t := \sum_{i \in [w]} \alpha^i t_i$ and $f := \sum_{i \in [w]} \alpha^i f_i$

if a row not in table,
~~check~~ likely to not $f(g^j) \notin t$