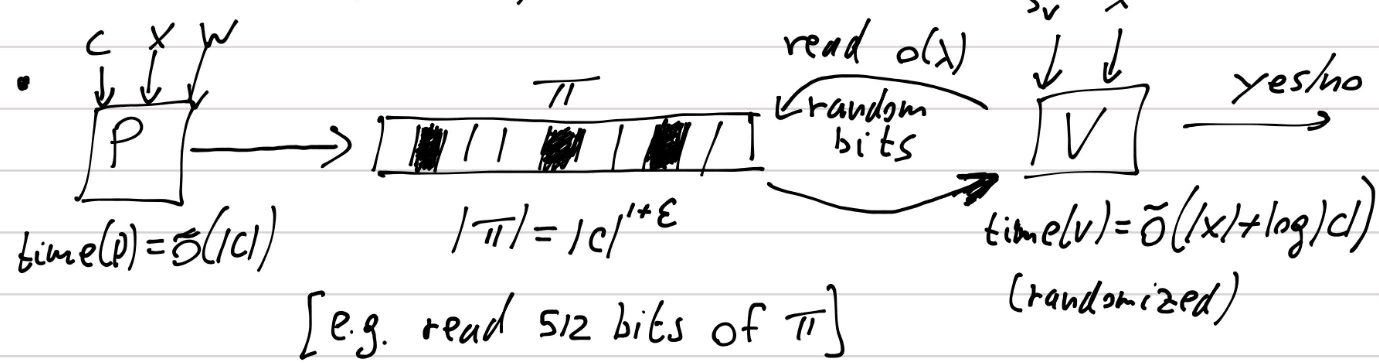
SNARKs Lecture 4:
Linear PCPs & Preprocessing SNARKs

Instructor: Ben Fisch

Recap: PCP Theorem

- 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

- $S(C) \to (S_p = C,\ S_v)$
- 

- If $C(x, w) = 0$ then V always accepts
- If $\nexists\, w$: $C(x, w) = 0$ then $\Pr[V\ accepts] < \left(\frac{1}{2}\right)^{\lambda}$ (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 π, and Verifier has oracle access to query locations of π 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 $\mathrm{f}_\pi \colon \mathbb{F}^\ell \to \mathbb{F}$
- <u>Equivalent:</u>
    - Prover sends proof vector π 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 ⇒ R1CS program **(last lecture!)**
- Step 2: R1CS program ⇒ (zk) linear PCP or IOP
- Step 3: (zk) linear PCP ⇒ trusted setup (zk) SNARK

  or multi-round linear IOP ⇒ transparent (zk) SNARK

Recap: R1CS

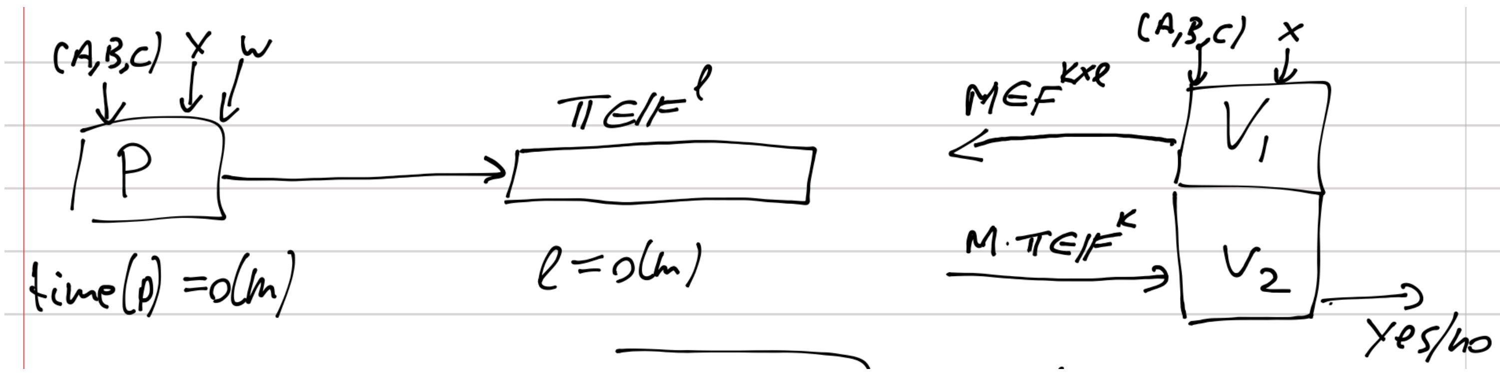
- Matrices $A, B, C \in \mathbb{F}^{m \times (n+1)}$
- Input $z = (1, x, w)$ over $\mathbb{F}^{n+1}$
- R1CS program "accepts" (x, w) if and only if:

$$(A \cdot z) \circ (B \cdot z) = (C \cdot z)$$

Component-wise product

Linear PCP for R1CS

- **Def:** A <u>linear PCP</u> for R1CS (A, B, C) over $\mathbb{F}$:



- V_1 runs in time O(m) and V_2 in time O(|x|)
- Prover honest ⇒ (V_1, V_2) always accepts
- Prover dishonest ⇒ $\Pr[(V_1, V_2)\ accepts] < \frac{m}{|\mathbb{F}|}$

Polynomial Interpolation

- Let $f(X) = f_0 + f_1 X + f_2 X^2 + \cdots + f_d X^d$ over $\mathbb{F}$
- **Interpolate:** Given evaluations $f(\alpha_0), \ldots, \mathrm{f}(\alpha_d)$ at any points $(\alpha_0, \ldots, \alpha_d)$ can uniquely recover $\boldsymbol{f} = (\mathrm{f}_0, \ldots, \mathrm{f}_\mathrm{d})$
- Proof: Vandermonde matrix is invertible.

$$V(\alpha_0, \ldots, \alpha_d) := \begin{bmatrix} 1 & \alpha_0 & \alpha_0^2 & \cdots & \alpha_0^d \\ \vdots & & & \ddots & \vdots \\ 1 & \alpha_d & \alpha_d^2 & \cdots & \alpha_d^d \end{bmatrix} \Rightarrow V(\alpha_0, \ldots, \alpha_d) \begin{bmatrix} f_0 \\ f_1 \\ . \\ . \\ f_d \end{bmatrix} = \begin{bmatrix} f(\alpha_0) \\ f(\alpha_1) \\ . \\ . \\ f(\alpha_d) \end{bmatrix}$$

Fundamental Theorem of Algebra

Fact: Let $f, g \in \mathbb{F}[\mathrm{X}]$ be polynomials of degree at most d. If $f \neq g$ then $\Pr[f(r) = g(r)] \leq \frac{d}{|\mathbb{F}|}$.

Proof:

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

h has at most d roots

Generalizes to multi-variate polynomials, d replaced by total degree (Schwartz-Zippel)

R1CS ⇒ 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 $f(\mathrm{i}) = (\mathrm{A} \cdot z)_i$
- 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 $i = m + 1, \dots, 2m$

<u>Idea:</u> If $h = f \cdot g$ then $(\mathrm{A} \cdot z) \circ (B \cdot z) = C \cdot z$
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: $f(r) = [1, r, r^2, .., r^{m-1}]\ [V(1, \dots, m)\ ]^{-1} \begin{bmatrix} f(1) \\ \dots \\ f(m) \end{bmatrix}$

$= [1, r, r^2, .., r^{m-1}]\ [V(1, \dots, m)\ ]^{-1} \mathrm{A} \cdot z = \langle q_1, z \rangle$

Similarly:

- $\mathrm{g}(r) = [1, r, r^2, .., r^{m-1}]\ [V(1, \dots, m)\ ]^{-1} \mathrm{B} \cdot z = \langle q_2, z \rangle$
- $\mathrm{h}(r) = [1, r, r^2, .., r^{2m-1}]\ [V(1, \dots, 2m)\ ]^{-1} \begin{bmatrix} C \\ I_m \end{bmatrix} \cdot \begin{bmatrix} z \\ h(m+1) \\ \dots \\ h(2m) \end{bmatrix}$

