Pairing Based zkSNARKs

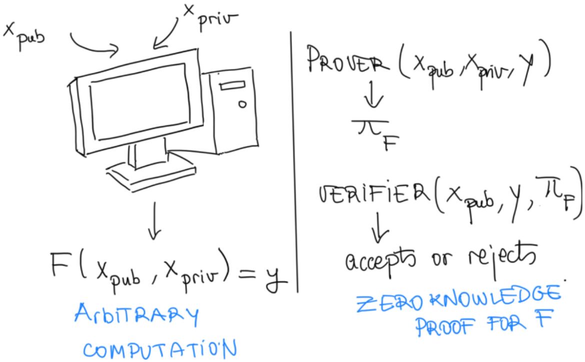
Carla Ràfols

September 2024

Universitat Pompeu Fabra Barcelona

(zk)-SNARKs

(zk)-SNARKs



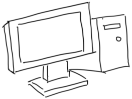
- We think of "practical" proofs as proofs of computational integrity;
- ZKPs reveal nothing about private inputs of the computation;
- (zk)SNARKs (**(zk-)Succinct Non-Interactive Arguments of Knowledge**) are **short** proofs, usually independent of computation size

$$|\pi_F| < |F|$$

How are many SNARKs built?

- **FRONTEND**

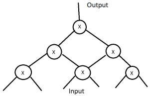
Computation



$\longrightarrow$

Computation Representation

e.g. Arith. Circuit, Arith. Circuit with Lookups



program

model with restricted operations

$\longrightarrow$

Algebraic Relations

R1CS, Plonkish, CCS

$e.g. \mathbf{A}, \mathbf{B}, \mathbf{C}$ s.t.

$\vec{z}$ satisfies circuit iff

$\mathbf{A}\vec{z} \circ \mathbf{B}\vec{z} = \mathbf{C}\vec{z}$

$\rightarrow$

Polynomial Relations

Univ or Multiv.

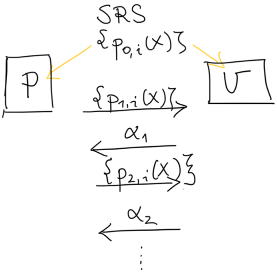
e.g.

$t(X)|A(X)B(X) - C(X)$

How are many SNARKs built?

- **BACKEND**

(Preprocessing) Polynomial IOP



$\longrightarrow$
Polynomial commitment
$+$
Fiat Shamir

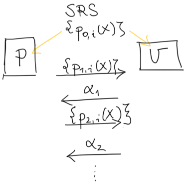
SNARK

SRS or CRS, π

How are many SNARKs built?

- **BACKEND**

(Preprocessing) Polynomial IOP



$\longrightarrow$

Polynomial commitment

$+$

Fiat Shamir

Compressing Step
Cryptography
Comp. Security

SNARK

SRS or CRS, π

- **Key Idea:**: Checking Polynomial Identities at Random Points (or in an elliptic curve)
  Can be done succinctly with Polynomial Commitments.
- **ZK** comes almost for free.

SNARKs for Proving Large Computations

- **FRONTEND**

LARGE **Computation** → LARGER **Computation Representation** → LONG VECTORS **Algebraic Relations** → LARGE DEGREE **Polynomial Relations**

- **BACKEND**

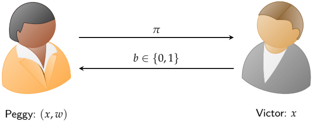
→ **PIOP** → **SNARK** A LOT OF CRYPTOGRAPHIC WORK for PROVER

Example of Practical Parameters:

- C circuit with 2^{20} multiplication gates over finite field of 255 bits;

What is a "good" SNARK

Performance measured in different parameters.



- Prover complexity/ Verifier complexity.
- Proof size
- Transparent Setup/Structured Reference String.
- Private vs Public Verification...
- Weaker/ Stronger Computational assumptions.

This talk:

- $O(n \log n)$ prover, $O(1)$ proof size, $O(1)/O(\log n)$ verification (preprocessing univariate PIOP, KZG Polynomial Commitment in pairing groups)

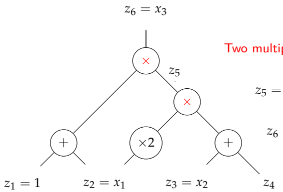
But recently many SNARKs,

- $O(n)$ prover, $O(\log n)$ proof size, $O(\log n)$ verification (preprocessing multivariate PIOP, sumcheck protocol)

Example: From Circuits to Algebraic Relations

Rank 1 Constraint Systems

Statement: $C(1, x_1, x_2, w) = x_3$ for some w, $\vec{x}$ public inputs.



