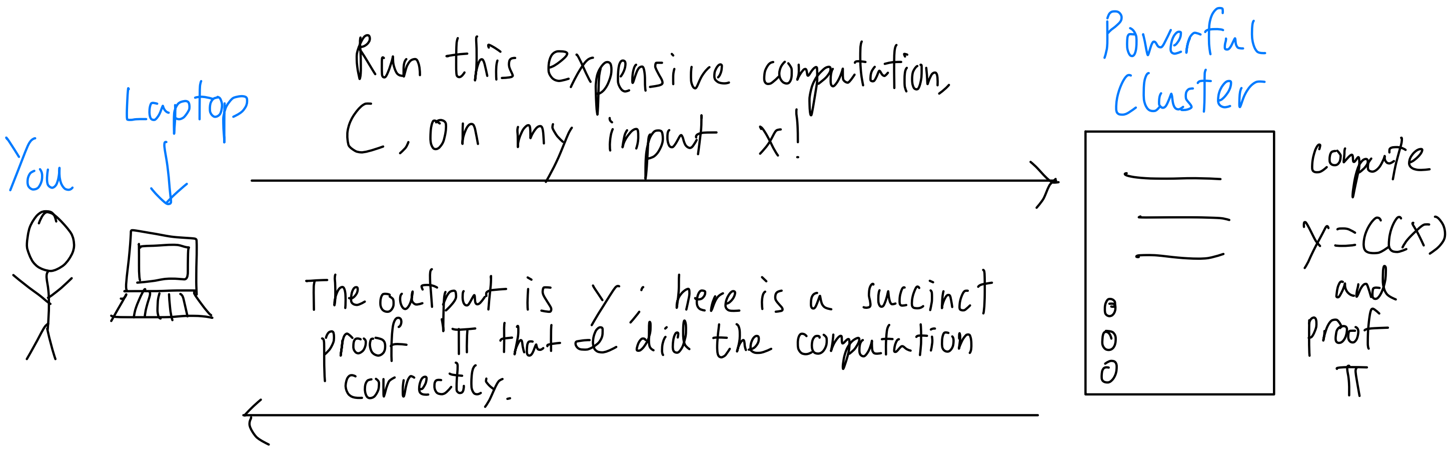
Outline

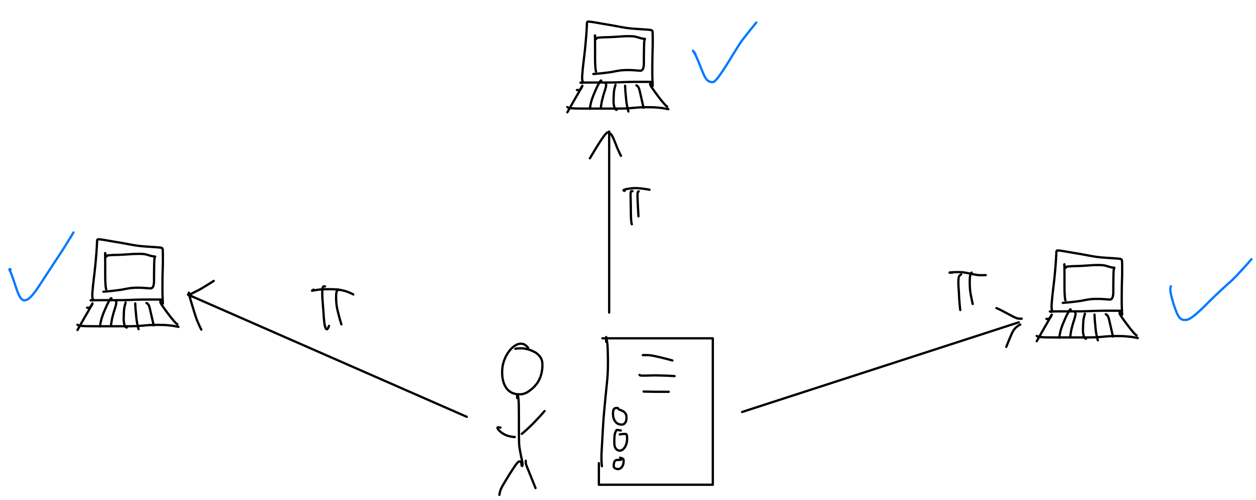
- succinct arguments / SNARGs
- PCP Theorem
- Our first SNARGs
- Poly IOPs / IOPs
- Polynomial Commitments / KZG

