Techniques in Universal and Updatable SNARKs

Carla Ràfols

ISC Winter School 2023 - March 1st

Overview

What are ZK Proofs?

	2		5		1		9	
8			2		3			6
	3			6			7	
		1				6		
5	4						1	9
		2				7		
	9			3			8	
2			8		4			7
	1		9		7		6	

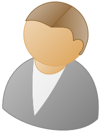
x="Unsolved Sudoku"

4	2	*6*	5	*7*	1	*3*	9	*8*
8	*5*	*7*	2	*9*	3	*1*	*4*	6
1	3	*9*	*4*	6	*8*	*2*	7	*5*
9	*7*	1	*3*	*8*	*5*	6	*2*	*4*
5	4	*3*	*7*	*2*	*6*	*8*	1	9
6	*8*	2	*1*	*4*	*9*	7	*5*	*3*
7	9	*4*	*6*	3	*2*	*5*	8	*1*
2	*6*	*5*	8	*1*	4	*9*	*3*	7
3	1	*8*	9	*5*	7	*4*	6	*2*

w="Solved Sudoku"

Proof $\longrightarrow$

Accept or Reject $\longleftarrow$



Peggy: (x, w)

Victor: x

A process in which a prover probabilistically convinces a verifier of the correctness of a mathematical proposition, and the verifier learns nothing else.

What are ZK Proofs?

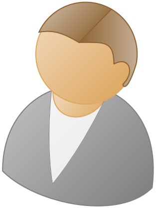
x = CircuitSat = (There exists w s.t. $C(w) = 1$) $\qquad w$

x = (There exist (p, q) s.t. $N = pq$) $\qquad w = (p, q)$

x= (I know sk) $\qquad$ sk

x is true ($x \in \mathcal{L}$) $\longrightarrow$

$\longleftarrow$ Accept or Reject

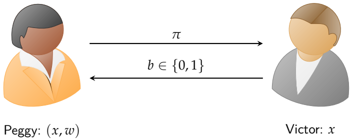


Peggy: (x, w)

Victor: x

A process in which a prover probabilistically convinces a verifier of the correctness of a mathematical proposition, and the verifier learns nothing else.

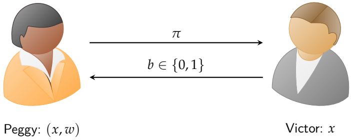
Properties of ZKProofs



- **Completeness.** If Peggy and Victor behave honestly, the proof will be accepted.
- **Soundness.** Peggy cannot prove false statements.
- **Zero-Knowledge.** Victor learns nothing beyond the truth of the statement.
- **Of Knowledge.** Victor is conviced that the prover knows a witness for the statement being true.

What is a “good” ZK Proof

Performance measured in different parameters.



- Expressivity.
- Prover complexity/ Verifier complexity.
- Proof size.
- Weaker/ Stronger Computational assumptions.
- Need for a trusted Setup.
- Amount of interaction.
- Of Knowledge.
- Private vs Public Verification...

(Pairing-Based) (zk)-SNARKs

ZK-**S**uccinct **N**on-Interactive **A**rguments of **K**nowledge

- Language: circuit satisfiability.
- Verifier: super efficient (and public).
- Proof: succinct.
- Long Structured Reference String.
- Very strong Assumptions

ZK Proofs History: The Hunting of the SNARK

1989 — Interactive Proof-Systems [GMR89]

— (...)

2010 — Groth. Succinct argument without PCPs (42 bilinear group elements)

2013 — QAPs: ZK friendly characterization of NP, linear SRS [GGPR13]
Implementation: Pinocchio: Nearly Practical Verifiable Computation" [PGH

2014 — ZeroCash

2016 — Groth. Most efficient zk-SNARK (3 bilinear group elements)

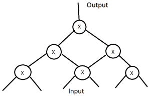
.... — and so much more...

Non-universal SNARKs: Technical core

Overview

- Information Theoretic Step: statement is encoded in a convenient way[1].

