SNARKs Lecture 4: Linear PCPs & Preprocessing SNARKs

Instructor: Ben Fisch
• PCP Theorems
  - Theorems proving statements with a specially formed long proof that verifier checks by reading only constant number of (random) locations
  - Can be done for any NP statement in theory
Recap: PCP

- $L(C) \rightarrow (\Sigma_p = \Sigma, \Sigma_v)$

- If $C(x, w) = 0$ then $V$ always accepts

- If $\exists w: C(x, w) = 0$ then $\Pr[V \text{ accepts}] < \left(\frac{1}{2}\right)^\lambda$ (is negl.)
Recap: Interactive Oracle Proofs

• **Oracle Proofs (PCPs)**
  - A PCP is a one-round oracle proof, where the prover sends a poly sized string $\pi$, and Verifier has oracle access to query locations of $\pi$ without reading the whole string

• **Interactive Oracle Proofs (IOPs)**
  - Multi-round, verifier receives new “proof strings” each round and has oracle access to all “proof strings” received
Linear PCP

• PCP is a function $f_{\pi}: \mathbb{F}^\ell \rightarrow \mathbb{F}$
• **Equivalent:**
  • Prover sends proof vector $\pi$ over the field $\mathbb{F}$
  • Verifier queries oracle on vector $q \in \mathbb{F}^\ell$ and receives back the inner-product $\langle \pi, q \rangle$
Building Efficient SNARKs

- Step 1: Circuit $C \Rightarrow$ R1CS program \textbf{(last lecture!)}
- Step 2: R1CS program $\Rightarrow$ (zk) linear PCP or IOP
- Step 3: (zk) linear PCP $\Rightarrow$ trusted setup (zk) SNARK
  or multi-round linear IOP $\Rightarrow$ transparent (zk) SNARK
Recap: R1CS

- Matrices $A, B, C \in \mathbb{F}^{m \times (n+1)}$
- Input $z = (1, x, w)$ over $\mathbb{F}^{n+1}$
- R1CS program “accepts” $(x, w)$ if and only if:
  \[(A \cdot z) \circ (B \cdot z) = (C \cdot z)\]
**Linear PCP for R1CS**

- **Def:** A linear PCP for R1CS (A, B, C) over \( \mathbb{F} \):
  - \( V_1 \) runs in time \( O(m) \) and \( V_2 \) in time \( O(|x|) \)
  - Prover honest \( \Rightarrow (V_1, V_2) \) always accepts
  - Prover dishonest \( \Rightarrow \Pr[(V_1, V_2) \text{ accepts}] < \frac{m}{|\mathbb{F}|} \)
Polynomial Interpolation

• Let \( f(X) = f_0 + f_1X + f_2X^2 + \cdots + f_dX^d \) over \( \mathbb{F} \)
• **Interpolate:** Given evaluations \( f(\alpha_0), \ldots, f(\alpha_d) \) at any points \( (\alpha_0, \ldots, \alpha_d) \) can uniquely recover \( f = (f_0, \ldots, f_d) \)
• Proof: Vandermonde matrix is invertible.

\[
V(\alpha_0, \ldots, \alpha_d) := \begin{bmatrix}
1 & \alpha_0 & \alpha_0^2 & \cdots & \alpha_0^d \\
1 & \alpha_1 & \alpha_1^2 & \cdots & \alpha_1^d \\
1 & \alpha_d & \alpha_d^2 & \cdots & \alpha_d^d \\
\end{bmatrix} \Rightarrow V(\alpha_0, \ldots, \alpha_d) \begin{bmatrix} f_0 \\ f_1 \\ \vdots \\ f_d \end{bmatrix} = \begin{bmatrix} f(\alpha_0) \\ f(\alpha_1) \\ \vdots \\ f(\alpha_d) \end{bmatrix}
\]
**Fact:** Let $f, g \in \mathbb{F}[X]$ be polynomials of degree at most $d$. If $f \neq g$ then $\Pr[f(r) = g(r)] \leq \frac{d}{|\mathbb{F}|}$.

