Building a SNARK

Dan Boneh
Recap: zk-SNARK applications

Private Tx on a public blockchain: Zcash, IronFish

Compliance:
- Proving that a private Tx are in compliance with banking laws
- Proving solvency in zero-knowledge

Scalability: privacy in a zk-SNARK Rollup (next week)

Bridging between blockchains: zkBridge
NARK: Non-interactive ARgument of Knowledge

Public arithmetic circuit: \( C( x, w ) \rightarrow \mathbb{F} \)

- Public statement in \( \mathbb{F}^n \)
- Secret witness in \( \mathbb{F}^m \)

Preprocessing (setup): \( S(C) \rightarrow \) public parameters \((pp, vp)\)

\[ pp, x, w \]

\[ \text{proof } \pi \text{ that } C(x, w) = 0 \]

\[ \text{Verifier} \rightarrow \text{accept or reject} \]
NARK: requirements (informal)

Prover $P(pp, x, w)$  
Verifier $V(vp, x, \pi)$

Proof $\pi$  
accept or reject

Complete: $\forall x, w: C(x, w) = 0 \implies \Pr[V(vp, x, P(pp, x, w)) = \text{accept}] = 1$

Adaptively knowledge sound: $V$ accepts $\implies P$ “knows” $w$ s.t. $C(x, w) = 0$  
(an extractor $E$ can extract a valid $w$ from $P$)

Optional: Zero knowledge: $(C, pp, vp, x, \pi)$ “reveal nothing new” about $w$  
(witness exists $\implies$ can simulate the proof)
A succinct preprocessing NARK is a triple \((S, P, V)\):

- **S**\((C)\) → public parameters \((pp, vp)\) for prover and verifier

- **P**\((pp, x, w)\) → short proof \(\pi\);  
  \(\text{len}(\pi) = O_\lambda(\text{polylog}(|C|))\)

- **V**\((vp, x, \pi)\) fast to verify;  
  \(\text{time}(V) = O_\lambda(|x|, \text{polylog}(|C|))\)

short “summary” of circuit
A simple PCP-based SNARK

[Kilian’92, Micali’94]
The PCP theorem: Let \( C(x, w) \) be an arithmetic circuit. there is a proof system that for every \( x \) proves \( \exists w: C(x, w) = 0 \) as follows:

Prover \( P(pp, x, w) \)

Verifier \( V(vp, x) \)

long proof \( \pi \)

read only \( O(\lambda) \) bits of \( \pi \), output accept or reject

V always accepts valid proof. If no \( w \), then V rejects with high prob.

size of proof \( \pi \) is \( poly(|C|) \). (not succinct)
Converting a PCP proof to a SNARK

Prover $P(pp, x, w)$

Verifier $V(vp, x)$

Merkle root $h$

open $k$ positions of $\pi$  ($k = O(\lambda)$)

$k$ opening and Merkle proofs

1 hash

$O(k \log |C|)$ hashes

output accept or reject

Verifier sees $O(\lambda \log |C|)$ data $\Rightarrow$ succinct proof.

Problem: interactive
Making the proof non-interactive

The **Fiat-Shamir transform**:  
• public-coin interactive protocol ⇒ non-interactive protocol  
  public coin: all verifier randomness is public (no secrets)

**Prover** $P(pp, x, w)$  
  
<table>
<thead>
<tr>
<th>msg1</th>
<th>Verifier $V(vp, x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r$</td>
<td>choose random bits $r$</td>
</tr>
<tr>
<td>$msg2$</td>
<td>accept or reject</td>
</tr>
</tbody>
</table>
Making the proof non-interactive

**Fiat-Shamir transform:** \( H: M \rightarrow R \) a cryptographic hash function

- idea: prover generates random bits on its own (!)

Prover \( P(pp, x, w) \)
- generate msg1
- \( r \leftarrow H(x, \text{msg1}) \)
- generate msg2

Verifier \( V(vp, x) \)
- \( r \leftarrow H(x, \text{msg1}) \)
- accept or reject

\[ |\pi| = O(\lambda \log |C|) \]

Fiat-Shamir: certain secure interactive protocols \( \Rightarrow \) non-interactive
Are we done?