Motivation - Verifiable Computation

1) Outsourcing computation



2) Eliminating repetitive work (Blockchain)



What does it mean to be succinct?

(Informally) Let R be an NP relation, and let T denote the runtime of the associated NP verifier. An interactive protocol, (P, V), for R is <u>succinct</u> if when ran on input $(x, w) \in R$:

1) Total Communication is $o(|w|)$.
2) V's runtime is $\text{poly}(|x|) + o(|w| + T)$.

In particular, these quantities depend polynomially in the instance size, but sublinear in the witness size and runtime of the NP verifier, T.

What does it mean to be non-interactive?

The complete interaction consists of just a single message, π, from the Prover to the Verifier.

PCP Theorem (informal)

$\forall$ NP relations R, there exists an interactive protocol, $(\tilde{P}, \tilde{V})$, called a probabilistically checkable proof, such that $\tilde{P}(x, w)$ outputs a long proof $\tilde{\pi} \in \{0,1\}^*$. To check this proof, the verifier $\tilde{V}^{\tilde{\pi}}(x)$ surprisingly only has to read a small number of bits (can be just 3!). This protocol is both complete and sound.

$\tilde{P}(x, w)$ $\qquad\qquad$ $\tilde{V}(x; r)$

$$\xrightarrow{\tilde{\pi} \in \{0,1\}^*}$$

- Sample Query Locations in $\{1, \ldots, |\tilde{\pi}|\}$

  $(i_1, i_2, i_3) \leftarrow \text{Sample}(r)$

- Execute a decision procedure based on the queried bits.

  $b \leftarrow \text{Decide}((\tilde{\pi}[i_1], \tilde{\pi}[i_2], \tilde{\pi}[i_3]), r)$

- Output the decision bit b.

From PCPs to SNARGs (Kilian & Micali construction)

$P(x, w)$ $\qquad\qquad$ $V(x; r)$

- Call the PCP Prover

  $\tilde{\pi} \leftarrow \tilde{P}(x, w)$

- Commit to $\tilde{\pi}$ using a Merkle Tree

  $c \leftarrow \text{Merkle.Commit}(\tilde{\pi})$

$$\xrightarrow{\quad c \quad}$$

$$\xleftarrow{\quad i_1, i_2, i_3 \quad}$$

$(i_1, i_2, i_3) \leftarrow \text{Sample}(r)$

- Open those indices of the Merkle tree

  $\pi_{MT} \leftarrow \text{Merkle.Open}((i_1, i_2, i_3), \tilde{\pi})$

$$\xrightarrow{(\tilde{\pi}[i_1], \tilde{\pi}[i_2], \tilde{\pi}[i_3]),\ \pi_{MT}}$$

Output

$\text{Decide}((\tilde{\pi}[i_1], \tilde{\pi}[i_2], \tilde{\pi}[i_3]), r)$

$\wedge$

$\text{Merkle.Verify}(c, (i_1, i_2, i_3), (\tilde{\pi}[i_1], \tilde{\pi}[i_2], \tilde{\pi}[i_3]), \pi_{MT})$

Completeness: Follows from completeness of Merkle Commitments and PCP

Soundness: Follows from index binding of MTs and soundness of PCP.

Non-Interactive: Can be made non-interactive in ROM with Fiat-Shamir.

Succinctness: $\underbrace{O_\lambda(1)}_{|c| + |\text{bits}|} + O_\lambda(\log(|\tilde{\pi}|))$

poly in $|w|$

This SNARG can be viewed more abstractly as instantiating an Interactive Oracle Proof with a succinct vector commitment scheme.

