Last Time
Using the PCP Thm and Merkle Commitments, we constructed a SNARG for NP.

Issue: Wildly Impractical!
No one has implemented it in fact.

Today
We will start to give a construction for a ZK-SNARK for Arithmetic Circuit SAT that is practical!
Arithmetic Circuit SAT := ∃ \( (C, x) , w \) : \( C(x, w) = 1 \)

Using a definition of succinctness from last lecture:

Communication

\[ O(1 |C(x)|) + o(|w| + T) \]

Verifier Runtime

\[ \text{Runtime of NP Verifier} \]

\[ \text{just ends up being proportional to } |C| \]

Because the verifier \( V(X) \) has to read in the instance \( X := (C, x) \), which is proportional to the runtime \( T \), this notion of succinctness doesn't capture what we would naturally consider succinct; i.e. \( O(|X|) + o(|w| + |C|) \).

Solution!

What if we split the verifier into two stages:

\[ V = (V_{off}, V_{on}) \] where

**Offline Stage:** \( V_{off}(C) \xrightarrow{\dagger} ppv \) where \(|ppv| < |C| \) and \( V_{off} \) can run in time \( \tilde{O}(|C|) \)

**Online Stage:** \( V_{on}(ppv, x) \) interacts with \( P((C, x), w) \).

\(|ppv, x| < |C, x| |
More formally, define an index relation as pairs
\[ R := \{ (i, x, w) \mid (i, x) \in X \times W \} \]
where
\[ X := I \times X. \] I is called the index space, \( X \) is the instance space and \( W \) is the witness space.

Coming back to our example: \( I \) would be the space of circuits, \( X \) the space of public inputs, \( W \) the set of private inputs.

A preprocessing ZK-SNARK for an index relation \( R \) is a tuple of \( \text{eff, interactive alg.s} \) \((P, V := (V_{\text{off}}, V_{\text{on}}))\) with

Completeness: \( \forall (i, x) \in L(R), \)
\[ \Pr\left[ \langle P((i, x), w), V_{\text{on}}(pp_v, x) \rangle = 1 \mid pp_v \xleftarrow{\$} V_{\text{off}}(i) \right] \geq 1 - \text{neg}(\lambda) \]

Soundness: \( \forall (i, x) \notin L(R), \forall \text{PPT} P^* , \)
\[ \Pr\left[ \langle P^*, V_{\text{on}}(pp_v, x) \rangle = 1 \mid pp_v \xleftarrow{\$} V_{\text{off}}(i) \right] \leq \text{neg}(\lambda) \]

\( \exists \text{eff Sim s.t. } \forall (i, x) \in L(R), \)
\[ \exists \text{Sim}((i, x)) \exists \exists \langle P((i, x), w), V_{\text{on}}(pp_v, x) \rangle \mid pp_v \xleftarrow{\$} V_{\text{off}}(i) \]

Additionally, succinctness, public coin, knowledge soundness

Very hairy to define precisely so I will omit this in lecture.
How are ZK-SNARKs constructed in practice?

**Intuition:**
Today, we will focus on the index relation for Circuit SAT.

\[ C \rightarrow \tilde{C} \in F[x] \quad , \quad \text{com} \leftarrow \text{Commit}(\tilde{C}) \]

Instead of the verifier reading in the full description of the circuit \( C \) during the online phase, we can cleverly encode the circuit \( C \) as some polynomial \( \tilde{C} \in F[x] \).
Then, the verifier can commit to this polynomial in an offline (preprocessing) stage.

During the online phase, whenever the verifier \( V_{on}(\text{com}, x) \) wants to query the encoding, it asks the Prover for help. The prover will respond with an evaluation \( y \) and an evaluation proof \( \Pi \).

\[ P \quad \text{What's } \tilde{C}(z) ? \quad \text{V}(\text{com}, x) \]

\[ \leftarrow \quad Y, \quad \Pi_z \quad \rightarrow \quad \text{Verify}(\text{com}, z, y, \Pi_z) \]
Additionally, the prover can encode their witness $w$ as a polynomial $\bar{w}$ and send a commitment to it. Then, the verifier can query both $\bar{w}$ and $\bar{c}$ to test for various properties of these polynomials.