Two multiplication gates g_1, g_2

$z_5 = (2z_2)(z_3 + z_4)$

$z_6 = (1 + z_2)z_5$

$$\mathbf{A}\vec{z} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ z_2 \\ z_3 \\ z_4 \\ z_5 \\ z_6 \end{pmatrix} = \begin{pmatrix} 1 \\ z_2 \\ z_3 \\ z_4 \\ 2z_2 \\ 1 + z_2 \end{pmatrix} \quad \mathbf{B}\vec{z} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ z_2 \\ z_3 \\ z_4 \\ z_5 \\ z_6 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ z_3 + z_4 \\ z_5 \end{pmatrix} \quad \mathbf{C}\vec{z} = \begin{pmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \\ z_5 \\ z_6 \end{pmatrix}$$

Statement true $\Longleftrightarrow$

$$\mathbf{A}\vec{z} \circ \mathbf{B}\vec{z} = \mathbf{C}\vec{z}, \text{ and } \{z_1 = 1, z_2 = x_1, z_3 = x_2, z_6 = x_3\}$$

From Circuit to Algebraic Relations, Takeaway

Statement: $C(1, x_1, x_2, w) = x_3$ for some w, $\vec{x}$ public inputs.

1. **Public Input Relations:**
   $\{z_1 = 1, z_2 = x_1, z_3 = x_2, z_6 = x_3\}$
2. **Hadamard Product Relation**:
   $\vec{a} \circ \vec{b} = \vec{c}$
3. **Linear Relations**:
   $\vec{a} = \mathbf{A}\vec{z}$, $\vec{b} = \mathbf{B}\vec{z}$, $\vec{c} = \mathbf{C}\vec{z}$.

- Matrices are public, part of the circuit description.
- They are sparse, but of dimension of the extended witness size (inputs + multiplicative gates).

From Algebraic Relations to Univariate Polynomials

Inner Product Relations and the Univariate Sumcheck

- $\mathcal{R} = \{r_0, \ldots, r_{n-1}\} \subset \mathbb{F}_p^*$, multiplicative subgroup

$$\lambda_i(X) = \prod_{j \neq i} \frac{(X - r_j)}{(r_i - r_j)}, \qquad t(X) = \prod_j (X - r_j).$$

Algebraic Formulation	Polynomial Formulation
Vector $\vec{y} = (y_0, \ldots, y_{n-1})$	Poly. $y(X) = \sum_{i=0}^{n-1} y_i \lambda_i(X) = \vec{\lambda}(X)^\top \vec{y}$
Public Input: $\vec{z}, \vec{x}$ agree on l positions	$z(X) - x(X)$ is divisible by $t_l(X)$
Hadamard Product $\vec{a} \circ \vec{b} = \vec{c}$	$a(X)b(X) - c(X)$ is divisible by $t(X)$
Inner product $\sigma = \vec{f} \cdot \vec{g}$	[Ben-Sasson et al. 18] $\exists R(X)$, $deg\ R(X) \leq n - 2$. $t(X)$ divides $f(X)g(X) - n^{-1}\sigma - XR(X)$

From Algebraic Relations to Univariate Polynomials

Inner Product Relations and the Univariate Sumcheck

- $\mathcal{R} = \{r_0, \ldots, r_{n-1}\} \subset \mathbb{F}_p^*$, multiplicative subgroup

$$\lambda_i(X) = \prod_{j \neq i} \frac{(X - r_j)}{(r_i - r_j)}, \qquad t(X) = \prod_j (X - r_j).$$

Algebraic Formulation	Polynomial Formulation
Vector $\vec{y} = (y_0, \ldots, y_{n-1})$	Poly. $y(X) = \sum_{i=0}^{n-1} y_i \lambda_i(X) = \vec{\lambda}(X)^\top \vec{y}$
Public Input: $\vec{z}, \vec{x}$ agree on l positions	$z(X) - x(X)$ is divisible by $t_l(X)$
Hadamard Product $\vec{a} \circ \vec{b} = \vec{c}$	$a(X)b(X) - c(X)$ is divisible by $t(X)$
Inner product $\sigma = \vec{f} \cdot \vec{g}$	[Ben-Sasson et al. 18] $\exists R(X)$, $deg\ R(X) \leq n - 2$. $t(X)$ divides $f(X)g(X) - n^{-1}\sigma - XR(X)$

We can immediately build a non-interactive IOP for any of these relations.