IOP + Vector Commitment

$$P \xrightarrow[\text{Oracles to Vectors}]{\boxed{V_i}} V$$

Can query vectors as oracles

$$P \xleftarrow[\text{randomness}]{r_i} V$$

Merkle Tree Commitment

||

"Information Theoretic Proof System" + Concrete cryptographic Commitment to oracle

In practice, the Kilian-Micali SNARG is wildly impractical due to the concrete cost of producing and commiting to the PCP $\tilde{\pi}$.

However, by generalizing the types of oracles and increasing the rounds, we will be able to obtain practically efficient SNARKs by following this paradigm. In particular,

	Info Theoretic Proof	+	Cryptographic Commitment
	IOP (vector oracles)	+	Vector Commitments
	$\downarrow$		$\downarrow$
☆ next lecture	Polynomial IOP (polynomial oracles)	+	Polynomial Commitments ☆ Today

Polynomial Commitment Schemes (PCS)

A PCS enables a commiter to commit to a polynomial over a finite field $\in \mathbb{F}[X]$ such that they can open the polynomial at any point in the field. A PCS is a tuple of 4 efficient algs:

- $\mathrm{Setup}(1^\lambda, d) \to pp$: takes in sec param and degree bound $\in \mathbb{N}$.
- $\mathrm{Commit}(pp, f) \to C$: takes in pp, poly $\in \mathbb{F}[X]$ of degree $\leq d$ and outputs a commitment.

- Open $(pp, f, z) \to \pi$: takes in pp, $f \in \mathbb{F}[x]$, $z \in \mathbb{F}$, and outputs a proof π that attests to the eval $y = f(z)$.

- Verify $(pp, c, z, y, \pi) \to 0/1$: Checks the evaluation proof π with respect to commitment c, point z, and claimed eval y.

Properties:

Correctness: $\forall$ poly-bounded $d \in \mathbb{N}$, $f \in \mathbb{F}[x]$, $z \in \mathbb{F}$,

$$\Pr\left[\text{Verify}(pp, c, z, f(z), \pi) = 1 \;:\; \begin{array}{l} pp \leftarrow \text{Setup}(1^\lambda, d) \\ c \leftarrow \text{Commit}(pp, f) \\ \pi \leftarrow \text{Open}(pp, f, z) \end{array}\right] = 1$$

Evaluation Binding: $\forall$ poly-bounded $d \in \mathbb{N}$, $\forall$ PPT A,

$$\Pr\left[\begin{array}{l} \text{Verify}(pp, c, z, y, \pi) = 1 \\ \quad \wedge \\ \text{Verify}(pp, c, z, y', \pi') = 1 \\ \wedge \quad y \neq y' \end{array} \;:\; \begin{array}{l} pp \leftarrow \text{Setup}(1^\lambda, d) \\ (c, z, y, \pi, y', \pi') \leftarrow A(pp) \end{array}\right]$$

$$\leq \text{negl}(\lambda)$$

KZG Commitments:

A polynomial commitment scheme from pairings!

Trusted Setup!

$\text{Setup}(1^\lambda, d)$:

Let's assume the group elts come with the description of the BG.

1) Sample a bilinear group,

$$\mathbb{B} := (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, e) \leftarrow \text{BGSample}(1^\lambda)$$

2) Sample a secret value $\boxed{\alpha \xleftarrow{\$} \mathbb{Z}_p.}$ Toxic waste - private randomness

3) Output $pp := \left(\{ g_1^{\alpha^i} \}_{i \in \{0,..,d\}}, \; g_2^{\alpha} \right)$

$\text{Commit}(pp, f)$:

1) Parse $\left(\{ g_1^{\alpha^i} \}_i, \; g_2^{\alpha} \right) \leftarrow pp.$

2) Output $c := \underbrace{\prod_{i=0}^{d} (g_1^{\alpha^i})^{f_i}}_{\text{Notice } c = g_1^{f(\alpha)}!}$ where f_i is the i-th coeff of f.