Simple transparent SNARK from the PCP theorem
• Use Fiat-Shamir transform to make non-interactive
• We will apply Fiat-Shamir in many other settings

The bad news: an impractical SNARK --- Prover time too high

Better SNARKs: Goal: $\text{Time}($Prover$) = \tilde{O}(|C|)$
Building an efficient SNARK
General paradigm: two steps

1. A polynomial commitment scheme (cryptographic object)
2. A polynomial interactive oracle proof (Poly-IOP) (info. theoretic object)

Let’s explain each concept ...

SNARK for general circuits
Recall: commitments

Two algorithms:

- $commit(m, r) \rightarrow \text{com}$  \hspace{1cm} ($r$ chose at random)
- $verify(m, \text{com}, r) \rightarrow$ accept or reject

Properties:

- **binding**: cannot produce two valid openings for $\text{com}$
- **hiding**: $\text{com}$ reveals nothing about committed data
(1) Polynomial commitment scheme (PCS)

Notation: \( \mathbb{F}_p^{(\leq d)}[X] \) is all polynomials in \( \mathbb{F}_p[X] \) of degree \( \leq d \).

Prover commits to a polynomial \( f(X) \) in \( \mathbb{F}_p^{(\leq d)}[X] \) (univariate)

- **eval**: for public \( u, v \in \mathbb{F}_p \), prover can convince the verifier that committed poly satisfies

\[
  f(u) = v \quad \text{and} \quad \deg(f) \leq d.
\]

verifier has \( (d, \text{com}_f, u, v) \)

- Eval proof size and verifier time should be \( O_\lambda(\log d) \)
(1) Polynomial commitment scheme (PCS)

- **setup**\((d) \rightarrow pp\), public parameters for polynomials of degree \( \leq d \)
- **commit**\((pp, f, r) \rightarrow com_f\) commitment to \( f \in \mathbb{F}_p^{(\leq d)} [X] \)
- **eval**: goal: for a given \( com_f \) and \( x, y \in \mathbb{F}_p \), prove that \( f(x) = y \).

Formally: \( eval = (s, P, V) \) is a SNARK for:

\[
\text{statement } st = (pp, com_f, x, y) \text{ with witness } w = (f, r)
\]

where \( C(st, w) = 0 \) iff

\[
[ f(x) = y \text{ and } f \in \mathbb{F}_p^{(\leq d)} [X] \text{ and } \text{commit}(pp, f, r) = com_f ]
\]
(1) Polynomial commitment scheme (PCS)

Properties:

• Binding: cannot produce two valid openings \((f_1, r_1), (f_2, r_2)\) for \(\text{com}_f\).

• eval is knowledge sound (can extract \((f, r)\) from a successful prover)

• optional:

  • commitment is hiding
  • eval is zero knowledge

Note: poly. commitments have many applications beyond SNARKs
Constructing a PCS

Not today ... (see readings or CS355)

Properties of the most widely used in practice (called KZG):

• trusted setup: secret randomness in setup.  \( |pp| = O_\lambda(d) \)

• \(com_f\): constant size (one group element)

• eval proof size: constant size (one group element)

• eval verify time: constant time.  Prover time: \( O_\lambda(d) \)
General paradigm: two steps

(1) A polynomial commitment scheme (cryptographic object)

(2) A polynomial interactive oracle proof (Poly-IOP) (info. theoretic object)

SNARK for general circuits

What is a Poly-IOP?
Let \( C(x, w) \) be some arithmetic circuit. Let \( x \in \mathbb{F}_p^n \).

**Poly-IOP**: a proof system that proves \( \exists w: C(x, w) = 0 \) as follows:

\[
\text{Setup}(C) \rightarrow \text{public parameters } pp \text{ and } vp = (f_0, f_{-1}, \ldots, f_{-s})
\]
Polynomial IOP

Prover $P(pp,x,w)$

Verifier $V(f_0, ..., f_{-s}, x)$

$r_1 \leftarrow \mathbb{F}_p$

commit

$f_1 \in \mathbb{F}_p^{(\leq d)} [X]$

\vdots

commit

$f_t \in \mathbb{F}_p^{(\leq d)} [X]$

query $f_{-s}, ..., f_t$ at a few points

send values (and eval proofs)

$\rightarrow$ accept/reject
**Goal:** construct a poly-IOP called *Plonk* (eprint/2019/953) 

[Gabizon – Williamson – Ciobotaru]

\[ \text{Plonk} + \text{PCS} \Rightarrow \text{SNARK} \]