Example Hadamard Product Relation

PIOP:

P $\xrightarrow{a(X),\, b(X),\, c(X),\, h(X)}$ V

$$a(X) \cdot b(X) - c(X) \stackrel{?}{=} h(X) \cdot t(X)$$

Example Hadamard Product Relation

PIOP:

P $\xrightarrow{a(X),\, b(X),\, c(X),\, h(X)}$ V

$$a(X)\cdot b(X) - c(X) \stackrel{?}{=} h(X)\cdot t(X)$$

Proof System:

"Compiled Protocol" ①

P $\xrightarrow{a(\tau)P_1,\, b(\tau)P_2,\, c(\tau)P_1,\, h(\tau)P_2}$ V

$$e(a(\tau)P_1, b(\tau)P_2) - e(c(\tau)P_1, P_2) \stackrel{?}{=} e(h(\tau)P_1, t(\tau)P_2)$$

"Compiled Protocol" ②

P $\xrightarrow{a(\tau)P_1,\, b(\tau)P_1,\, c(\tau)P_1,\, h(\tau)P_1}$ V

$\xleftarrow{s}$ $s \leftarrow \mathbb{Z}_p$

$\xrightarrow{\pi_{KZG},\, a(s),\, b(s),\, c(s),\, h(s)}$

π_{KZG} verifies $\wedge$ $a(s)b(s) - c(s) = h(s)t(s)$

SRS: $(P_1, \tau P_1, \tau^2 P_1, \ldots, \tau^{n-1}P_1),\ (P_2, \tau P_2, \tau^2 P_2, \ldots, \tau^{n-1}P_2)$

How to prove Many Linear Relations?

SNARKs with Constant Proof Size

- **Statement:** $\vec{y} = \mathbf{M}\vec{z}$.
- No efficient extension of the univariate sumcheck to prove many inner product relations.

Groth16, ...	**Plonk,...** Permutation-based arguments $\mathbf{M}$ is a permutation	**Marlin** Reduce many to one relation and use inner product
QA-NIZK Arguments	$\vec{y} = \mathbf{M}\vec{z}$ iff $\prod(X + y_i) = \prod(X + z_i)$.	$\vec{y} = \mathbf{M}\vec{z} \Longrightarrow r^\top \cdot \vec{y} = (\vec{r}^\top \mathbf{M})\vec{z}$, $\vec{r}$ sufficiently random

QA-NIZK Arguments
Groth16

Motivation

- To prove R1CS, we need to prove the linear relations:

$$\begin{pmatrix} \vec{a} \\ \vec{b} \\ \vec{c} \end{pmatrix} = \begin{pmatrix} \mathbf{A} \\ \mathbf{B} \\ \mathbf{C} \end{pmatrix} \vec{z} \iff$$

$$\begin{pmatrix} a(X) = \vec{\lambda}(X)^\top \vec{a} \\ b(X) = \vec{\lambda}(X)^\top \vec{b} \\ c(X) = \vec{\lambda}(X)^\top \vec{c} \end{pmatrix} = \begin{pmatrix} \lambda(X)^\top \mathbf{A} \\ \lambda(X)^\top \mathbf{B} \\ \vec{\lambda}(X)^\top \mathbf{C} \end{pmatrix} \vec{z} = \begin{pmatrix} u_1(X) & \dots & u_m(X) \\ v_1(X) & \dots & v_m(X) \\ w_1(X) & \dots & w_m(X) \end{pmatrix} \vec{z}$$

In “Compiled”Protocol we need to prove:

$$\begin{pmatrix} a(\tau)\mathcal{P} \\ b(\tau)\mathcal{P} \\ c(\tau)\mathcal{P} \end{pmatrix} = \begin{pmatrix} u_1(\tau)\mathcal{P} & \dots & u_m(\tau)\mathcal{P} \\ v_1(\tau)\mathcal{P} & \dots & v_m(\tau)\mathcal{P} \\ w_1(\tau)\mathcal{P} & \dots & w_m(\tau)\mathcal{P} \end{pmatrix} \vec{z}$$

"Membership" of vector of $\mathbb{G}^3$ in column space of matrix $3 \times |m|$.