$\text{Open}(pp, f, z)$:

1) Parse $\left(\{ g_1^{\alpha^i} \}_i, \; g_2^{\alpha} \right) \leftarrow pp.$

2) Compute the quotient $\dfrac{\overset{\text{poly}}{f(X)} - \overset{\text{eval}}{f(z)}}{X - z} = \overset{\text{deg } d-1}{q(X)}$

3) Output $\pi := \text{Commit}(pp, q)] = g_1^{q(\alpha)}$

Verify(pp, c, z, y, π):

1) Parse pp to obtain g_2^α.
2) Output accept iff

$$e(\pi, g_2^\alpha / g_2^z) = e(c / g_1^y, g_2)$$

Correctness:

$$e(\pi, g_2^\alpha / g_2^z) = e(g_1^{q(\alpha)}, g_2^{\alpha - z}) = e(g_1^{q(\alpha)(\alpha - z)}, g_2)$$

$$= e(g_1^{f(\alpha) - y}, g_2) = e(g_1^{f(\alpha)} / g_1^y, g_2)$$

$$= e(c / g_1^y, g_2)$$

Eval Binding: Reduces to breaking the d-BSDH assumption: "Given $(\{g_1^{\alpha^i}\}_{i \in \{0, \ldots, d\}}, g_2^\alpha)$ it is hard to come up with $(\gamma \in \mathbb{F}, h \in \mathbb{G}_T)$ such that $\gamma \neq \alpha$ and $h = e(g_1, g_2)^{\frac{1}{(\alpha - \gamma)}}$.

Consider $A(pp) \to (c, z, y, \pi, y', \pi')$ that breaks eval binding. Since both verification equations accept we know,

- $e(\pi, g_2^\alpha / g_2^z) = e(c / g_1^y, g_2) \Rightarrow \pi^{\alpha - z} \cdot g_1^y = c$
- $e(\pi', g_2^\alpha / g_2^z) = e(c / g_1^{y'}, g_2) \Rightarrow (\pi')^{\alpha - z} \cdot g_1^{y'} = c$

- Thus, we must have

$$\left(\pi/\pi'\right)^{\alpha - z} g_1^{\,y-y'} = 1 \Rightarrow \left(\pi/\pi'\right)^{\frac{1}{y'-y}} = g_1^{\frac{1}{\alpha-z}}$$

- Therefore, we output $\gamma = z$ and $h = e\left(\left(\pi/\pi'\right)^{\frac{1}{y'-y}}, g_2\right)$.

Now we can commit to polynomials!

Building Efficient SNARKs in Practice

- Trusted Setup

$$\mathrm{Setup}(1^\lambda) \to pp$$

performed by trusted party or multiparty computation

- Preprocessing Stage

desc of computation

$$pp + C \xmapsto{\text{Preprocess}} \begin{cases} \{f_i\}_i & \text{Proving key: polynomial encoding of } C \\ \{c_i\}_i & \text{Verification key: commits to poly encodings} \end{cases}$$