(and also a zk-SNARK)

[ PCS = Polynomial Commitment Scheme]
Proving properties of committed polynomials

Goal: succinct proofs
Proving properties of committed polynomials

Prover P(f, g)

Goal: convince verifier that \( f, g \in \mathbb{F}_p^{(\leq d)} [X] \) satisfy some properties

Verifier V(f, g)

Proof systems presented as a Poly-IOP:

- Prover:
  - Sends \( r \) from \( \mathbb{F}_p \)
  - Sends \( q \) (a commitment to some poly. \( q \))

- Verifier:
  - Receives \( f(X), g(X), q(X) \) at some points in \( \mathbb{F}_p \)
  - Accept or reject
A simple example: polynomial equality testing

Goal: convince verifier that $f = g$

Prover

$$f, g \in \mathbb{F}_p^{(\leq d)}[X]$$

Verifier

query $f(X)$ and $g(X)$ at $r$

$$r \leftarrow \mathbb{F}_p$$

learn $f(r), g(r)$

accept if:

$$f(r) = g(r)$$

Why is this sound?
A key fact: for non-zero \( f \in \mathbb{F}_p^{(\leq d)} [X] \)

\[
\text{for } r \leftarrow \mathbb{F}_p : \quad \Pr[f(r) = 0] \leq \frac{d}{p} \tag{\star}
\]

\( \Rightarrow \) suppose \( p \approx 2^{256} \) and \( d \leq 2^{40} \) then \( d/p \) is negligible

\( \Rightarrow \) for \( r \leftarrow \mathbb{F}_p \): if \( f(r) = 0 \) then \( f \) is identically zero w.h.p

\( \Rightarrow \) a simple test if a committed poly. is the zero poly.

**SZDL lemma:** \( \star \) also holds for multivariate polynomials (where \( d \) is total degree of \( f \))
Why is this sound?

Suppose $p \approx 2^{256}$ and $d \leq 2^{40}$ so that $d/p$ is negligible.

Let $f, g \in \mathbb{F}_p^{(\leq d)}[X]$. For $r \leftarrow \mathbb{F}_p$, if $f(r) = g(r)$ then $f = g$ w.h.p.

$$f(r) - g(r) = 0 \implies f - g = 0 \text{ w.h.p}$$

⇒ a simple equality test for two committed polynomials
The polynomial equality testing protocol

**Goal:** convince verifier that $f = g$

- **Prover**
  - $f, g \in \mathbb{F}_p^{(\leq d)}[X]$
  - query $f(X)$ and $g(X)$ at $r$

- **Verifier**
  - $r \leftarrow \mathbb{F}_p$
  - learn $f(r), g(r)$
  - accept if: $f(r) = g(r)$

**Lemma:** complete and sound assuming $d/p$ is negligible
Review: the compiled proof system

Prover

\[ f, g \in \mathbb{F}_p^{(\leq d)}[X] \]

\[ y \leftarrow f(r) \]
\[ y' \leftarrow g(r) \]

Make non-interactive using Fiat-Shamir

\[ r \]

Verifier

\[ r \leftarrow \mathbb{F}_p \]

learn \( f(r), g(r) \)

accept if:

(i) \( y = y' \) and
(ii) \( \pi_f, \pi_g \) are valid

proof that

\[ y = f(r) \]

proof that

\[ y' = g(r) \]
Important proof gadgets for univariates

Let $\Omega$ be some subset of $\mathbb{F}_p$ of size $k$.

Let $f \in \mathbb{F}_p^{(\leq d)} [X]$ \hspace{1cm} (d $\geq$ $k$) \hspace{1cm} Verifier has $\fbox{f}$

Let us construct efficient Poly-IOPs for the following tasks:

Task 1 (ZeroTest): prove that $f$ is identically zero on $\Omega$

Task 2 (SumCheck): prove that $\sum_{a \in \Omega} f(a) = 0$

Task 3 (ProdCheck): prove that $\prod_{a \in \Omega} f(a) = 1$
The vanishing polynomial

Let $\Omega$ be some subset of $\mathbb{F}_p$ of size $k$.

**Def:** the **vanishing polynomial** of $\Omega$ is $Z_\Omega(X) := \prod_{a \in \Omega}(X - a)$

\[
\deg(Z_\Omega) = k
\]

Let $\omega \in \mathbb{F}_p$ be a primitive $k$-th root of unity (so that $\omega^k = 1$).