CircuitSat Relation
Circuit, $\vec{c}$



$\rightarrow$

Algebraic Relation
Rank 1 Constraint System

$\mathbf{L}, \mathbf{R}$ s.t.
$\vec{c}$ satisfies circuit iff
$\mathbf{L}\vec{c} \circ \mathbf{R}\vec{c} = \vec{c}$

$\rightarrow$

Polynomial Relation
Quadratic Arithmetic Program

$t(X), \{v_i(X), w_i(X), \lambda_i(X))$
s.t.$\vec{c}$ satisfies circuit $\Leftrightarrow$
$t(X)$ divides
$\left(\sum_i c_i v_i(X)\right)\left(\sum_i c_i w_i(X)\right) - \sum_i c_i \lambda_i(X)$

- Computational Step: statement is compressed.

Quadratic Arithmetic Program

$t(X), \{v_i(X), w_i(X), \lambda_i(X)\}_i$ $\xrightarrow{\text{Compiler}}$

SNARK

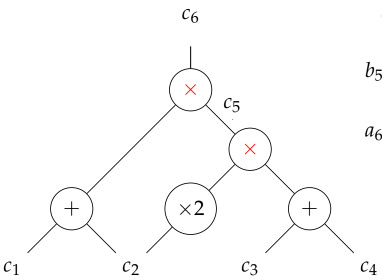
SRS, π

[1] For ease of presentation in this talke we R1CS to refer to a simpler form called R1CS-lite due to Campanelli et al. Asiacrypt'21.

From Circuit to Algebraic Relations

$$C : \mathbb{Z}_p^4 \to \mathbb{Z}_p, \qquad C(c_1, c_2, c_3, c_4) = (c_1 + c_2)(2c_2(c_3 + c_4)).$$

$\vec{a}$, $\vec{b}$, $\vec{c}$: left, right and output wires for multiplication gates.



$$a_5 = (2c_2)$$

$$b_5 = (c_3 + c_4)$$

$$a_6 = (c_1 + c_2) \qquad c_5 = a_5 b_5$$

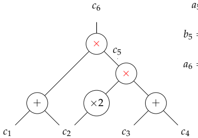
$$b_6 = c_5 \qquad c_6 = a_5$$

Hadamard Product Relation: $\begin{pmatrix} a_5 \\ a_6 \end{pmatrix} \circ \begin{pmatrix} b_5 \\ b_6 \end{pmatrix} = \begin{pmatrix} c_5 \\ c_6 \end{pmatrix}$

Linear Relations:

$$\begin{pmatrix} a_5 \\ a_6 \end{pmatrix} = \begin{pmatrix} 0 & 2 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \end{pmatrix} \vec{c} = \mathbf{F}\vec{c}, \qquad \begin{pmatrix} b_5 \\ b_6 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \vec{c} = \mathbf{G}\vec{c}.$$

From Circuit to Algebraic Relations, simplified



$a_5 = (2c_2)$

$b_5 = (c_3 + c_4)$

$a_6 = (c_1 + c_2)$ $\quad$ $c_5 = a_5 b_5$

$b_6 = c_5$ $\quad$ $c_6 = a_5$

Hadamard Product Relation: $\vec{a} \circ \vec{b} = \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \end{pmatrix} \circ \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \end{pmatrix} = \begin{pmatrix} 1 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \end{pmatrix}$

Linear Relations:

$\vec{a} = \mathbf{L}\vec{c}$, $\vec{b} = \mathbf{R}\vec{c}$, where $\mathbf{L} = \begin{pmatrix} \mathbf{I}_{5\times 5} \\ \mathbf{F} \end{pmatrix}$, $\mathbf{R} = \begin{pmatrix} \tilde{\mathbf{I}} \\ \mathbf{G} \end{pmatrix}$, $\tilde{\mathbf{I}} = \begin{pmatrix} \vec{1}_5 & \mathbf{0}_{5\times 4} \end{pmatrix}$.