**Proof:**

$f(r) = g(r) \Rightarrow r$ is a root of $h = f - g$

$h$ has at most $d$ roots

*Generalizes to multi-variate polynomials, $d$ replaced by total degree (Schwartz-Zippel)*
R1CS $\Rightarrow$ Linear PCP

- Matrices $A, B, C \in \mathbb{F}^{m \times (n+1)}$ and $z = (1, x, w) \in \mathbb{F}^{n+1}$
- Define (degree $m-1$) $f(X)$ by as interpolation of $f(i) = (A \cdot z)_i$
- Define (degree $m-1$) $g(X)$ as interpolation of $g(i) = (B \cdot z)_i$
- Define (degree $2m-1$) $h(X)$ as interpolation of $h(i) = (C \cdot z)_i$ for $i \leq m$ and $h(i) = f(i)g(i)$ for $i = m + 1, \ldots, 2m$

**Idea:** If $h = f \cdot g$ then $(A \cdot z) \circ (B \cdot z) = C \cdot z$

If prover is honest, then $h = f \cdot g$ as defined above

Linear PCP verifier “checks” $h(r) = f(r)g(r)$ at random $r$

$\Rightarrow h = f \cdot g$ with probability $1 - \frac{2m}{|\mathbb{F}|}$
R1CS ⇒ Linear PCP

Observe: \[ f(r) = [1, r, r^2, \ldots, r^{m-1}] [V(1, \ldots, m)]^{-1} \begin{bmatrix} f(1) \\ \vdots \\ f(m) \end{bmatrix} \]
\[ = [1, r, r^2, \ldots, r^{m-1}] [V(1, \ldots, m)]^{-1} A \cdot z = \langle q_1, z \rangle \]

Similarly:
- \[ g(r) = [1, r, r^2, \ldots, r^{m-1}] [V(1, \ldots, m)]^{-1} B \cdot z = \langle q_2, z \rangle \]
- \[ h(r) = [1, r, r^2, \ldots, r^{2m-1}] [V(1, \ldots, 2m)]^{-1} \begin{bmatrix} C \\ I_m \end{bmatrix} \cdot \begin{bmatrix} z \\ h(m + 1) \\ \vdots \\ h(2m) \end{bmatrix} \]
\[ = \langle q_3, (z, h(m + 1), \ldots, h(2m)) \rangle \]
• Prover sends $\pi = [w, h(m + 1), \ldots, h(2m)]$

• Verifier samples $r$ and computes the highlighted portion, i.e. vectors $q_1, q_2, \text{and } q_3$

• Verifier has the first $n$ components of $z$, i.e. $(1, x)$. Let $q_i = (q_i^L, q_i^R)$ where $q_i^L$ denotes the first $n$ components.

• The three LPCP queries and responses are:
  - $a = \langle (q_1^R 0^m), \pi \rangle$; $b = \langle (q_2^R 0^m), \pi \rangle$; $c = \langle (q_3^R), \pi \rangle$

• Verifier checks:
  $$(\langle q_1^L, (1, x) \rangle + a) \cdot (\langle q_2^L, (1, x) \rangle + b) = \langle q_3^L, (1, x) \rangle + c$$
Linear-only encoding

- $\text{Gen}(\lambda) \rightarrow pk, C$  // $C$ is an encoding space
- $\text{Enc}(pk, x \in \mathbb{F}) \rightarrow c \in C$
- $\text{Verify}(pk, c \in C) \rightarrow 0$ or $1$ checks that $c$ is a valid encoding
- $\text{Add}(pk, c_1 \in C, c_2 \in C) \rightarrow c^* \in C$
  - If $c_i = \text{Enc}(pk, x_i)$ then $c^* = \text{Enc}(pk, x_1 + x_2)$
- $\text{QuadTest}(pk, (c_1, c_2, c_3) \in C^{3n}, \alpha \in \mathbb{F}^n) \rightarrow 0$ or $1$
  - Output $1$ iff $c_i[j] = \text{Enc}(pk, x_i[j])$ and $\langle x_1, x_2 \rangle = \langle \alpha, x_3 \rangle$