- if $\Omega = \{ 1, \omega, \omega^2, ..., \omega^{k-1} \} \subseteq \mathbb{F}_p$ then $Z_\Omega(X) = X^k - 1$

$\Rightarrow$ for $r \in \mathbb{F}_p$, evaluating $Z_\Omega(r)$ takes $2 \log_2 k$ field operations
(1) ZeroTest on $\Omega$ ($\Omega = \{1, \omega, \omega^2, ..., \omega^{k-1}\}$)

**Prover** $P(f)$

$q(X) \leftarrow f(X)/Z_\Omega(X)$

Query $q(X)$ and $f(X)$ at $r$

**Verifier** $V(\square)$

$r \leftarrow \mathbb{F}_p$ (verifier evaluates $Z_\Omega(r)$ by itself)

Learn $q(r), f(r)$

Accept if $f(r) \cong q(r) \cdot Z_\Omega(r)$

(implies that $f(X) = q(X) \cdot Z_\Omega(X)$ w.h.p)

**Lemma:** $f$ is zero on $\Omega$ if and only if $f(X)$ is divisible by $Z_\Omega(X)$

**Thm:** this protocol is complete and sound, assuming $d/p$ is negligible.
(1) ZeroTest on $\Omega$ $(\Omega = \{1, \omega, \omega^2, \ldots, \omega^{k-1}\})$

**Prover** $P(f)$

$q(X) \leftarrow f(X)/Z_{\Omega}(X)$

**Verifier** $V(f)$

$r \leftarrow \mathbb{F}_p$ verifier evaluates $Z_{\Omega}(r)$ by itself

query $q(X)$ and $f(X)$ at $r$

learn $q(r), f(r)$

accept if $f(r) \equiv q(r) \cdot Z_{\Omega}(r)$

(implies that $f(X) = q(X) \cdot Z_{\Omega}(X)$ w.h.p)

**Lemma:** $f$ is zero on $\Omega$ if and only if $f(X)$ is divisible by $Z_{\Omega}(X)$

**Verifier time:** $O(\log k)$ and two poly queries (but can be batched)

**Prover time:** dominated by time to compute $q(X)$ and commit to $q(X)$
(4) Another useful gadget: permutation check

Let \( f, g \) polynomials in \( \mathbb{F}_p^{(\leq d)} [X] \). Verifier has \( f, g \).

Prover wants to prove that

\[
\begin{pmatrix} f(1), f(\omega), f(\omega^2), \ldots, f(\omega^{k-1}) \end{pmatrix} \in \mathbb{F}_p^k
\]

is a permutation of

\[
\begin{pmatrix} g(1), g(\omega), g(\omega^2), \ldots, g(\omega^{k-1}) \end{pmatrix} \in \mathbb{F}_p^k
\]

\[\Rightarrow\] Proves that \( g(\Omega) \) is the same as \( f(\Omega) \), just permuted
(4) Another useful gadget: permutation check

Prover $P(f, g)$

Let $\hat{f}(X) = \prod_{a \in \Omega} (X - f(a))$ and $\hat{g}(X) = \prod_{a \in \Omega} (X - g(a))$

Then: $\hat{f}(X) = \hat{g}(X) \iff g(\Omega)$ is a permutation of $f(\Omega)$

prove that $\hat{f}(r) = \hat{g}(r)$

prod-check: $\frac{\hat{f}(r)}{\hat{g}(r)} = \prod_{a \in \Omega} \left( \frac{r - f(a)}{r - g(a)} \right) = 1$

implies $\hat{f}(X) = \hat{g}(X)$ w.h.p
accept or reject

Verifier $V(f, g)$

$Lipton's$ $trick$, $1989$
(5) final gadget: prescribed permutation check

\[ W: \Omega \rightarrow \Omega \] is a **permutation** of \( \Omega \) if \( \forall i \in [k]: W(\omega^i) = \omega^j \) a bijection

example \((k = 3)\): \( W(\omega^0) = \omega^2 \), \( W(\omega^1) = \omega^0 \), \( W(\omega^2) = \omega^1 \)

Let \( f, g \) polynomials in \( \mathbb{F}_p^{(\leq d)}[X] \). Verifier has \( f \), \( g \), \( W \).