Hash Proof System [CraSho02]

Notation: $[a] := a\mathcal{P}$.

$$\sigma = (gk, [\mathbf{M}], [\vec{k}^\top \mathbf{M}])$$

P → V

$$\boldsymbol{\pi} = [\vec{k}^\top \mathbf{M}]\vec{w}$$

V: $\vec{k} \leftarrow \mathcal{K}$

$[\vec{y}] : \vec{y} = \mathbf{M}\vec{w}$

$\boldsymbol{\pi} \stackrel{?}{=} \vec{k}^\top [\vec{y}]$

- **Example:**

$$[\mathbf{M}] = \begin{pmatrix} \mathcal{P} \\ H \end{pmatrix} \qquad \text{Statement: } [\vec{y}] = \begin{pmatrix} [y]_1 \\ [y]_2 \end{pmatrix} = [\mathbf{M}]w = \begin{pmatrix} w\mathcal{P} \\ wH \end{pmatrix}$$

$$[\vec{k}^\top \mathbf{M}] = k_1 P + k_2 H \qquad \boldsymbol{\pi} = w(k_1 P + k_2 H)$$

Hash Proof System [CraSho02]

Notation: $[a] := a\mathcal{P}$.

$$\sigma = (gk, [\mathbf{M}], [\vec{k}^\top \mathbf{M}])$$

P $\longrightarrow$ V

$$\boldsymbol{\pi} = [\vec{k}^\top \mathbf{M}]\vec{w}$$

V: $\vec{k} \leftarrow \mathcal{K}$

P: $[\vec{y}] : \vec{y} = \mathbf{M}\vec{w}$

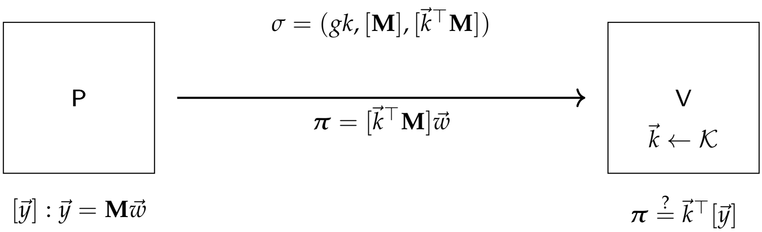
V: $\boldsymbol{\pi} \stackrel{?}{=} \vec{k}^\top [\vec{y}]$

- **Completeness:**

$$[\vec{k}^\top \mathbf{M}]\vec{w} = \vec{k}^\top [\mathbf{M}\vec{w}] = \vec{k}^\top \vec{y}.$$

Hash Proof System [CraSho02]

Notation: $[a] := a\mathcal{P}$.



- **Soundness:**

  If $\vec{y} \notin \mathrm{Im}(\mathbf{M})$, $\vec{k}^\top [\vec{y}]$ information theoretically hidden!

  If designated verifier key is leaked, no soundnes!

QA-NIZK for Linear Spaces [LibPetJoyYun14,KiWeel15]

$$(gk, [\mathbf{M}], [\mathbf{K}^\top \mathbf{M}], [\mathbf{A}], [\mathbf{A}\mathbf{K}^\top])$$

P → V

$$\pi = [\mathbf{K}^\top \mathbf{M}]\vec{w}$$

$[\vec{y}] : \vec{y} = \mathbf{M}\vec{w}$

$$e([\mathbf{A}], \pi) \stackrel{?}{=} e([\mathbf{A}\mathbf{K}^\top], [\vec{y}])$$

- **Completeness, Zero-Knowledge**: Unchanged.
- **Soundness**: Computational: unless the prover knows $\vec{w}$ s.t $[\vec{y}] = [\mathbf{M}]\vec{w}$, it cannot compute $[\pi]$.

QA-NIZK Proof for Linear Spaces for R1CS[1]

- To prove R1CS, the "Compiled"Protocol needs to prove:

$$\begin{pmatrix} a(\tau)\mathcal{P} \\ b(\tau)\mathcal{P} \\ c(\tau)\mathcal{P} \end{pmatrix} = \begin{pmatrix} u_1(\tau)\mathcal{P} & \dots & u_m(\tau)\mathcal{P} \\ v_1(\tau)\mathcal{P} & \dots & v_m(\tau)\mathcal{P} \\ w_1(\tau)\mathcal{P} & \dots & w_m(\tau)\mathcal{P} \end{pmatrix} \vec{z},$$

i.e. "Membership" in column space of matrix $3 \times |m|$.