**Recipe**

A statistically sound proof system where in prover is unbounded in some idealized oracle model.

A computationally sound commit scheme to oracles.

Today, we will focus on the statistically sound portion of the construction. In particular, we will construct a PIOP for Circuit-SAT.
A Polynomial Interactive Oracle Proof (PIOP) system for an Index Relation $R$ over a field $F$ is a tuple of eff interactive algs $(P, V := (\text{Voff}, \text{Von}))$ where the Prover's messages are restricted to black box polynomials $\in F[x]$, which the verifier can query as oracles.

**Offline Stage:** $\tilde{\text{ppv}} \leftarrow \text{Voff}(i)$ outputs a set of blackbox polynomials.

**Online Stage:**

\[
P((i, x), w) \quad \text{Von}(\tilde{\text{ppv}}, x)
\]

For round $i$, generate a set of black box polys $\tilde{m}_i$,

\[
\begin{array}{c}
g \leftarrow r_i \quad \text{Sample Randomness} \\
\tilde{m}_i \leftarrow \text{Query}(\tilde{\text{ppv}}, x, r_i, x)
\end{array}
\]

After Interaction: Verifier queries polynomials $\tilde{\text{ppv}}$ and $\tilde{m}_i$ based on $r_i$ randomness sent.

\[
evals \leftarrow \text{Query}(\tilde{\text{ppv}}, \tilde{m}_i, r_i, x)
\]

Then, verifier outputs 0/1 from this info.

\[
0/1 \leftarrow \text{Decide}(\text{evals}, r_i, x)
\]

*This idealized protocol can be compiled to the standard model by sending polynomial commitments as eval proofs.*
Notions of completeness/soundness mirror that of Preprocessing SNARKs:

**Completeness:** \( \forall (i,x) \in L(R), \)
\[
\Pr \left[ \langle P((i,x), w), v_{on}(\widetilde{p}, x) \rangle = 1 \mid \widetilde{p} \in V_{off}(i) \right] \geq 1 - \text{neg}(\lambda)
\]

**Soundness:** \( \forall (i,x) \notin L(R), \forall P^* \) that send message polys whose \( \deg \leq \text{poly}(\lambda), \)
\[
\Pr \left[ \langle P^*, v_{on}(\widetilde{p}, x) \rangle = 1 \mid \widetilde{p} \in V_{off}(i) \right] \leq \text{neg}(\lambda)
\]

**Preliminaries**

Define \( \mathbb{F} \) to be a field of order \( p \) such that
- \( |p| \approx 2^\lambda \)
- \( 3 \cdot 2^\lambda \mid p-1 \) for \( \lambda \in \mathbb{N} \), \( 2^\lambda \approx \text{poly}(\lambda) \)

Since \( \mathbb{F}^* \) is cyclic, \( \exists \) a multiplicative subgroup \( H \subseteq \mathbb{F}^* \) whose order is \( n = 3 \cdot 2^\lambda \). Furthermore, \( \exists \) a generator \( g \in \mathbb{F}^* \) s.t.
\[
H = \{ g^0, g^1, g^2, \ldots, g^{n-1} \}
\]

We will refer to \( H \) as our \( \text{evaluation domain} \).

**Vanishing Poly:** \( H \) is the exact set of roots of \( X^{n-1} \in \mathbb{F}[X] \).

**Fact:** \( (X^{n-1}) \mid f(X) \iff f(X) \) vanishes on \( H \)
Consider a polynomial \( f = \sum_{i=0}^{n-1} f_i x^i \in \mathbb{F}[x] \) whose degree < n.