From Circuit to Algebraic Relations, Example

$$C : \mathbb{Z}_p^4 \to \mathbb{Z}_p, \qquad C(c_1, c_2, c_3, c_4) = (c_1 + c_2)(2c_2(c_3 + c_4)).$$

Statement: There exists c_3, c_4 such that $C(1, 2, c_3, c_4) = 84$.

1. **Public Input Relations:**
   $c_0 = 1$, $c_1 = 1$, $c_2 = 2$, $c_6 = 84$.
2. **Hadamard Product Relation**:
   $\vec{a} \circ \vec{b} = \vec{c}$
3. **Linear Relations**:
   $\vec{a} = \mathbf{L}\vec{c}$, $\vec{b} = \mathbf{R}\vec{c}$.

Witness: $\vec{c} = (1, 1, 2, 3, 4, 28, 84)^\top \in (\mathbb{F}_p)^7$.

Hadamard Product and Lagrange Interpolation

- Let $\mathcal{R} = \{r_0, \ldots, r_{m-1}\}$ multiplicative subgroup of $\mathbb{F}_p^*$, $\lambda_i(X)$ ith Lagrange interpolation polynomial:

$$\lambda_i(X) = \prod_{j \neq i} \frac{(X - r_j)}{(r_i - r_j)}, \qquad \lambda_i(r_j) = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}, \qquad t(X) = \prod_j (X - r_j)$$

$$\boldsymbol{\lambda}(X)^\top = (\lambda_0(X), \ldots, \lambda_{m-1}(X)).$$

- We can encode vectors as polynomials to do "linear algebra"with polynomials:

$$\vec{y} = (y_0, \ldots, y_{m-1}) \longleftrightarrow y(X) = \sum_{i=0}^{m-1} y_i \lambda_i(X) = \boldsymbol{\lambda}(X)^\top \vec{y} \qquad \textbf{Obs:}\, y(r_j) = y_j$$

- Hadamard Product can be encoded as divisibility relation: for any $\vec{c}$, $\vec{a}$, $\vec{b}$,

$$a(X)b(X) - b(X) = H(X)t(X) \iff c = \vec{a} \circ \vec{b}$$

Linear Relations as Polynomial Relations

$$\mathbf{L} = \begin{pmatrix} v_0(r_0) & \dots & v_6(r_0) \\ \vdots & & \vdots \\ v_0(r_6) & \dots & v_6(r_6) \end{pmatrix} \iff \boldsymbol{\lambda}(X)^\top \mathbf{L} = (v_0(X), \dots, v_6(X)),$$

$$\mathbf{R} = \begin{pmatrix} w_0(r_0) & \dots & w_6(r_0) \\ \vdots & & \vdots \\ w_0(r_6) & \dots & w_6(r_6) \end{pmatrix} \iff \boldsymbol{\lambda}(X)^\top \mathbf{R} = (w_0(X), \dots, w_6(X)),$$

$\vec{a} = \mathbf{L}\vec{c}$ AND $\vec{b} = \mathbf{R}\vec{c}$ $\iff$

$$a(X) = \boldsymbol{\lambda}(X)^\top \vec{a} = \boldsymbol{\lambda}(X)^\top \mathbf{L}\vec{c} = \sum_{j=0}^{6} c_j w_j(X) \text{ AND}$$

$$b(X) = \boldsymbol{\lambda}(X)^\top \vec{b} = \boldsymbol{\lambda}(X)^\top \mathbf{R}\vec{c} = \sum_{j=0}^{6} c_j w_j(X)$$

From Algebraic Relations to Polynomial Relations

In summary

- Public Input Relation:

$$c_0 = 1, c_1 = 1, c_2 = 2, c_6 = 84.$$

- Hadamard Relation:

$$a(X)b(X) - c(X) = H(X)t(X).$$

- Linear Relations: There exists $\vec{c}$ such that

$$a(X) = \sum_{j=0}^{6} c_j v_j(X) \qquad b(X) = \sum_{j=0}^{6} c_j v_j(X) \qquad c(X) = \sum_{j=0}^{6} c_j \lambda_j(X)$$

We discuss how to prove linear relations next.