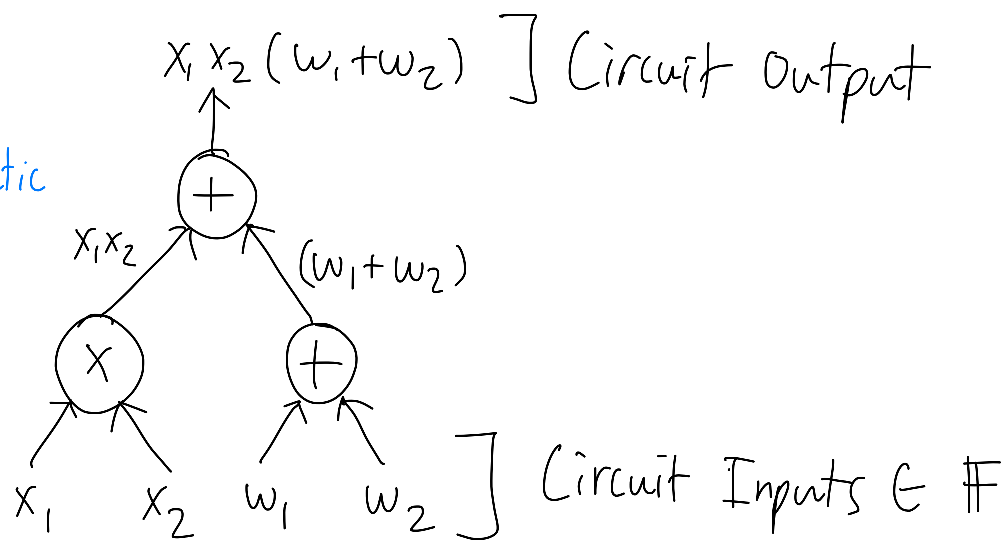
- Prover / Verifier

PolyIOP + PCS

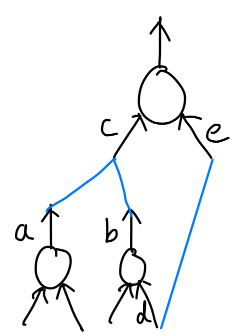
Arithmetic Circuits (a computational model)

- An arithmetic circuit is a directed acyclic graph where each node is an addition or multiplication gate with input-arity 2 and output-arity 1. The edges (wires) take on values from a specified field $\mathbb{F}$.
- These wires must satisfy the constraints enforced by the constraints.

Here is a simple example of an Arithmetic Circuit C with inputs $(x,w) \in \mathbb{F}^4$ that computes $x_1 x_2 (w_1 + w_2)$.



- Copy Constraints (optional): Field values can instead be assigned to gate pins (black) and wiring (blue) enforce equality. (For example, here we enforce $a = b = c$, $d = e$)



- We will consider the family of NP-Relations $\{R_C\}_C$ determined by Arithmetic Circuits C,

$$R_C := \{ (x ; w) \mid C(x,w) = 0 \}$$

Succinct Argument (SNARG)

A SNARG is a tuple of efficient algs

$$(\text{Setup}, \text{Preprocess}, P, V)$$

- $\underline{\text{Setup}(1^\lambda) \to pp}$: Takes in as input the security parameter, 1^λ, outputs public parameters, pp.
- $\underline{\text{Preprocess}(pp, C) \to (pk, vk)}$: Takes in as input pp and a description of a circuit C, outputs a proving key and succinct verification key.
- $\underline{P(pk, x, w) \to \pi}$: Takes in the proving key and instance x, witness w, outputs a succinct proof π.
- $\underline{V(vk, x, \pi) \to 0/1}$: Takes in the succinct verification key, vk, instance x, and proof π.

<u>Completeness:</u> $\forall C, (x, w) \in R_C$,

$$\Pr\left[V(vk, x, \pi) = 1 \;:\; \begin{array}{l} pp \leftarrow \text{Setup}(1^\lambda) \\ (pk, vk) \leftarrow \text{Preprocess}(pp, C) \\ \pi \leftarrow P(pk, x, w) \end{array} \right] \geq 1 - \text{negl}(\lambda)$$

<u>Soundness:</u> $\forall x \notin \mathcal{L}(R_C)$, $\forall$ PPT P^*,

$$\Pr\left[V(vk, x, \pi) = 1 \;:\; \begin{array}{l} pp \leftarrow \text{Setup}(1^\lambda) \\ (pk, vk) \leftarrow \text{Preprocess}(pp, C) \\ \pi \leftarrow P^*(pp) \end{array} \right] \leq \text{negl}(\lambda)$$

Succinctness:

Succinct Proof: $|\pi| \in o(|w|)$

Succint Verifier: $|V| \in O(|x|) + o(|w| + |C|)$

Verifier runtime

Non-interactive: ✓ by definition of P,V algs.