**One-way**: Infeasible to “decrypt” encoding of random $x$

**Linear-only**: If A receives encodings $\{c_i\}$ of $\{x_i\}$ and outputs valid encoding $c^*$ then $c^*$ must encode affine-linear combination of $\{x_i\}$, $c^* = \text{Enc}(pk, \sum a_i x_i + b)$
In practice these linear-only encodings are constructed using bilinear-groups.
Values are encoding “in the exponent” as \( g^x \).
The pairing operation in bilinear-groups is used for QuadTest, which can perform “multiplication in the exponent”:

\[
e(g^x, g^y) = e(g^{x \cdot y}, g)
\]
Linear PCP $\Rightarrow$ SNARK

- Trusted party runs $G(A, B, C)$:
  - Choose secret random $r \leftarrow \mathbb{F}$
  - $S_V \leftarrow Enc(q^L_1), Enc(q^L_2), Enc(q^L_3)$ (encode individual components)
  - $S_P \leftarrow Enc(q^R_1), Enc(q^R_2), Enc(q^R_3)$ (encode individual components)

- Idea:
  - Prover can only output encodings of linear transformations of queries:
    $[a] = Enc(\langle q^R_1, w_1 \rangle), [b] = Enc(\langle q^R_2, w_2 \rangle), [c] = Enc(\langle q^R_3, w_3 \rangle)$
  - Verifier can check that $a \cdot b = c$ using QuadTest($pk, [a], [b], [c]$)
  - Remaining issue: how to force prover to use the same $w_1 = w_2 = w_3$?
Linear PCP $\Rightarrow$ SNARK

- Solution: add one more query that does random linear check
  - Choose random $\alpha, \beta, \gamma$
  - Query for $d = \langle q^*, w \rangle$ where $q^* = \alpha q_1^R + \beta q_2^R + \gamma q_3^R$
  - Check that $d = \alpha a + \beta b + \gamma c$

- Prover forced to choose $w_1, w_2, w_3$ independently of the secret $\alpha, \beta, \gamma$, thus:
  
  $d - (\alpha a + \beta b + \gamma c) = \alpha \langle q_1^R, w - w_1 \rangle + \beta \langle q_1^R, w - w_2 \rangle + \gamma \langle q_1^R, w - w_3 \rangle$

  If any $\langle q_i^R, w \rangle \neq \langle q_i^R, w_i \rangle$ then this is 0 with probability $< 1/|\mathbb{F}|$
Linear PCP ⇒ SNARK

- **Another problem?** Prover can encode \((u_1, u_2, u_3, u_4) \in \mathbb{F}^4\) of its choice and respond with encodings of \(a' = \langle q_1^R, w_1 \rangle + u_1, b' = \langle q_2^R, w_2 \rangle + u_2, c' = \langle q_3^R, w_3 \rangle + u_3, d' = \langle q^*, w \rangle + u_4\)

- In other words, prover can respond with affine-linear transformation \(Mw + u \in \mathbb{F}^3\)

- But if \(u \neq 0\) then:
  \[
  d' - (\alpha a' + \beta b' + \gamma c') \\
  = \alpha(\langle q_1^R, w = w_1 \rangle - u_1) + \beta(\langle q_1^R, w = w_2 \rangle - u_2) + \gamma(\langle q_1^R, w = w_3 \rangle - u_3) + u_4
  \]

  is 0 with probability \(< 1/|\mathbb{F}|\)
Summary so far

• R1CS ⇒ Linear PCP
• Linear PCP ⇒ SNARK
  • Trusted setup S forms the LPCP queries and encodes them.
  • SNARK prover forced to output affine-linear transformations of queries
  • Extra query forces prover to apply same linear transformation to each query
• Proof is 4 encoded elements. Verifier does two QuadTests.
**Zero Knowledge**