Bilinear map or Pairing

Compressing or Computational Step

- Implicit notation: $[a]_i = a\mathcal{P}_i$.

Definition

$\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ cyclic groups of order p where DLOG is hard, $\mathcal{P}_1, \mathcal{P}_2$ generators of $\mathbb{G}_1, \mathbb{G}_2$ respectively, $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a non-degenerate bilinear map (or pairing) if

- for all $([\alpha]_1, [\beta]_2) \in \mathbb{G}_1 \times \mathbb{G}_2$,

$$e([\alpha]_1, [\beta]_2) = e(\mathcal{P}_1, \mathcal{P}_2)^{\alpha\beta} \text{ (Bilinearity)},$$

- $e([\alpha]_1, [\beta]_2) \neq 1_{\mathbb{G}_T}$ (Non-degeneracy)

(Bilinear) groups: What can we efficiently do?

- (Recall: Implicit notation: $[a] = a\mathcal{P}$, group of order p).
- Essentially all we can efficiently do: given $[x_1], \ldots, [x_n]$, compute combinations with known linear coefficients $c_i \in \mathbb{Z}_p$:

$$\sum c_i [x_i].$$

- In particular, given some element $[p(\tau)]$ a polynomial $p(X)$ with known coefficients $c_i \in \mathbb{Z}_p$, and $[1], [\tau], \ldots, [\tau^q]$:
  - If $p(X)$ is divisible by $t(X)$: $[p(\tau)/t(\tau)]$ easy to compute.

$$h(X) := p(X)/t(X), \qquad [p(\tau)/t(\tau)]_1 = \sum h_i [\tau^i].$$

  - If $p(X)$ is not divisible by $t(X)$: $[p(\tau)/t(\tau)]$ hard to compute (q-Strong Diffie Hellman type of assumption).

SNARK construction (an abstraction of [ParGenHowRay13])

Setup: Chooses $\tau \leftarrow \mathbb{Z}_p$, evaluates $t(\tau), \{v_i(\tau), w_i(\tau), \lambda_i(\tau)\}_i$ and appends $[t(\tau)]_{1,2}$, $[v_i(\tau)]_1$, $[w_i(\tau)]_2$, $[\lambda_i(\tau)]_1$, $[\tau^i]_{1,2}$ to SRS.

SNARK construction (an abstraction of [ParGenHowRay13])

Setup: Chooses $\tau \leftarrow \mathbb{Z}_p$, evaluates $t(\tau), \{v_i(\tau), w_i(\tau), \lambda_i(\tau)\}_i$ and appends $[t(\tau)]_{1,2}$, $[v_i(\tau)]_1$, $[w_i(\tau)]_2$, $[\lambda_i(\tau)]_1$, $[\tau^i]_{1,2}$ to SRS.
Prover (SRS, $\vec{c}$): Samples $\delta_1, \delta_2, \delta_3 \leftarrow \mathbb{Z}_p^*$, and doing linear combination of elements of SRS computes:

$$A = [a(\tau) + \delta_1 t(\tau)]_1 \qquad B = [b(\tau) + \delta_2 t(\tau)]_2,$$
$$C = [c(\tau) + \delta_3 t(\tau)]_1, \text{ and}$$

1. A proof H that divisibility relation holds at point τ.

$$H = [\frac{1}{t(\tau)}\Big((a(\tau)b(\tau) - c(\tau) + (\delta_1\delta_2 - \delta_3)t(\tau)\Big)]_1,$$

2. A proof Π that A, B, C are well formed, in "span" of $\{v_i(\tau)\}$ (resp. $\{w_i(\tau)\}$, $\{\lambda_i(\tau)\}$ for same witness.)