**Goal**: prover wants to prove that \( f(y) = g(W(y)) \) for all \( y \in \Omega \)

\[ \Rightarrow \] Proves that \( g(\Omega) \) is the same as \( f(\Omega) \), permuted by the prescribed \( W \)
Prescribed permutation check

How? Use a zero-test to prove $f(y) - g(W(y)) = 0$ on $\Omega$

The problem: the polynomial $f(y) - g(W(y))$ has degree $k^2$

⇒ prover would need to manipulate polynomials of degree $k^2$

⇒ quadratic time prover !! (goal: linear time prover)

Can reduce this to a prod-check on a poly of degree $2k$ (not $k^2$)
Summary of proof gadgets

- polynomial equality testing
- zero test on $\Omega$
- product check, sum check
- permutation check
- prescribed permutation check check
The PLONK Poly-IOP for general circuits
eprint/2019/953
PLONK: widely used in practice

The Plonk Poly-IOP

polynomial commitment scheme

KZG’10 (pairings)

Bulletproofs (no pairings)

FRI (hashing)

SNARK system

Aztec, JellyFish

Halo2 (slow verifier) (no trusted setup)

Plonky2, Redshift (no trusted setup)
PLONK: a poly-IOP for a general circuit \( C(x, w) \)

**Step 1:** compile circuit to a computation trace (gate fan-in = 2)

The computation trace (arithmetization):

\[
(x_1 + x_2)(x_2 + w_1)
\]

Inputs: 5, 6, 1

Gate 0: 5, 6, 11
Gate 1: 6, 1, 7
Gate 2: 11, 7, 77

Example input:

\[
\begin{array}{c}
5 \\
6 \\
1 \\
\end{array}
\]

Left inputs: 5, 6, 1
Right inputs: 5, 6, 11
Outputs: 77
Encoding the trace as a polynomial

|C| := total # of gates in C, \quad |I| := |I_x| + |I_w| = # inputs to C

let \( d := 3 \, |C| + |I| \) (in example, \( d = 12 \)) and \( \Omega := \{ 1, \omega, \omega^2, \ldots, \omega^{d-1} \} \)

The plan:
prover interpolates a poly. \( T \in \mathbb{F}_p^{(\leq d)}[X] \)
that encodes the entire trace.

Let’s see how ...

inputs: \( 5, 6, 1 \)

Gate 0: \( 5, 6, 11 \)
Gate 1: \( 6, 1, 7 \)
Gate 2: \( 11, 7, 77 \)
Encoding the trace as a polynomial

The plan: Prover interpolates $T \in \mathbb{F}_p^{(\leq d)}[X]$ such that

1. **$T$ encodes all inputs:** $T(\omega^{-j}) = \text{input } \#j$ for $j = 1, \ldots, |I|$

2. **$T$ encodes all wires:** $\forall \ l = 0, \ldots, |C| - 1$:
   - $T(\omega^{3l})$: left input to gate $\#l$
   - $T(\omega^{3l+1})$: right input to gate $\#l$
   - $T(\omega^{3l+2})$: output of gate $\#l$

<table>
<thead>
<tr>
<th>inputs:</th>
<th>5, 6, 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gate 0:</td>
<td>5, 6, 11</td>
</tr>
<tr>
<td>Gate 1:</td>
<td>6, 1, 7</td>
</tr>
<tr>
<td>Gate 2:</td>
<td>11, 7, 77</td>
</tr>
</tbody>
</table>
Encoding the trace as a polynomial

In our example, Prover interpolates $T(X)$ such that:

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Gate 0</th>
<th>Gate 1</th>
<th>Gate 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T(\omega^{-1}) = 5$, $T(\omega^{-2}) = 6$, $T(\omega^{-3}) = 1$</td>
<td>$T(\omega^0) = 5$, $T(\omega^1) = 6$, $T(\omega^2) = 11$</td>
<td>$T(\omega^3) = 6$, $T(\omega^4) = 1$, $T(\omega^5) = 7$</td>
<td>$T(\omega^6) = 11$, $T(\omega^7) = 7$, $T(\omega^8) = 77$</td>
</tr>
</tbody>
</table>

Prover can use FFT to compute the coefficients of $T$ in time $O(d \log d)$
Step 2: proving validity of $T$