The Fast Fourier Transform is an alg running in \( O(n \log n) \) time that computes the evals of \( f \) over \( \mathbb{F} \) given coeffs \( f_i \):

\[
\text{FFT}(f) \rightarrow (f(1), f(2), \ldots, f(2^n - 1))
\]

Similarly, the inverse FFT interpolates coeffs of \( f \) from evals over \( \mathbb{F} \) in \( O(n \log n) \) time.

\[
\text{FFT}^{-1}((f(1), f(2), \ldots, f(2^n - 1)) \rightarrow f
\]

Fibonacci Example

Fibonacci Sequence

\( t_0 = 0 \), \( t_1 = 1 \), \( \forall j \geq 2 \), \( t_j := t_{j-2} + t_{j-1} \)

| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | ...
|-------|---|---|---|---|---|---|---|---
| term  | 0 | 1 | 1 | 2 | 3 | 5 | 8 | ...

\( R_{fib} := \exists ( (j, t_j), (t_0, t_1, \ldots, t_{j-1}) ) : (t_0, \ldots, t_j) \) are \( j \) terms of fibonacci

We will describe a PIOP for this trivial relation where the verifier does a constant number of queries and a small number of arithmetic operations.
Intuition
Prover could commit to its witness as a poly \( f \in F^n[x] \) such that \( \forall i \leq j, \ f(g^i) = t_i \) (assume \( j = n-1 \) for simplicity)

\[
\begin{array}{c|cccccccc}
\forall i \leq j : & t_0 & t_1 & t_2 & \ldots & \ldots & t_j \\
\forall h \in H : & f(g^0) & f(g^1) & f(g^2) & \ldots & \ldots & f(g^m) \\
\end{array}
\]

I.E., we encode the witness as evaluations over \( H \).

Then, the prover could send \( f \) to the verifier. The verifier could query \( f(g^{n-1}) \) to get the \( j \)-th value....

But how can the verifier be sure that \( f(x) \) actually encodes the correct values of the fibonacci sequence???

**Polynomial Property Testing!**

We want to test whether \( f \) encodes the Fib seq.

A first check: the verifier can query \( f(g^0)=0, f(g^1)=1 \).

Then, we would like to check: \( \forall h \in H \setminus \{g^0, g^1\}, \ f(g^{-2}h) + f(g^{-1}h) = f(h) \)

Rearranging,
\[
f(g^{-2}h) + f(g^{-1}h) - f(h) = 0
\]
Another way of viewing that, we want the polynomial
\[ f(g^{-2}X) + f(g^{-1}X) - f(X) \]
to vanish on \( H \setminus \{g^{0}, g^{1}\} \).

\[ \iff \quad F(X) := (f(g^{-2}X) + f(g^{-1}X) - f(X))(X - g^{0})(X - g^{1}) \]
vanishes on \( H \).

Using our earlier fact that
\[ x^{n-1} \mid F(X) \iff F(X) \text{ vanishes on } H \]

The prover can derive a quotient poly \( q(x) := F(x)/(x^{n-1}) \) and send this to the verifier.

Notice if \( F(X) \neq (x^{n-1})q(x) \), then \( F(X) - (x^{n-1})q(x) \neq 0 \).

Thus, the verifier can check
\[ F(X) \div (x^{n-1})q(x) \]

By querying \( q \) and \( f \) at a random point \( \alpha \in \mathbb{F} \).

\[ F(\alpha) \div (x^{n-1})q(\alpha) \]

\[ (f(g^{-2}\alpha) + f(g^{-1}\alpha) - f(\alpha))(\alpha - g^{0})(\alpha - g^{1}) \div (x^{n-1})q(\alpha) \]
\[\begin{align*}
\neg \quad \text{Interpolate } f & \leftarrow \text{FFT}^{-1}( f(g^i) = t; \forall i \leq j ) \\
\neg \quad \text{Define } F(X) & := (f(g^2x) + f(g^i x) - f(x))(x - g^0)(x - g^1) \\
\neg \quad \text{compute quotient poly } q(x) & := F(X)/(x^{n-1}) \in \mathbb{F}[x] \\
\neg \quad \text{Send } f, q \text{ to verifier} \\
\end{align*}\]

\[V(j, t_i) \]

- Query \( f \) at \( g^0, g^1, g^j \) to check \( f(g^0) = 0, f(g^1) = 1, f(g^j) = t_j \)
- Sample \( \alpha \in \mathbb{F} \), query \( f \) at \( g^2 \alpha, g^i \alpha, \alpha \) to compute \( F(\alpha) = (f(g^2 \alpha) + f(g^i \alpha) - f(\alpha))(\alpha - g^0)(\alpha - g^1) \)
- Query \( q \) at \( \alpha \) to check \( (\alpha^{n-1})q(\alpha) = F(\alpha) \)
- Output accept if all checks pass

**Completeness**

\( x^{n-1} \mid F(X) \) iff \( H \) is a set of roots of \( F \).