SNARK construction (an abstraction of [ParGenHowRay13])

Setup: Chooses $\tau \leftarrow \mathbb{Z}_p$, evaluates $t(\tau), \{v_i(\tau), w_i(\tau), \lambda_i(\tau)\}_i$ and appends $[t(\tau)]_{1,2}$, $[v_i(\tau)]_1$, $[w_i(\tau)]_2$, $[\lambda_i(\tau)]_1$, $[\tau^i]_{1,2}$ to SRS.
Prover (SRS, $\vec{c}$): Samples $\delta_1, \delta_2, \delta_3 \leftarrow \mathbb{Z}_p^*$, and doing linear combination of elements of SRS computes:

$$A = [a(\tau)+\delta_1 t(\tau)]_1 \qquad B = [b(\tau)+\delta_2 t(\tau)]_2,$$

$$C = [c(\tau)+\delta_3 t(\tau)]_1, \text{ and}$$

1. A proof H that divisibility relation holds at point τ.

$$H = [\frac{1}{t(\tau)}\Big((a(\tau)b(\tau) - c(\tau)+(\delta_1\delta_2 - \delta_3)t(\tau)\Big)]_1,$$

2. A proof Π that A, B, C are well formed, in "span" of $\{v_i(\tau)\}$ (resp. $\{w_i(\tau)\}$, $\{\lambda_i(\tau)\}$ for same witness.)

Verifier (SRS, H, A, B, C):

1. Checks well-formedness of A, B, C + divisibility at point τ using pairings

$$e(H, [t(\tau)]_2) \stackrel{?}{=} e(A, B)e(C, [1]_2)^{-1}.$$

SNARK construction: Linear Relations
(Simplified)

Proof that A, B, C is in the span of $\{v_i(\tau)\}$, $\{w_i(\tau)\}, \{\lambda_i(\tau)\}$ (with same $\vec{c}$):

1. Include in SRS:
$$\left(\{[\alpha v_i(\tau) + \beta w_i(\tau) + \gamma\lambda_i(\tau)]_1\}, [\alpha]_2, [\beta]_2, [\gamma]_2\right)$$
2. Prover:
$$\pi' = \sum_{i=0}^{6} c_i[(\alpha v_i(\tau) + \beta w_i(\tau) + \gamma\lambda_i(\tau))]_1$$
3. Verifier:
$$e(A, [\alpha]_2) + e([\beta]_1, B) + e(C, [\gamma]_2) \stackrel{?}{=} e(\pi', [1]_2).$$

SNARK construction: Security

- Perfect Zero-Knowledge: Randomization! (proof distribution is uniform conditioned on being accepted by Verifier.)

- Soundness:
  1. Extract a "witness candidate"$\vec{c}$ from proof of well formedness, i.e.

$$A = \sum c_i v_i(\tau), \qquad B = \sum c_i w_i(\tau) \qquad C = \sum c_i \lambda_i(\tau).$$

  2. If adversary breaks soundness, $p(X) = (\sum c_i v_i(X))(\sum c_i w_i(X)) - (\sum c_i \lambda_i(X))$ not divisible by $t(X)$, but adversary has computed $p(\tau)/t(\tau)$ in the exponent!!

     For soundness, it is crucial that s is secret!!

  3. Linear Relations:

     For soundness, it is crucial that α, β, γ are secret and tied together to polynomials $\{v_i(X), w_i(X), \lambda_i(X)\}$. They cannot be reused for the SRS for another circuit!!!

SNARK construction: Security II

- Step 2 and 3 are standard: for Step 1, we need a non-falsifiable assumption.

Definition (q-Power Knowledge of Exponent Assumption)

For every PPT $\mathcal{A}$ which, on input $[1]_1, [\tau]_1, \ldots, [\tau^q]_1$ and $[\alpha]_1, [\alpha]_2, [\alpha\tau]_1, \ldots, [\alpha\tau^q]_1$, outputs $V, \alpha V \in \mathbb{G}_1$, there exists a PPT extractor which outputs $c_1, \ldots, c_q \in \mathbb{Z}_p$ such that $V = \sum c_i \tau^i$.