- The SRS needs to include, among others:

$$[\mathbf{K}^\top \mathbf{M}] = (\frac{\beta}{\delta}, \frac{\alpha}{\delta}, \frac{1}{\delta}) \begin{pmatrix} u_1(\tau)\mathcal{P} & \dots & u_m(\tau)\mathcal{P} \\ v_1(\tau)\mathcal{P} & \dots & v_m(\tau)\mathcal{P} \\ w_1(\tau)\mathcal{P} & \dots & w_m(\tau)\mathcal{P} \end{pmatrix} = (\frac{\beta u_j(\tau) + \alpha v_j(\tau) + w_j(\tau)}{\delta})_{j=1}^m.$$

- Security relies crucially on the fact that it is impossible to calculate $\mathbf{K}^\top \vec{y}$ if does not have a witness for $[\vec{y}] \in Col(\mathbf{M}) \implies$ New key $\mathbf{K}$ for every circuit!!

[1] Our aim here is to present all techniques in the literature in a unified way, not an attribution of these techniques to the QA-NIZK literature.

Groth16

- Combination of Hadamard Argument + QANIZK (in asymmetric bilinear groups) super compressed, using full power of unfalsifiable assumptions;
- SRS is:

$$\alpha, \beta, \delta, \{\tau^i\}_{i=0}^{n-1}, \{u_j(\tau)\beta + v_j(\tau)\alpha + w_j(\tau)\}_{j=0}^{l}$$

$$\left\{\frac{u_j(\tau)\beta + v_j(\tau)\alpha + w_j(\tau)}{\delta}\right\}_{j=l+1}^{m}, \{x^i t(\tau)/\delta\}_{i=0}^{n-2},$$

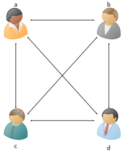
- Prover cost: a few multiexponentiations of size $O(|m.gates|)$, 7 FFT of size $|m.gates|$ ($O(|m.gates| \log |m.gates|)$ field operations).
- Proof size 3 group elements, super efficient verification 3 pairings (independent of circuit size!)
- Trusted Setup is inherently circuit dependent.

Trusted Setups

Z. Wilcox (ZCash) on his knees destroying a computer after parameter generation.

- SNARKs require a trusted party to generate the parameters.
- Knowledge of randomness to generate parameters: complete failure.
- Solution: distribute trust in a **Setup Ceremony**.
- Costly and complicated process.

SNARKs: Improving Parameter Generation [GroKohMalMeiMie18]



Multiparty Computation Model

Updatable Model

- Updatable Model: for soundness it suffices that one party is honest, and CRS can always be updated NI.
- In [BowGabMie17]: after a trusted and updatable setup phase to generate $(\tau\mathcal{P}_i, \tau^2\mathcal{P}_i, \ldots, \tau^q\mathcal{P}_i)$, $i = 1, 2$, circuit dependent setup of Groth16 is updatable.
- Universal and Updatable SNARKs: after a trusted and updatable setup phase to generate $(\tau\mathcal{P}_i, \tau^2\mathcal{P}_i, \ldots, \tau^q\mathcal{P}_i)$, $i = 1, 2$, a circuit dependent SRS that preprocesses the circuit is derived.

Marlin

Marlin: How to Prove Many Inner Product Relations

- **Problem 1.** No efficient extension of the univariate sumcheck to prove m inner product relations.
- **Solution 1.** Prove one *sufficiently random relation:*

  Checking if $\vec{y} = \mathbf{M}\vec{z}$ $\quad vs \quad$ Checking if $\vec{r}^\top \vec{y} = (\vec{r}^\top \mathbf{M}) \cdot \vec{z}$, where $\vec{r}$ is sufficiently random, chosen by verifier!!

- **Problem 2** Although matrix $\mathbf{M}$ is public, a sublinear verifier cannot afford to sample a random vector in rowspace of $\mathbf{M}$ (since number of rows $= O(|C|)$)
- **Solution 2:** Prover needs to show that $\vec{r}^\top \mathbf{M}$ is correct.

