Lecture 11: SNARGs

May 12th 2020
Plan:

Succinct Non-interactive Arguments (SNARGs)

- PCPs
- Linear PCPs
- SNARGs from linear PCPs

Recall the language of Boolean circuit satisfiability:

\[ L_c = \left\{ x \in \Sigma^* \mid \exists w \in \Sigma^m : C(x, w) = 1 \right\} \]

Proof systems for \( L_c \):

<table>
<thead>
<tr>
<th></th>
<th>NP</th>
<th>( \Sigma )-Protocol</th>
<th>NIZK</th>
</tr>
</thead>
<tbody>
<tr>
<td>P(x,w)</td>
<td>P(x) \rightarrow V(x)</td>
<td>( P \leftarrow P \rightarrow V )</td>
<td>( P \xrightarrow{\Pi} V )</td>
</tr>
<tr>
<td>check that ( C(x, w) = 1 )</td>
<td>( C(x, w) = 1 )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Communication complexity</td>
<td>( \Omega(</td>
<td>C</td>
<td>) )</td>
</tr>
<tr>
<td>Verifier complexity</td>
<td>( \Omega(</td>
<td>x</td>
<td>+</td>
</tr>
</tbody>
</table>

\( \Rightarrow \) Can we do better?

\( \Rightarrow \) Can we have communication complexity & verifier complexity smaller than the witness size?
* Previous lectures: Previous lectures: Previous lectures: Previous lectures: Zero-knowledge (minimizing knowledge complexity) non-interaction (minimizing communication rounds)

* This lecture: succinctness (minimizing communication complexity & verifier complexity)

---

**Definition:** A non-interactive proof/argument system for $L_C$ is succinct if:

- the proof $\Pi$ is of length $|\Pi| = \text{poly}(\lambda, \log |C|)$
- the verifier runs in time $\text{poly}(\lambda, |x|, \log |C|)$

$\Rightarrow$ proof size & verifier complexity much smaller than the NP witness

---

**Related Primitives:**

- **SNARK = SNARG + Proof-of-Knowledge**
- **2K-SNARK = SNARK + Zero-Knowledge** $\rightarrow$ core building block of 2Cash

---

**Why succinct proofs?**

- **Verifiable computation:** client outsources computation of $f(x)$ to the cloud, but wants to verify the result
  $\Rightarrow$ verification must be cheaper than evaluating $f(x)$ locally

- **Cryptocurrencies:** compressing blockchains
  anonymous transactions
  private smart contracts

---

CS 251
How to construct SNARGs (for NP)?

1) In the standard model: unlikely

2) In the random-oracle model: "CS-Proofs" [Kilian, Micali]

3) In the Common-reference-String (CRS) model (with preprocessing)
   - constructions from pairings
   - linearly homomorphic encryption (secret-verifiability)

(Very) active research area? (motivated by many cryptocurrency applications)

Many dimensions of interest:

* Concrete costs (proof size, verifier time, prover overhead)
* Post-quantum security
* Trust assumptions
* Universality

...
A blueprint for constructing SNARGs [Bitansky et al., 2013]

1. Construct an interactive information-theoretic proof-system, assuming some "algebraic" oracle ← "trusted 3rd party"

2. Use cryptography to "emulate" this oracle and to make the proof system non-interactive & succinct
Information-theoretic primitive: Probabilistically-checkable proofs (PCPs)

* Traditional PCPs:

Verifier reads a constant number of randomly chosen bits of the PCP proof, and is convinced with constant probability.

[One of the deepest results in complexity theory: long line of results]

* Linear PCPs:

Verifier gets to ask "linear queries" to the proof oracle.

\[ A \text{ note that } V \text{'s queries are not succinct. We'll need to get around that} \]
**Definition:** A linear PCP for a language \( L \subseteq \text{NP} \) (with corresponding relation \( R \)) is a pair of efficient algorithms \((P, V)\) satisfying the following properties:

- **Completeness:** if \((x, w) \in R\), and \(T \in P(x, w)\), then:
  \[ \Pr \left[ V^{<T>} (x) = 1 \right] = 1 \]

- **Soundness:** for all \(x \in \mathbb{L}\) and all \(T^* \in \mathbb{F}^*\):
  \[ \Pr \left[ V^{<T^*>} (x) = 1 \right] \leq E \quad \text{soundness error} \]

---

**Constructing linear PCPs for Boolean circuit satisfiability:**

- From Walsh-Hadamard codes [Arora-Lund-Motwani-Sudan-Seager 1992] (\(V\) makes 3 queries of length \( n = \mathcal{O}(1|\mathcal{F}|^2) \), soundness error \( E = \frac{2^{|\mathcal{F}|}}{|\mathcal{F}|} \))

- From quadratic span programs [Gennaro-Gentry-Parno-Raykova 2013] (\(V\) makes 3 queries of length \( n = \mathcal{O}(1|\mathcal{F}|) \), soundness error \( E = \frac{|\mathcal{F}|}{|\mathcal{F}|} \))

These constructions have a very useful verifier structure:

\[ V(x) : \]