Non-falsifiable Assumption. Black-box extraction is information theoretically impossible, would also mean the SNARK contradicts known impossibility results (e.g. [GenWic11])

Remarks

- Construction generalizes to case where some $c_1, \dots, c_\ell$ are public (as in example)
- Simulation: given $\tau \in \mathbb{Z}_p$, we can simulate any proof by dividing by $t(\tau)$!!
- Best zk-SNARK construction by Groth 2016 based on similar ideas.

m Circuit size, ℓ public inputs,

- Prover computation $O(m \log m)$.
- Verifier's computation 3 Pairings $+ O(\ell)$ exponentiations.
- Constant communication complexity! (just 3 group elements in Groth16)

Setups

Motivation: SRS

Observation

The SRS in the previous SNARK consists of two pieces: (given as points in an elliptic curve)

(1) A part that is *circuit-independent*, or *universal*: $1, \tau, \tau^2, \ldots$

(2) A part that is *circuit-dependent*: α, β, γ, $\{\alpha v_j(\tau) + \beta w_j(\tau) + \gamma \lambda_j(\tau)\}_{j=1,\ldots,m}$

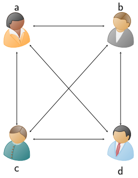
- (1) Can be generated once for all circuits (2) needs to be generated for each circuit.
- In both cases, the information used to generate the SRS can be used to completely break security.

In the SRS generator we trust...

Z. Wilcox (ZCash) on his knees destroying a computer after parameter generation. https://z.cash/technology/paramgen/

- SNARKs require a trusted party to generate the parameters.
- Knowledge of randomness to generate parameters: complete failure.
- Solution: distribute trust.
- Two problems: how to update an SRS? How can we avoid doing this expensive setup for each circuit?

SNARKs: Updatable Model [GroKohMalMeiMie18]



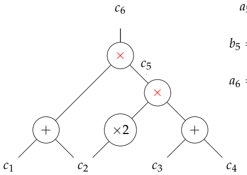
Multiparty Computation Model

Updatable Model

- Updatable Model: for soundness it suffices that one party is honest, and SRS can always be updated NI.
- In [BowGabMie17]: after a trusted setup phase to generate $[\tau], [\tau^2], \ldots, [\tau^q]$, circuit dependent setup is updatable.
- [GroKohMalMeiMie18]: Universal and (single phase) updatable setup: universal setup is updatable, circuit dependent setup is public, no secrets involved (just preprocessing.)

Universal and Updatable SNARKs: Technical Core

From Circuit to Algebraic Relations, simplified



$a_5 = (2c_2)$

$b_5 = (c_3 + c_4)$

$a_6 = (c_1 + c_2)$ $\qquad$ $c_5 = a_5 b_5$

$b_6 = c_5$ $\qquad$ $c_6 = a_5$

Hadamard Product Relation:

$$\vec{a} \circ \vec{b} = \vec{c}$$

Universal

Linear Relations:

$$\vec{a} = \mathbf{L}\vec{c},\ \vec{b} = \mathbf{R}\vec{c},\ \text{or equivalently,}\ \begin{pmatrix} -\mathbf{I} & \mathbf{0} & \mathbf{L} \\ \mathbf{0} & -\mathbf{I} & \mathbf{R} \end{pmatrix} \begin{pmatrix} \vec{a} \\ \vec{b} \\ \vec{c} \end{pmatrix} = \vec{0}.$$

Previous techniques to prove this relation required circuit-dependent trusted parameters!! New techniques for Linear Relations are necessary.

From Algebraic Relations to Polynomials

Inner Product Relations and the Univariate Sumcheck

- $\mathcal{R} = \{r_0, \ldots, r_{m-1}\} \subset \mathbb{F}_p^*$, 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 $\vec{y} = (y_0, \ldots, y_{m-1})$	Polynomial $\sum_{i=0}^{m-1} y_i \lambda_i(X)$
Inner product $z = \vec{w} \cdot \vec{y}$	[Ben-Sasson et al. 18][2] $w(X)y(X) - m^{-1}z = XR(X) + H(X)t(X)$ for some $R(X)$ s.t. $deg\ R(X) \leq m - 2$.