From Algebraic Relations to Polynomials

Reducing Many to One Relations

Given $\mathbf{M} \in \mathbb{F}^{n\times n}$, define the bivariate polynomial:

$$P(X,Y) = (\lambda_0(Y),\ldots,\lambda_{n-1}(Y))\ \mathbf{M} \begin{pmatrix} \lambda_0(X) \\ \vdots \\ \lambda_{n-1}(X) \end{pmatrix} = \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} m_{ij}\lambda_i(Y)\lambda_j(X)$$

- Given random x, the vector

  $$\vec{d} = (\lambda_0(x),\ldots,\lambda_{n-1}(x))\ \mathbf{M}$$

  is a sufficiently random vector in the row span of $\mathbf{M}$.

- The partial evaluation

  $$D(X) = P(X,x) = \sum_{i=0}^{n-1} d_i\lambda_i(X) = (\lambda_0(x),\ldots,\lambda_{n-1}(x))\ \mathbf{M} \begin{pmatrix} \lambda_0(X) \\ \vdots \\ \lambda_{m-1}(X) \end{pmatrix}$$

  is a polynomial encoding of $\vec{d}$ in the Lagrange basis.

- The partial evaluation

$$D(X) = P(X, x) = \sum_{i=0}^{n-1} d_i \lambda_i(X) = (\lambda_0(x), \ldots, \lambda_{n-1}(x)) \; \mathbf{M} \begin{pmatrix} \lambda_0(X) \\ \vdots \\ \lambda_{n-1}(X) \end{pmatrix}$$

  is a polynomial encoding of $\vec{d}$ in the Lagrange basis.
- The prover needs to show that $D(X)$ is correct.
- Preprocessing $\mathbf{M}$ naively does not work, would mean quadratic SRS.
- Idea: in the preprocessing phase, polynomials $\mathsf{col}, \mathsf{row} : K \longrightarrow \{0, \ldots, n-1\}$ and $\mathsf{val} : K \longrightarrow \mathbb{Z}_p$ are defined such that:

$$D(X) = \sum_{k \in K} \mathsf{val}(k) \lambda_{\mathsf{row}(k)}(x) \lambda_{\mathsf{col}(k)}(X)$$

  where K is the number of non-zero entries of $\mathbf{M}$.

Summary

How to prove Many Linear Relations?

- **Statement:** $\vec{y} = \mathbf{M}\vec{z}$.
- $\tilde{O}(n) = O(n \log_2 n)$, quasi linear

Groth16, ...

Trusted setup for each $\mathbf{A}, \mathbf{B}, \mathbf{C}$

Not universal!

Prover: $\tilde{O}(|m.gates|)$

Plonk,...

Permutation-based arguments

$\mathbf{M}$ is a permutation

$\vec{y} = \mathbf{M}\vec{z}$ iff

$$\prod(X + y_i) = \prod(X + z_i).$$

Prover: $\tilde{O}(|total\ gates|)$

Spartan, Marlin

Reduce many to one relation and use inner product

$$\vec{y} = \mathbf{M}\vec{z} \implies (\vec{r}^\top \mathbf{M})\vec{z} = \vec{r}^\top \vec{z},$$

$\vec{r}$ sufficiently random

Prover: $\tilde{O}(|sparsity\ matrix|)$

Summary

How to prove Many Linear Relations?

- **Statement:** $\vec{y} = \mathbf{M}\vec{z}$.
- $\tilde{O}(n) = O(n \log_2 n)$, quasi linear

Groth16, ...

Trusted setup for each $\mathbf{A}, \mathbf{B}, \mathbf{C}$

Not universal!

Prover: $\tilde{O}(|m.gates|)$

Plonk,...

Permutation-based arguments

$\mathbf{M}$ is a permutation

$\vec{y} = \mathbf{M}\vec{z}$ iff

$$\prod(X + y_i) = \prod(X + z_i).$$

Prover: $\tilde{O}(|total\ gates|)$

Spartan, Marlin

Reduce many to one relation and use inner product

$\vec{y} = \mathbf{M}\vec{z} \implies (\vec{r}^\top \mathbf{M})\vec{z} = \vec{r}^\top \vec{z}$,

$\vec{r}$ sufficiently random

Prover: $\tilde{O}(|sparsity\ matrix|)$

Conclusion: Choice of technique to prove linear constraints determines much of the characteristics of proof system:

- Plonk, Marlin need randomized checks, thus random oracles.
- Groth16 does not need ROs but circuit dependent setup;
- Plonk changes arithmetization so that checking permutations is enough;
- Different prover perfomance for each technique, free additive gates in Groth16.