Lecture 13 - SNARGs
Plan
* Recap: linear PCBs
* Applications

* Stretch Break
* SNARE Design
* SNARE from LPCP + Lin. Hom. Enc.

Logistics
* HW4 due Friday Sep via Gradescope
* Please fill out survey on Piazza (even listeners)
Recap: Linear PCP

Consider statements of form "Arithmetic det $C$ is satisfiable."

(over finite field $F$)

Normal proof:

* $V$ reads entire proof
* $V$ accepts/rejects

Linear PCP:

* $P$ outputs pair $(x, \Pi)$
* $V$ makes "linear query" to $(x, \Pi)$
* $V$ accepts/rejects.

Recall:

For $x = (x_1, \ldots, x_n) \in F^n$
$y = (y_1, \ldots, y_n) \in F^n$

$\langle x, y \rangle \overset{\text{def}}{=} \sum_{i=1}^{n} x_i \cdot y_i \in F^n.$
Q: Why does this matter?
Seems like a bogus/contrived notion... queries are as large as the proof... I can't send them.

A: These are a useful building block, as we will see today.

Linear PCP for ckt SAT \rightarrow \text{Succinct Ek proof for ckt SAT} (Bitsuisky, Chiesa, Ishai, Ostrovsky, 2013)
Properties of Linear PCP \((P, V)\) for circuit SAT:

1. Completeness: If \(C(x) = 0\), \(\Pi \leftarrow P(x)\)

\[
P_r \left[ V^{x, \Pi}() = \text{"accept"} \right] = 1.
\]

2. Soundness: If \(C\) is UNSAT then \(V(x^*, \pi^*)\)

\[
P_r \left[ V^{x^*, \pi^*}() = \text{"accept"} \right] < \frac{1}{3}.
\]

3. Honest Verifier ZK:

\[
\forall \text{ SAT circuit } C \exists \text{ sim } \text{Sim} \text{ sat } C \text{ and } \forall \text{ view } v \text{ in interaction with oracles } <x, \pi> \text{:\n}
\[
\{ \text{Sim}() \} = \{ V\text{'s view in interaction with oracle } <x, \pi> \} = \{ \text{Sim}() \}
\]
Constructions

We didn’t look at them last time.

They’re clever, but not complicated.

Pretty easy to implement w/ good constants.

Thing to remember:

If $C$ is a circuit of size $s$ (one it), then there’s a linear PCP for $C$ in which

- Proof has size $O(s)$
- $V$ makes 3 queries

Can optimize # of queries further (Gennaro, Gentry, Parno, Raykova implicitly gave this first construction of LCP with proof size $O(s)$ ... 2015)
Intuition

Why the existence of constant-query LPCCPs should surprise and delight you.

* Normal proof: If satisfying input is \( n \) elements long, \( V \) has to read all \( n \) elements to check proof.

* Linear PCP: Verifier gets only a constant # of field elements worth of info back from the proof.

No matter how big clot is or how long SAT assignat is!

...until you see it, it's hard to understand how this could be possible.
Succinct Non-interactive Argument (SNARG)

- Short proof that convinces $V$ that $C$ is SAT
  - Complete, Sound
- Zero knowledge: Leaks no other info (simulation)
- Succinct: $|\Pi|$ depends only on Sec param... not on size of circuit or sat assignment
  - Time to check proof also depends only on Sec param

Notice: $|\Pi|$ could be much smaller than the NP witness (sat input). Useful property even w/ zkP?
Applications

**Elk Bug Bounty**

Sec. Research

Tech Co.

"There is an input to your program that causes it to segfault."

**Anonymous Auth**

Sec. Research

MIT Library

"I know the sk corresponding to the pk of some person at MIT."

**Delegated computation**

Amazon EC2

Customer

"The output of your multiplication computation is ——."

Here, succinctness is critical.
Stretch Break!
Constructing SNARGs from linear PCPs

As a simplifying assumption, let's first consider "designated-verifier SNARGs".

Samdress only holds if prover cannot get ahold of $vk$.

Furthermore, assume LPCP verifier has structure

\[
\text{Query}() \rightarrow \begin{array}{c}
q_1 \\
q_2 \\
q_3
\end{array}
\]

\[
\text{Decide}(state, a_1, a_2, a_3) \rightarrow \text{accept} / \text{reject}
\]

In other words, the verifier's queries are non-adaptive and are independent of the statement being proved.
Construction

Uses linearly homomorphic encryption w/ keyspace K

\[ E(sk, m_1) + E(sk, m_2) = E(sk, m_1 + m_2) \]

... can build from an array of "public-key assumption"

\[ \text{Picard, DDH, LWE, ...} \]

If ckt is CPC over field \( \mathbb{F} \), then msg in lin hom one scheme should be \( \mathbb{F} \) elements.

---

\[ E(q_1), E(q_2), E(q_3) \]

\[ \mathbb{T} = (E(a_1), E(a_2), \ldots, E(a_k)) \]

SNARK Prover

- Proving key = Enc of LPCP queries
  Verif key = LPCP verif state
- SNARK prover computes LPCP answers "under encryption"
- SNARK verifier decrypts and runs LPCP verifier
**Setup**:

\[
( q_1, q_2, q_3, \text{state} ) \leftarrow \text{LPCP}\cdot \text{Query}(1)
\]

Choose random \(\alpha_1, \alpha_2, \alpha_3 \leftarrow \mathbb{F}\)

\[q_4 \leftarrow \sum_{i=1}^{3} \alpha_i q_i \in \mathbb{F}^{nm}\]

\(\text{sk} \leftarrow \mathcal{K}\)

\(\text{return} \quad pk = (E(\text{sk}, q_1), \ldots, E(\text{sk}, q_4))\)

\(vk = (\text{state}, \alpha_1, \alpha_2, \alpha_3, \text{sk})\)

---

**Prove** \((pk = (Q_1, \ldots, Q_4), x)\):

\(\pi \leftarrow \text{LPCP}\cdot \text{Prove}(x)\)

For \(i = 1, \ldots, 4\):

\(A_i \leftarrow \langle Q_i, \pi \rangle\)

\(\text{return} \quad (A_1, \ldots, A_4)\)

---

**Verify** \((vk = (\text{state}, \alpha_1, \ldots, \alpha_4, \text{sk}), \pi = (A_1, \ldots, A_4))\):

For \(i = 1, \ldots, 4\):

\(a_i \leftarrow \text{Dec}(\text{sk}, A_i)\)

Reject if \(\text{LPCP}\cdot \text{Verify}(\text{state}, a_1, \ldots, a_4) = \text{"reject"}\)

Reject if \(a_4 \neq \sum_{i=1}^{3} \alpha_i a_i \in \mathbb{F}\)

Accept!
This is a very slick construction!

No craziness hiding. It’s really clean and even easy to memorize.

If you’re stuck on a desert island and need a succinct proof system, this is what you’d use.

**Soundness:**

- Essentially follows from LRP soundness.
- Only tricky part is that $P$ can answer different queries w/ different proofs.
  
  → Random linear combinatorial defeats this attack

→ Need a new assumption “linear-only enc” to formally argue soundness. Not great, but also no reason to suspect these assumptions are more false than any other crazy assumption we make.

**Zk:**

- Verifier only gets answers to LRP queries (computed honestly in setup).
- Zk of SNARL follows directly from Zk of LPCP.
Q: Can P & V reuse setup for multiple interactions?

A: Yes, prove statements of the form $C(x)$ is SAT and first $L$ elms of a SAT input are """"."