[2] In [RZ21] new proof where $\mathcal{R}$ is not necessarily a subgroup.

From Algebraic Relations to Polynomials

Inner Product Relations and the Univariate Sumcheck

Algebraic Formulation	Polynomial Formulation
Vector $\vec{y} = (y_0, \ldots, y_{m-1})$	Polynomial $\sum_{i=0}^{m-1} y_i \lambda_i(X)$
Inner product $z = \vec{w} \cdot \vec{y}$	[Ben-Sasson et al. 18] $w(X)y(X) - m^{-1}z = XR(X) + H(X)t(X)$ for some $R(X)$ s.t. $deg\ R(X) \leq m-2$.

From Algebraic Relations to Polynomials

Inner Product Relations and the Univariate Sumcheck

Algebraic Formulation	Polynomial Formulation
Vector $\vec{y} = (y_0, \ldots, y_{m-1})$	Polynomial $\sum_{i=0}^{m-1} y_i \lambda_i(X)$
Inner product $z = \vec{w} \cdot \vec{y}$	[Ben-Sasson et al. 18] $w(X)y(X) - m^{-1}z = XR(X) + H(X)t(X)$ for some $R(X)$ s.t. $deg\ R(X) \leq m-2$.

Proof: Let $P(X) = \sum w_i y_i \lambda_i(X)$. It holds that $w(X)y(X) = P(X) + H(X)t(X)$. But, evaluating at 0, and using that $\lambda_i(0) = m^{-1}$ for all i, if $\mathcal{R}$ is a subgroup of roots of unity, $P(0) = m^{-1}\vec{y} \cdot \vec{w}$. Therefore, $P(X) - zm^{-1}$ is 0 at 0 if and only if $z = \vec{y} \cdot \vec{w}$.

How to Prove Many Inner Product Relations

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

$$\text{Checking if } \mathbf{M}\vec{x} = \vec{0} \quad vs \quad \text{Checking if } (\vec{v}^\top \mathbf{M}) \cdot \vec{x} = \vec{0},$$

where $\vec{v}$

is sufficiently random!!

- **Problem** Although matrix $\mathbf{M}$ is public, a sublinear verifier cannot afford to sample a random vector in rowspace of $\mathbf{M}$ (since in the case of interest the number of rows of the matrix is two times the size of the circuit!)

From Algebraic Relations to Polynomials

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

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

- Given random x, the vector

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

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

- The partial evaluation

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

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

Polynomial Relations for a Universal SNARK

Define $P^{-\mathbf{I}}(X,Y)$, $P^{\mathbf{L}}(X,Y)$ and $P^{\mathbf{R}}(X,Y)$ bivariate encodings of matrices $-\mathbf{I},\mathbf{L},\mathbf{O}$.

- Compute $a(X), b(X), b(X)$ the polynomial encoding of $\vec{a}, \vec{b}, \vec{c}$ and prove the Hadamard product relation $\vec{a} \circ \vec{b} = \vec{c}$.
- Verifier sends challenge x.
- Prover samples polynomial $D^{\mathbf{I}}(X) = P^{\mathbf{I}}(X,x)$ and $P^{\mathbf{L}}(X,x)$ which is the encoding of random vectors $\vec{d}_{-\mathbf{I}}$ and $\vec{d}_{\mathbf{L}}$ in the span of $-\mathbf{I}$ and $\mathbf{L}$.
- Prover shows that the inner product of $\vec{d}_{-\mathbf{I}} \cdot \vec{a} = \vec{d}_{\mathbf{L}} \cdot \vec{c}$.
- Prover repeats last two steps for proving $\vec{b} = \mathbf{R}\vec{c}$.