**SNARK ZK Def:** There exists an efficient simulator $Sim$ that for all inputs $x$ where $\exists w: C(x, w) = 0$ outputs:

$$Sim(x) \rightarrow (S_P^*, S_V^*, \pi^*)$$

which, as a random variable over the internal randomness of the program $Sim$, is distributed identically to $(S_P, S_V, \pi)$ sampled as:

$$(S_P, S_V) \leftarrow S(C)$$

$$\pi \leftarrow P(S_P, x, w)$$
**Intuition behind definition:**

- *Sim* can simulate the proof \( \pi \) without the witness \( w \). Therefore, the proof reveals nothing about \( w \) that an observer did not already know before seeing \( \pi \).

- But how? Clearly the *Sim* cannot produce a convincing proof \( \pi^* \), otherwise soundness breaks. Then how can it simulate? Seems like a contradiction..

\[ ⇒ \] The proof \( \pi^* \) is not convincing because *Sim* can “cheat” on the setup (e.g. by generating \( S_P^*, S_V^* \) after creating \( \pi^* \), or by using **knowledge of the secret** in its simulation of a trusted-setup)

\[ ⇒ \] In other words, *Sim* has power the real prover is not given.
ZK Linear PCP

**Linear PCP ZK Def:** A linear PCP for an R1CS program \((A, B, C)\) is zero-knowledge if there exists \(Sim\) that takes an input \(x\) and verifier program \(V'\) satisfying the following:

For all \(x \in \mathbb{F}^{n_1}\) s.t. \(\exists w \in \mathbb{F}^{m-n_1}\) s.t. program accepts \((x, w)\) and all verifiers \(V'\) making queries \(M', Sim\) outputs:

\[
Sim(x, V') \rightarrow (S^*_P, S^*_V, v^*)
\]

a random variable distributed identically to \((S_P, S_V, M' \pi)\) from:

\[
(S_P, S_V) \leftarrow S(A, B, C) \\
\pi \leftarrow P(S_P, x, w)
\]
Honest-Verifier-ZK (HVZK) for LPCP: Assume verifier follows the protocol, then \( Sim \) produces output \((S_P^*, S_V^*, \nu^*)\) which is distributed identically to \((S_P, S_V, M\pi)\).

- Why is this enough? Because in transformation to SNARK, our trusted setup selects the query vectors honestly.
• Prover samples random $\delta_1, \delta_2 \in \mathbb{F}$.
• Interpolate $f', g'$ at one more point: $f'(0) = \delta_1, g'(0) = \delta_2$
• $h'(0) = f'(0)g'(0) = \delta_1\delta_2$
• $f', g'$ now degree m each and $h$ is degree 2m.
• R1CS Constraint Equation still equivalent to $h'(i) = f'(i)g'(i)$ for $i = 1, \ldots, m$, thus implied by $h' = f' \cdot g'$
• Why ZK? Queries reveal no more than $f'(r)$ and $g'(r)$ at secret point. These are independent & uniformly distributed if $r \notin \{1, \ldots, m\}$.

\[
\begin{align*}
  f'(X) - f(X) &= \alpha \prod_{i \in [1, m]} (X - i) \Rightarrow \alpha = \frac{\delta_1 - f(0)}{(-1)^m m!} \quad \text{(ind. uni. random)} \\
  f'(r) &= f(r) + \alpha \prod_i (r - i)
\end{align*}
\]
• The construction method we showed fundamentally requires trusted-setup per C
• Transparent SNARKs from interactive oracle proofs:
  - Verifier selects randomness for queries rather than setup. Then we remove interaction with Fiat-Shamir
  - Needs succinct queries (not linear PCPs) \( \rightarrow \) polynomial IOPs or point IOPs
  - Compile using Merkle commitments or polynomial commitments
Conclusion