We constructed \( F(X) \) and \( f \) such that \( \forall h \in H \setminus \{g^0, g^1\}, f(g^2h) + f(g^i h) - f(h) = 0 \)

For \( h \in \{g^0, g^1\}, (x - g^0)(x - g^1) = 0 \)

Thus, \( F \) vanishes on \( H \).
Soundness:
Say a malicious prover $P^*$ claims that $t_j = c$ (where $c$ is not the $j$th fib)
They send polys $f, g \in \mathbb{F}[X]$ such that $\deg(f), \deg(g) \leq \text{poly}(\lambda)$.

Let us denote the event $E_0$ as the event where

$$ f(g^0) = 0, \ f(g^1) = 1, \ f(g^j) = c $$

and denote $E_1$ as the event where $(x^{n-1})q(x) = F(x)$.

$$ \Pr[V\text{accepts}] = \Pr[V\text{accept}\mid E_0] \Pr[E_0] + \Pr[V\text{accept}\mid \neg E_0] \Pr[\neg E_0] $$

$$ \leq \Pr[V\text{accept}\mid E_0] + 0 $$

$$ = \Pr[V\text{accept}\mid E_0 \land E_1] \Pr[E_1\mid E_0] + \Pr[V\text{accept}\mid E_0 \land \neg E_1] \Pr[\neg E_1\mid E_0] $$

$$ \leq \Pr[E_1\mid E_0] + \Pr[V\text{accept}\mid E_0 \land \neg E_1] $$

$\Pr[V\text{accept}\mid E_0 \land \neg E_1]$:

If $(x^{n-1})q(x) \neq F(x)$, then $(x^{n-1})q(x) - F(x) \neq 0$. Thus,

$$ \Pr[(x^{n-1})q(x) - F(x) = 0 \mid x \in \mathbb{F}] \leq \frac{\deg(Q)}{|\mathbb{F}|} = \frac{\text{poly}(\lambda)}{|\mathbb{F}|} \leq \text{neg}(\lambda) $$

$Q$ can have at most $\deg(Q)$ roots over $\mathbb{F}$.

Thus, $\Pr[V\text{accept}\mid E_0 \land \neg E_1] \leq \text{neg}(\lambda)$.

$\Pr[E_1\mid E_0]$:

We will argue $E_0 \land E_1$ cannot occur $\Rightarrow \Pr[E_1\mid E_0] = 0$.
Lemma: If $\exists f, q \in \mathbb{F}[x], s.t. (x^n-1)q(x) = F(x)$ and $f(g^0) = 0, f(g^1) = 1, f(g^j) = c$ then $t_j = c$.

$(x^n - 1)q(x) = F(x) \iff F$ vanishes on $H$

$\iff \forall h \in H, (f(g^0) + f(g^1) - f(h))(x-g^0)(x-g^1) = 0$

\[
\begin{align*}
    f(g^0) &= 0, f(g^1) = 1, f(g^j) = c \\
    \downarrow \\
    \text{By induction on indices, } t_j = c.
\end{align*}
\]

By contrapositive of lemma, $\not\exists f, g$ s.t. $E_0 \land E_1$ occurs. Thus, $\Pr[E_1 | E_0] = 0$.

Therefore $\Pr[\text{Vac}] \leq 0 + \neg\neg\neg c(\lambda) \leq \neg\neg c(\lambda)$.

Plonk (GWC19)

- SNARK for Circuit-SAT
- Clever encoding of ACs
Consider the following Arithmetic circuit $C$, run on an instance $x := (x_0, x_1) \in \mathbb{F}^2$ and witness $w := (w_0, w_1) \in \mathbb{F}^2$.