Polynomial Relations for a Universal SNARK

Define $P^{-\mathbf{I}}(X,Y)$, $P^{\mathbf{L}}(X,Y)$ and $P^{\mathbf{R}}(X,Y)$ bivariate encodings of matrices $-\mathbf{I}, \mathbf{L}, \mathbf{O}$.

- Compute $a(X), b(X), b(X)$ the polynomial encoding of $\vec{a}, \vec{b}, \vec{c}$ and prove the Hadamard product relation $\vec{a} \circ \vec{b} = \vec{c}$.
- Verifier sends challenge x.
- Prover samples polynomial $D^{\mathbf{I}}(X) = P^{\mathbf{I}}(X,x)$ and $P^{\mathbf{L}}(X,x)$ which is the encoding of random vectors $\vec{d}_{-\mathbf{I}}$ and $\vec{d}_{\mathbf{L}}$ in the span of $-\mathbf{I}$ and $\mathbf{L}$.
- Prover shows that the inner product of $\vec{d}_{-\mathbf{I}} \cdot \vec{a} = \vec{d}_{\mathbf{L}} \cdot \vec{c}$.
- Prover repeats last two steps for proving $\vec{b} = \mathbf{R}\vec{c}$.

Problem: How can verifier test that $D(X)'s$ are correct?

Checkable Subspace Sampling [RafZap21]

Definition

- *Offline phase:* A $\mathbf{M}$ is preprocessed and encoded as a set of polynomials.
- *Online phase:*
  - *Sampling:*
    Interactive protocol in which Verifier sends random challenge α and Prover outputs polynomial $D(X)$.
  - *Prove Sampling:*
    Prover computes proof π that $D(X)$ is sampled correctly.
- *Decision phase:* Verifier accepts iff $D(X)$ encodes the vector $\vec{v}_\alpha^\top \mathbf{M}$ for some sufficiently random vector $\vec{v}_\alpha$ determined by challenge α.

Sampling in the rowspace is delegated to the prover, who needs to show that it is sampling the vector according to the coins of the verifier.

Which matrices have efficient Checkable Subspace Sampling?

Results of [RZ21]

- Sparse Matrices (Marlin)
- Matrices with a bounded number of non-zero elements per column.
- Matrices with Low Tensor Rank.
- Any combination of those.

Example CSS

$\mathbf{M} = (m_{ij}) \in \mathbb{F}^{m \times m}$ a matrix with one non-zero element per column. Number non-zero values from 1 to m such that $m_{\mathsf{row}(\ell),\ell} \neq 0$ for some functions val : $[m] \to \mathbb{F}$, row : $[m] \to [m]$.

- **Offline Phase:** On input $\mathbb{F}_p, \mathbf{M}$, the indexer outputs $\{v_1(X), v_2(X)\}$, where

$$v_1(X) = \sum_{\ell=1}^{m} r_{\mathsf{row}(\ell)} \lambda_\ell(X), \qquad v_2(X) = m^{-1} \sum_{\ell=1}^{m} \mathsf{val}(\ell) r_{\mathsf{row}(\ell)} \lambda_\ell(X).$$

- **Online Phase:**
  - Sampling Phase: The verifier outputs $x \leftarrow \mathbb{F}$ and prover sends $D(X) = P(X, x)$.
  - Proving Phase: the prover finds and outputs $H(X)$ such that

$$D(X)\big(x - v_1(X)\big) = t(x) v_2(X) + H(X) t(X)$$

Conclusion

- We have identified the main challenges in building updatable and universal SNARK.
- In particular, we have explained that there is a certain building block in these SNARKs, a Checkable Subspace Sampling subargument, that is particularly challenging to build.
- The CSS Example is for a very simple matrix, but it gets more complex for more expressive types of matrices.
- In particular, the cost of the CSS represents a significant part of the prover cost in several universal and updatable SNARKs (like Sonic, Marlin, Lunar, Basilisk, Counting Vampires), where it is fundamental to guarantee sublinear verification.
- We did not cover Plonk, which is probably the most well known universal and updatable SNARK and which takes a different approach to deal with Linear Relations.