Prover $P(S_p, x, w)$

Verifier $V(S_v, x)$

build $T(X) \in \mathbb{F}_p^{(\leq d)} [X]$

Prover needs to prove that $T$ is a correct computation trace:

\begin{itemize}
  \item (1) $T$ encodes the correct inputs,
  \item (2) every gate is evaluated correctly,
  \item (3) the wiring is implemented correctly,
  \item (4) the output of last gate is 0
\end{itemize}

Proving (4) is easy: prove $T(\omega^{3|c|-1}) = 0$

(wiring constraints)

<table>
<thead>
<tr>
<th>inputs</th>
<th>5, 6, 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gate 0</td>
<td>5, 6, 11</td>
</tr>
<tr>
<td>Gate 1</td>
<td>6, 1, 7</td>
</tr>
<tr>
<td>Gate 2</td>
<td>11, 7, 77</td>
</tr>
</tbody>
</table>
Proving (1): T encodes the correct inputs

Both prover and verifier interpolate a polynomial \( v(X) \in \mathbb{F}_p^{(\leq |I_x|)} [X] \) that encodes the \( x \)-inputs to the circuit:

\[
\text{for } j = 1, \ldots, |I_x|: \quad v(\omega^{-j}) = \text{input \#j}
\]

In our example: \( v(\omega^{-1}) = 5, \quad v(\omega^{-2}) = 6 \). \( v \) is linear.

constructing \( v(X) \) takes time proportional to the size of input \( x \)

\( \Rightarrow \) verifier has time to do this
Both prover and verifier interpolate a polynomial \( v(X) \in \mathbb{F}_p^{(\leq |I_x|)} [X] \) that encodes the \( x \)-inputs to the circuit:

\[
\text{for } j = 1, \ldots, |I_x|: \quad v(\omega^{-j}) = \text{input } \#j
\]

Let \( \Omega_{\text{inp}} := \{ \omega^{-1}, \omega^{-2}, \ldots, \omega^{-|I_x|} \} \subseteq \Omega \) (points encoding the input)

Prover proves (1) by using a ZeroTest on \( \Omega_{\text{inp}} \) to prove that

\[
T(y) - v(y) = 0 \quad \forall \ y \in \Omega_{\text{inp}}
\]
Proving (2): every gate is evaluated correctly

Idea: encode gate types using a **selector** polynomial $S(X)$

Define $S(X) \in \mathbb{F}_p^{(\leq d)}[X]$ such that $\forall \ l = 0, \ldots, |C| - 1$:

- $S(\omega^{3l}) = 1$ if gate $\#l$ is an addition gate
- $S(\omega^{3l}) = 0$ if gate $\#l$ is a multiplication gate

<table>
<thead>
<tr>
<th>inputs:</th>
<th>5, 6, 1</th>
<th>$S(X)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gate 0 ($\omega^0$):</td>
<td>5, 6, 11</td>
<td>1 (+)</td>
</tr>
<tr>
<td>Gate 1 ($\omega^3$):</td>
<td>6, 1, 7</td>
<td>1 (+)</td>
</tr>
<tr>
<td>Gate 2 ($\omega^6$):</td>
<td>11, 7, 77</td>
<td>0 (×)</td>
</tr>
</tbody>
</table>
Idea: encode gate types using a selector polynomial $S(X)$

\[
S(X) \in \mathbb{F}_p^{(≤d)}[X] \text{ such that } \forall l = 0, ..., |C| - 1:
\]
\[
S(\omega^{3l}) = 1 \text{ if gate } #l \text{ is an addition gate}
\]
\[
S(\omega^{3l}) = 0 \text{ if gate } #l \text{ is a multiplication gate}
\]

Then $\forall y \in \Omega_{\text{gates}} := \{1, \omega^3, \omega^6, \omega^9, ..., \omega^3(|C|-1)\}$:

\[
S(y) \cdot [T(y) + T(\omega y)] + (1 - S(y)) \cdot T(y) \cdot T(\omega y) = T(\omega^2 y)
\]
Proving (2): every gate is evaluated correctly

\[ S(y) \cdot [T(y) + T(\omega y)] + (1 - S(y)) \cdot T(y) \cdot T(\omega y) - T(\omega^2 y) = 0 \]

Prover uses ZeroTest to prove that for all \( \forall \ y \in \Omega_{gates} \):

Setup(\( C \)) \rightarrow \( pp := S \) and \( \nu p := (S) \)
Proving (3): the wiring is correct