![Arithmetic Circuit Diagram]

where $V_0, \ldots, V_8$ are the wire values. We call the vector $W := (V_0, V_1, \ldots, V_8)$ the extended witness.

For $W$ to be a valid assignment of the wires, the following must hold:

1) Gates Checks:
   - $V_0 \leftarrow V_1 \leftarrow V_2 \leftarrow V_3 \leftarrow V_4 \leftarrow V_5 \leftarrow V_6 \leftarrow V_7 \leftarrow V_8$
   - $+ = \rightarrow$
   - $+ = \rightarrow$
   - $x = \rightarrow$

2) Copy Checks
   - $V_0 \leftarrow V_1 \leftarrow V_2 \leftarrow V_3 \leftarrow V_4 \leftarrow V_5 \leftarrow V_6 \leftarrow V_7 \leftarrow V_8$
   - $= \$


We can define two vectors based on C:

- Gate selector: \((S_0 := 0, S_1 := 0, S_2 := 1)\) where
  \(S_i := 0\) if Gate \(i\) is an addition gate and \(1\) o/w

- Permutation Vector: \((0, 1, 6, 3, 4, 7, 2, 5, 8)\)

\[ \Pi \uparrow \uparrow \uparrow \uparrow \uparrow \uparrow \uparrow \uparrow \uparrow \uparrow \]
\[ (0, 1, 2, 3, 4, 5, 6, 7, 8) \]

that encodes a perm \(\Pi\).

Notice checks 1) and 2) are equivalent to checking

1) \(\forall i \leq 3, j := 3i \)
\[(1 - S_i)(V_j + V_{j+1}) + S_i (V_i \cdot V_{j+1}) = V_{j+2}\]

2) \(\forall j \leq 9, V_j = V_{\Pi(j)}\)

We will translate these checks into statements about polys!

To encode circuit \(C\) as a polynomial, interpolate a poly \(S\) and \(\Pi\) (assuming \(|H| = 9, |H'| \leq H, H' = \ell g^0, g^3, g^6\))

\[
\begin{array}{c|ccc}
S & g^0 & g^3 & g^6 \\
----&---&---&--- \\
0 & 0 & 0 & 1 \\
\end{array}
\quad
\begin{array}{c|cccccccc}
\Pi & g^0 & g^1 & g^2 & g^3 & g^4 & g^5 & g^6 & g^7 & g^8 & g^9 \\
----&---&---&---&---&---&---&---&---&---&--- \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

These polynomials will be outputted by Voss in preprocessing. The prover can interpolate a poly \(W\) s.t.

\[
\begin{array}{c|cccccccc}
W & g^0 & g^1 & g^2 & g^3 & g^4 & g^5 & g^6 & g^7 & g^8 & g^9 \\
----&---&---&---&---&---&---&---&---&---&--- \\
V_0 & V_1 & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & V_9 \\
\end{array}
\]

\* normally we would pad \(\Pi\) s.t. \(|H'| = 2^e\) for some \(e\) but lets ignore that.
The prover sends $W$ to the online verifier $V_{on}$ who has oracle access to $S(X)$ and $T\mid(X)$.

To check perform the gate check, the verifier would like to check

\[ \forall h \in H', (1 - S(h)) (W(h) + W(gh)) + S(h) (W(h) W(gh)) = W(g^2 h). \]

This is equivalent to checking if the polynomial

\[ (1 - S(X)) (W(X) + W(g^{-1} X)) + S(X) (W(X) W(gX)) - W(g^2 X) \]

vanishes on $H'$.

To check the copy constraints is more complicated:

Required efficiently testing that:

\[ \forall h \in H, \quad W(h) = W(T\mid(h)). \]

Using our old strategy to test if

\[ W(X) = W(T\mid(X)) \]

vanishes on $H$ would require the prover compute a quotient $q(X)$ whose $\deg \geq n^2$ (quadratic).

Turns out with clever randomized tests, it's possible to check this condition without a quadratic blow-up.