  $= \langle q_3, (z, h(m+1), \dots, h(2m)) \rangle$

R1CS $\Rightarrow$ Linear PCP

- 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, and q_3
- Verifier has the first ℓ components of z, i.e. (1, x). Let $q_i = (q_i^L \; q_i^R)$ where q_i^L denotes the first ℓ components.
- The three LPCP queries and responses are:
  - $a = \langle (q_1^R 0^m), \pi \rangle$; $b = \langle (q_2^R 0^m, \pi \rangle$; $c = \langle (q_3^R 0^m, \pi \rangle$
- Verifier checks:

$$\left(\langle q_1^L, (1, x)\rangle + a\right) \cdot \left(\langle q_2^L, (1, x)\rangle + b\right) = \langle q_3^L, (1, x)\rangle + c$$

Linear-only encoding

- $Gen(\lambda) \rightarrow pk, C \quad // \ C \ is \ an \ encoding \ space$
- $Enc(pk, \mathrm{x} \in \mathbb{F}) \rightarrow c \in C$
- Verify$(pk, c \in C) \rightarrow 0 \ or \ 1$ checks that c is a valid encoding
- $Add(pk, c_1 \in C, c_2 \in C) \rightarrow c^* \in C$
  - If $c_i = Enc(pk, x_i)$ then $c^* = Enc(pk, x_1 + x_2)$
- $QuadTest(pk, (\boldsymbol{c}_1, \boldsymbol{c}_2, \mathbf{c}_3) \in C^{3n}, \boldsymbol{\alpha} \in \mathbb{F}^n) \rightarrow$ 0 or 1
  - Output 1 iff $\boldsymbol{c_i}[j] = Enc(pk, \boldsymbol{x}_i[j])$ and $\langle \boldsymbol{x_1}, \boldsymbol{x_2} \rangle = \langle \boldsymbol{\alpha}, \boldsymbol{x_3} \rangle$

Linear combinations

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 $\{x_i\}$, $c^* = Enc(pk, \sum a_i x_i + b)$

Quadratic Test

- $QuadTest(pk, (\boldsymbol{c}_1, \boldsymbol{c}_2, \mathbf{c}_3) \in C^{3n}, \boldsymbol{\alpha} \in \mathbb{F}^n) \rightarrow$ 0 or 1
  - Output 1 iff $\boldsymbol{c_i}[j] = Enc(pk, \boldsymbol{x}_i[j])$ and $\langle \boldsymbol{x_1}, \boldsymbol{x_2} \rangle = \langle \boldsymbol{\alpha}, \boldsymbol{x_3} \rangle$
- This can be used to test any quadratic equation over encoded elements
- Example: given $c_1 = Enc(a_1), c_2 = Enc(a_2), c_3 = Enc(a_3)$ test that $(a_1 + 1)(a_2 + a_3) - a_3 = 0$
  - Rewrite equation as $a_1 a_2 + a_3 a_1 + a_2 = 0$
  - Place into format: $\langle [a_1, a_3], [a_2, a_1] \rangle = -a_2$
  - QuadTest on inputs $\boldsymbol{c_1} = [c_1, c_3], \boldsymbol{c_2} = [c_2, c_1], \boldsymbol{c_3} = c_2, \alpha = -1$

Linear-only encoding

- 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 ⇒ SNARK

- Trusted setup $S(A, B, C) \rightarrow (S_V, S_P)$:
  - Choose secret random $r \leftarrow \mathbb{F}$
  - $S_V \leftarrow Enc(q_1^L), Enc(q_2^{L:}), Enc(q_3^L)$ (encoding individual components)
  - $S_P \leftarrow Enc(q_1^R), Enc(q_2^{R:}), Enc(q_3^R)$ (encoding individual components)
- (in more detail) **encoding vectors:**
  - If **v**$= (v_1, \ldots v_n)$ then $Enc(\boldsymbol{v}) = \{Enc(pk, v_1), \ldots, Enc(pk, v_n)\}$
  - $S_V = \{Enc(pk, q_{1,j}^L)\}_{j\in[\ell]}, \{Enc(pk, q_{2,j}^L)\}_{j\in[\ell]}, \{Enc(pk, q_{3,j}^L)\}_{j\in[\ell]}$
  - $S_P = \{Enc(pk, q_{1,j}^L)\}_{j\in[n-\ell]}, \{Enc(pk, q_{2,j}^L)\}_{j\in[n-\ell]}, \{Enc(pk, q_{3,j}^L)\}_{j\in[n-\ell]}$

