Succinct Proofs

Lecture 17
Different Approaches
Verifying outsourced computation (in the form of an arithmetic circuit $C$) efficiently

- Algebraic circuit: gates are addition or multiplication gates
- Verifier knows the inputs to $C$
- Computer/Prover’s cost: proportional to the size of the circuit $|C|$
- Verifier’s cost: $O(d \log |C|)$ where $d = \text{depth of } C$
- Verifier has the multi-linear extension of the “wiring predicates” for each level of $C$

What if Verifier doesn’t need to know the entire input?
SNARKs
Succinct Non-Interactive Arguments of Knowledge

Recall an NP language: \( \{ x \mid \exists w \text{ s.t. } (x,w) \in R \} \) where \( R \) is in P (i.e., a deterministic polynomial time computable language)

NP languages have a trivial non-interactive proof of knowledge: prover sends \( w \) and let verifier check if \( (x,w) \in R \)

Suppose the verifier is only interested in \( x \), not \( w \), and \( |w| >> |x| \)

Succinct: the entire proof is shorter than the witness

Argument: soundness needs to hold only against polynomial time adversaries

Knowledge-soundess: For every PPT malicious prover \( \mathcal{P}^* \), there is an extractor \( \mathcal{E} \) s.t. for any \( x \) for which \( \mathcal{P}^* \) has a non-negligible probability \( p \) of convincing an honest verifier, \( \mathcal{E} \) has close to 1 probability of outputting \( w \) s.t. \( (x,w) \in R \), in poly(1/p) time
SNARKs

from Outsourced Computation

NP languages have a trivial non-interactive proof of knowledge: prover sends $w$ and let verifier check if $(x, w) \in R$

Use GKR to outsource the computation of $R$ back to prover?

- $w$ still needs to be sent, and proof becomes interactive

- Need to add knowledge-soundness

Avoiding sending $w$: Polynomial Commitment scheme

- $(x, w)$ is encoded as a multilinear polynomial $X_0 \cdot P(X_1, ..., X_k) + (1 - X_0) \cdot P'(X_1, ..., X_k')$ where $P, P'$ are multi-linear polynomials that encode $x, w$ respectively. $P'$ is sent as a commitment.

Avoiding interaction: Fiat–Shamir Transformation

- The above two need to also provide Knowledge Soundness
Polynomial Commitment

Prover wants to (succinctly) commit to a polynomial and later let the verifier (interactively) evaluate it on points of its choice.

Generally, a multi-variate polynomial with a known number of variables and known degree.

- e.g., a multi-linear polynomial in GKR. In some other applications, univariate polynomial of a known degree.

Trivial solution: send the coefficients of the polynomial.

- But not succinct and evaluating the polynomial is expensive.

Want verifier’s computation/communication to be sub-linear in the size of the polynomial.

Non-trivial solutions: Using Merkle hashes and low-degree tests; from hardness of discrete logarithm; from bilinear pairings; using “IOPs”; ... [Later]
Removing Interaction

Fiat-Shamir Transformation

Recall GKR verifier sends random evaluation points as challenges

- Can be sent by a trusted third party
- Must be unpredictable for the adversary before sending (or committing) to the polynomial
- OK to allow the adversary to try a polynomial number of times

Recall Random Oracle Model

- Can use Hash(transcript) as the public random coin
  - Transcript includes the statement $x$
  - Otherwise, adversary can choose the statement to be one that passes the (already fixed) checks at the end
SNARKs

from Outsourced Computation

Interactive public-coin version:

- Prover sends a polynomial commitment to \( P' \) which is a multilinear extension of the witness
- Prover and Verifier run GKR, till the last step when verifier wants to evaluate \( X_0 \cdot P(X_1,\ldots,X_k) + (1-X_0) \cdot P'(X_1,\ldots,X_k) \) on a random point. Verifier knows \( P \), and uses the polynomial commitment to evaluate \( P' \)

Even if the polynomial commitment scheme allows a fraction of the committed points to not match the committed polynomial, this fraction just adds to the soundness error

- Can reduce the error by independent parallel repetitions
- If the polynomial commitment is knowledge-sound the interactive proof is knowledge-sound
SNARKs

from Outsourced Computation

Interactive public-coin version:

- Prover sends a polynomial commitment to $P'$ which is a multilinear extension of the witness.

- Prover and Verifier run GKR, till the last step when verifier wants to evaluate $X_0 \cdot P(X_1, \ldots, X_k) + (1-X_0) \cdot P'(X_1, \ldots, X_{k'})$ on a random point. Verifier knows $P$, and uses the polynomial commitment to evaluate $P'$.

- If the polynomial commitment is knowledge-sound the interactive proof is knowledge-sound.

Non-interactive version: Using Fiat-Shamir Transformation

SNARKs
from PCPs

PCP allows verifying a proof by reading a few positions

A proof $\pi$, which the verifier queries on a few places chosen probabilistically and checks a predicate of the statement and the query/answers. Perfect completeness, and soundness error at most $1/2$ (say).

PCP Theorem: Every NP language $L$ has a PCP in which the proof for $x \in L$ is poly($|x|$) long, the verifier queries a constant number of positions chosen using $O(\log |x|)$ bits.

PCP + Merkle tree commitments gives a succinct interactive proof

$\pi$ is committed, and queries are answered by opening bits of $\pi$

Queries by the PCP verifier are public-coin

So can make it non-interactive by Fiat-Shamir heuristic

PCPs are not very concretely efficient: too much work for prover
SNARKs
from Linear PCPs

- PCP with a very long (super-polynomial) but more structured proof
  - Proof $\pi$ is the evaluation of a multi-linear polynomial with 0 as the constant term
    - $\pi(ax+by) = a\pi(x) + b\pi(y)$ for $x,y \in F^k$ and $a,b \in F$
  - Idea: can commit to such a multi-linear polynomial efficiently [Later]

- Linear PCPs + non-interactive multi-linear polynomial commitment schemes yield practical SNARKs
  - e.g., “Groth16”
SNARKs
from MIPs

MIP in which one prover is asked to evaluate a polynomial at one point and the other is asked to evaluate it in a line passing through that point (without revealing the point).

The second prover is used to prevent the first prover from adaptively choosing how to answer.

Use a polynomial commitment scheme instead of the second prover: the single prover must now evaluate the polynomial at the random point.
SNARKs
from IOPs

- Interactive version of PCP: Allow committing to multiple strings over multiple rounds
- Can be made into a proof system using Merkle hashes
- Polynomial IOP: the strings are polynomial evaluations
- Can be implemented using any polynomial commitment scheme
- In particular, there are polynomial commitment schemes which are derived from “standard” IOPs (in turn implemented using Merkle hashes)
- Public coin
- So that it can be made non-interactive