**Step 4:** encode the wires of $C$:

\[
\begin{align*}
T(\omega^{-2}) &= T(\omega^1) = T(\omega^3) \\
T(\omega^{-1}) &= T(\omega^0) \\
T(\omega^2) &= T(\omega^6) \\
T(\omega^{-3}) &= T(\omega^4)
\end{align*}
\]

**Example:** $x_1=5, \ x_2=6, \ w_1=1$

\[
\begin{align*}
\omega^{-1}, \ \omega^{-2}, \ \omega^{-3}: & \quad 5, \ 6, \ 1 \\
0: \ \omega^0, \ \omega^1, \ \omega^2: & \quad 5, \ 6, \ 11 \\
1: \ \omega^3, \ \omega^4, \ \omega^5: & \quad 6, \ 1, \ 7 \\
2: \ \omega^6, \ \omega^7, \ \omega^8: & \quad 11, \ 7, \ 77
\end{align*}
\]

Define a polynomial $W: \Omega \rightarrow \Omega$ that implements a rotation:

\[ W(\omega^{-2}, \omega^1, \omega^3) = (\omega^1, \ \omega^3, \ \omega^{-2}) \quad , \quad W(\omega^{-1}, \omega^0) = (\omega^0, \ \omega^{-1}) \quad , \quad \ldots \]

**Lemma:** $\forall \ y \in \Omega: T(y) = T(W(y)) \ \Rightarrow \ \text{wire constraints are satisfied}$
Proving (3): the wiring is correct

Step 4: encode the wires of $C$:

\[ T(\omega^{-2}) = T(\omega^1) = T(\omega^3) \]
\[ T(\omega^{-1}) = T(\omega^0) \]
\[ T(\omega^2) = T(\omega^6) \]

Define a polynomial $W : \Omega \rightarrow \Omega$ that implements a rotation:

\[ W(\omega^{-2}, \omega^1, \omega^3) = (\omega^1, \omega^3, \omega^{-2}) \]
\[ W(\omega^{-1}, \omega^0) = (\omega^0, \omega^{-1}) \], ...

Lemma: $\forall y \in \Omega: T(y) = T(W(y)) \Rightarrow$ wire constraints are satisfied
The complete Plonk Poly-IOP (and SNARK)

Setup($C$) $\rightarrow$ $pp := (S,W)$ and $vp := (\boxed{S}$ and $\boxed{W})$ (untrusted)

Prover $P(pp, x, w)$

- build $T(X) \in \mathbb{F}_p^{(\leq d)}[X]$

Verifier $V(vp, x)$

- build $v(X) \in \mathbb{F}_p^{(\leq |I_x|)}[X]$

Prover proves:

- gates: (1) $S(y) \cdot [T(y) + T(\omega y)] + (1 - S(y)) \cdot T(y) \cdot T(\omega y) - T(\omega^2 y) = 0$; $\forall y \in \Omega_{\text{gates}}$
- inputs: (2) $T(y) - v(y) = 0$ $\forall y \in \Omega_{\text{inp}}$
- wires: (3) $T(y) - T(W(y)) = 0$ (using prescribed perm. check) $\forall y \in \Omega$
- output: (4) $T(\omega^{3|C|^{-1}}) = 0$ (output of last gate = 0)
The complete Plonk Poly-IOP (and SNARK)

Setup($C$) $\rightarrow$ $pp := (S, W)$ and $vp := (\boxed{S}$ and $\boxed{W})$ (untrusted)

**Prover** $P(pp, x, w)$
- build $T(X) \in \mathbb{F}_p^{(\leq d)}[X]

**Verifier** $V(vp, x)$
- build $v(X) \in \mathbb{F}_p^{(\leq |x|)}[X]

**Thm:** The Plonk Poly-IOP is complete and knowledge sound, assuming $7|C|/p$ is negligible

(eprint/2019/953)
Many extensions ...

• Plonk proof: a short proof (O(1) commitments), fast verifier
• The SNARK can be made into a zk-SNARK

Main challenge: reduce prover time

• **Hyperplonk**: replace $\Omega$ with $\{0,1\}^t$ (where $t = \log_2|\Omega|$)
  • The polynomial $T$ is now a multilinear polynomial in $t$ variables
  • ZeroTest is replaced by a multilinear SumCheck (linear time)
Next lecture: scaling the blockchain