Linear PCP ⇒ SNARK

- Trusted setup $S(A, B, C) \to (S_V, S_P)$:
  - Choose secret random $r \leftarrow \mathbb{F}$
  - $S_V \leftarrow Enc(q_1^L), Enc(q_2^{L:}), Enc(q_3^L)$ (encode individual components)
  - $S_P \leftarrow Enc(q_1^R), Enc(q_2^{R:}), Enc(q_3^R)$ (encode individual components)
- **Idea:**
  - Prover can only output encodings of linear transformations of queries: $[\mathrm{a}] = Enc(\langle q_1^R, \pi_1 \rangle)$, $[\mathrm{b}] = Enc(\langle q_2^R, \pi_2 \rangle)$, $[\mathrm{c}] = Enc(\langle q_3^R, \pi_3 \rangle)$
  - Verifier can check that $a \cdot b = c$ using QuadTest$(pk, [a], [b], [c])$
  - *Remaining issue! What if the prover does not set* $\pi_1 = \pi_2 = \pi_3 = \pi$?

Linear PCP $\Rightarrow$ SNARK

- Remaining Issue: Prover could output encodings of $\langle q_1^R, \pi_1 \rangle, \langle q_2^R, \pi_2 \rangle, \langle q_3^R, \pi_3 \rangle$ for different $\pi_1 \neq \pi_2 \neq \pi_3$. This would be different from LPCP Prover, which sends a single π oracle, and verifier makes multiple linear queries to same π.
- Need a solution such that if the Prover uses distinct $\pi_1 \neq \pi_2 \neq \pi_3$, then it fails some check.
- Solution! Modify the original LPCP to add one more query that is a random linear combination of the first three queries, and modify LPCP verifier to check consistency of response

Random Linear Combination Query

- Random linear combination (RLC) query
  - LPCP verifier chooses random $\alpha, \beta, \gamma \in \mathbb{F}$
  - LPCP verifier queries $\mathrm{d} = \langle q^*, \pi \rangle$ where $q^* = \alpha q_1^R + \beta q_2^R + \gamma q_3^R$
  - LPCP verifier checks that $d - \alpha a + \beta b + \gamma c = 0$
- Note that if $\mathrm{a}' = \langle q_1^R, \pi_1 \rangle, b' = \langle q_2^R, \pi_2 \rangle, c' = \langle q_3^R, \pi_3 \rangle$ then:

  $\Delta \coloneqq d - (\alpha a' + \beta b' + \gamma c') = \alpha \langle q_1^R, \pi - \pi_1 \rangle + \beta \langle q_1^R, \pi - \pi_2 \rangle + \gamma \langle q_1^R, \pi - \pi_3 \rangle)$

  thus if $\langle q_i^R, \pi \rangle \neq \langle q_i^R, \pi_i \rangle$ for some $i \in \{1,2,3\}$, then the probability that $\Delta = 0$ is bounded by $1/|\mathbb{F}|$

Random Linear Combination Query

- <u>SNARK Setup S(A, B, C):</u>
  - Choose random $\mathrm{r}, \alpha, \beta, \gamma \in \mathbb{F}$
  - Form vectors $\{q_1^L, q_1^R\}, \{q_2^L, q_2^R\}, \{q_3^L, q_3^R\}, q^* = \alpha q_1 + \beta q_2 + \gamma q_3$
  - Output encodings of these vectors and α, β, γ
- <u>SNARK Prover:</u>
  - Prover outputs encodings of a', b', c', d.
  - Since α, β, γ are secret, prover chooses π_1, π_2, π_3 independently of α, β, γ, thus unless $\pi_1 = \pi_2 = \pi_3$, then $\Delta = 0$ with negl. Probability
- <u>SNARK Verifier:</u>
  - Uses QuadTest to check $d - \alpha a' - \beta b' - \gamma c' = 0$

Linear PCP ⇒ SNARK

- Another problem? Prover can encode $(\mathrm{u}_1, \mathrm{u}_2, \mathrm{u}_3, \mathrm{u}_4) \in \mathbb{F}^4$ 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 $M\boldsymbol{w} + \boldsymbol{u} \in \mathbb{F}^3$
- RLC query handles this: if $\boldsymbol{u} \neq 0$ then:

$$
\begin{aligned}
&d' - (\alpha a' + \beta b' + \gamma c') \\
&= \alpha\left(\langle q_1^R, w - w_1 \rangle - u_1\right) + \beta\left(\langle q_1^R, w - w_2 \rangle - u_2\right) \\
&+ \gamma\left(\langle q_1^R, w - w_3 \rangle - u_3\right) + u_4
\end{aligned}
$$

is 0 with probability < 1/|$\mathbb{F}$|

Final SNARK

SNARK Setup S(A, B, C):

- Choose random r, $\alpha, \beta, \gamma \in \mathbb{F}$
- Form vectors $\{q_1^L, q_1^R\}, \{q_2^L, q_2^R\}, \{q_3^L, q_3^R\}, q^* = \alpha q_1 + \beta q_2 + \gamma q_3$
- Output $S_V \leftarrow Enc(q_1^L), Enc(q_2^L), Enc(q_3^L), Enc([\alpha, \beta, \gamma])$
- Output $S_P \leftarrow Enc(q_1^R), Enc(q_2^R), Enc(q_3^R)$

SNARK Prover:

- Output encodings $[a]$, $[b]$, $[c]$, $[d]$ of $a = \langle q_1^R, \pi \rangle, b = \langle q_2^R, \pi \rangle, c = \langle q_3^R, \pi \rangle$ and $\mathrm{d} = \langle q^*, \pi \rangle$ using the linear-only encoding operation Add

SNARK Verifier:

- Derive encodings [e],[f],[g] of $\mathrm{e} = \langle q_1^L, (1, x) \rangle, f = \langle q_2^L, (1, x) \rangle$, g = $\langle q_3^L, (1, x) \rangle$
- Apply QuadTest to $[a]$, $[b]$, $[c]$, $[d]$, $[e]$, $[f]$, $[g]$ to check that:

1. $d - \alpha a - \beta b - \gamma c = 0$ and 2. $(e + a) \cdot (f + b) = (g + c)$

Summary so far

- R1CS $\Rightarrow$ $Linear\ PCP$
- Linear PCP $\Rightarrow$ $SNARK$
    - 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 just 4 encoded elements. Verifier does two QuadTests.

Zero Knowledge

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

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

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

$$(S_P, S_V) \leftarrow S(C)$$
$$\pi \leftarrow P(S_P, x, w)$$

Zero Knowledge

Intuition behind definition:

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

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

  ⇒ The proof π^* is not convincing because Sim can "cheat" on the setup (*e.g.* by generating S_P^*, S_V^* **<u>after</u>** creating π^*, or by using **<u>knowledge of the secret</u>** 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:

$$Sim(\mathrm{x}, V') \rightarrow (S_P^*, S_V^*, v^*)$$

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 Zero Knowledge

Honest-Verifier-ZK (HVZK) for LPCP: Assume verifier follows the protocol, then Sim produces output (S_P^*, S_V^*, v^*) 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

R1CS ⇒ HVZK Linear PCP

- Prover samples random $\delta_1, \delta_2 \in \mathbb{F}$.
- Interpolate f', g' at one more point: $f'(0) = \delta_1, \text{g}'(0) = \delta_2$
- $h'(0) = f'(0)g'(0) = \delta_1\delta_2$
- f', g' now degree m each and h is degree 2m.
- R1CS Constraint Equation still equivalent to $h'(i) = f'(i)g'(i)$ for $i = 1, \dots, m$, 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 $r \notin \{1, \dots, m\}$.
  - $f'(X) - f(X) = \alpha \prod_{i \in [1,m]} (X - i) \Rightarrow \alpha = \frac{\delta_1 - f(0)}{(-1)^m m!}$ (ind. uni. random)
  - $f'(r) = f(r) + \alpha \prod_i (r - i)$

Next time…

- 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~~) → polynomial IOPs or point IOPs
    - Compile using Merkle commitments or polynomial commitments

Conclusion