1. \( q_1, \ldots, q_k, \pi_\text{prep} \leftarrow \text{Gen Queries}() \)
2. \( r_1, \ldots, r_k \leftarrow <\pi_1, q_1>, \ldots, <\pi_k, q_k> \)
3. Output \( \text{Final Check}(\pi_\text{prep}, x, r_1, \ldots, r_k) \)

\( \Rightarrow \) running time is \( \tilde{O}(1|\mathcal{F}|) \)

\( \Rightarrow \) Final Check is a quadratic function of \( n, \ldots, r_k \) (useful for pairing based constructions)
From linear PCPs to $\sqrt{\text{SNARGs}}$ (in the CRS model)

**Problem:** Verifier's queries are not succinct: $O(\lambda^2)$

**Solution:** 

*pre-processing* (embed queries into the CRS)

independent of the statement $x$
can be reused for many statements

* Setup ($1^t$): 1. $q_1, \ldots, q_k, \Psi_{LCP} \leftarrow \text{Verifier. Gen Queries}()$
   2. return $\text{CRS} = (q_1, \ldots, q_k)$ and $\Upsilon = \Psi_{LCP}$

* Prover ($x, w, \text{CRS}$): 1. Encode statement-witness $(x, w)$ as a linear PCP $\Pi \in \mathcal{F}$
   2. Compute oracle responses $r_1 = <\Pi, q_1>, \ldots, r_k = <\Pi, q_k>$
   3. Send proof $\Pi' = (r_1, \ldots, r_k)$ to verifier

* Verifier ($x, \Pi', \Upsilon$): run $\text{Final Check} (\Psi_{LCP}, x, r, \ldots, r_k)$

This is succinct:

\[ \text{proof size is } k \cdot |F| = \tilde{O}(\lambda) \]

\[ \text{verifier runtime is } O(\lambda \cdot |F|) = \tilde{O}(\lambda \cdot |F|) \]

But it's not sound! why?
**Problem:** Prover can choose $TIE^m$ after seeing the Verifier's queries $\Rightarrow$ cannot appeal to soundness of linear PCP!

**Solution:** Encrypt the verifier's queries with additively-homomorphic encryption $\Rightarrow$ this gives a "designated-verifier" SNARG

**Setup** $(1^*)$: $(pk, sk) \leftarrow \text{KeyGen}(1^*)$
- $q_1, \ldots, q_k, ^\perp_{	ext{cece}} \leftarrow \text{Verifier. Gen Queries}(\cdot)$
- Compute $ct_{ij} \leftarrow \text{Encrypt}(pk, (q)_{ij})$ for $i \in [k], j \in [n]$
- Output $\text{CRS} = (pk, \{ct_{ij}: i \in [k], j \in [n]\})$ and $\mathcal{N} = (sk, ^\perp_{	ext{cece}})$

**Prover:** homomorphically compute $ct'_x \leftarrow \sum_{j \in [n]} \Pi_{ij} \cdot ct_{ij} = \text{Encrypt}(pk, ct_{ij})$

**Verifier:** given $ct'_1, \ldots, ct'_k$, decrypt using $sk$ to obtain $r_1, \ldots, r_k$ and run linear PCP Final Check

**Problem:** Prover is not limited to computing linear functions of the queries

**Solution:** Assume encryption scheme is "linear-only" (i.e. it only supports linear homomorphisms)

- Informally: any valid ciphertext that the prover can compute can be "explained" by a linear function of the provided ciphertexts
  $\Rightarrow$ formally captured by the existence of an extractor that can produce the linear function that created the ciphertext.

⚠️ This is an "atypical" cryptographic assumption (non-falsifiable)

⇒ typical assumptions like DDH, factoring, … can be formulated as a game between a challenger and adversary
⇒ to break the linear-only assumption, you need to show an adversary for which there exists no extractor...
Problem: the prover need not use the same linear function for all queries.

Solution: random consistency check

Verifier chooses \( k_1, \ldots, k_k \in F \) and makes a final query \( q_{k+1} = \sum_{i \in [n]} \alpha_i \cdot q_i \)

Verifier checks that \( r_{k+1} = \sum_{i \in [n]} \alpha_i \cdot c_i \)

Case 1: prover is honest and replies \( r_i = \langle q_i, \Pi_i \rangle \) for all \( i \in [k+1] \)

Then, \( \sum_{i \in [k]} \alpha_i \cdot c_i = \sum_{i \in [k]} \alpha_i \cdot \langle q_i; \Pi_i \rangle = \langle \sum_{i \in [k]} \alpha_i \cdot q_i, \Pi_i \rangle = \langle q_{k+1}, \Pi_i \rangle = r_{k+1} \)

Case 2: prover is dishonest and replies

\[
\begin{align*}
r_1 &= \langle q_1, \Pi_1 \rangle \\
r_2 &= \langle q_2, \Pi_2 \rangle \\
\vdots \\
r_{k+1} &= \langle q_{k+1}, \Pi_{k+1} \rangle
\end{align*}
\]

not fully general, as a linear-only prover could compute \( r_i = f(q_i, \ldots, q_{k+1}) \) for some linear function \( f \) — but the same random check idea applies for the general case also

where \( \langle q_i, \Pi_i \rangle \neq \langle q_j, \Pi_j \rangle \) for some \( i, j \in [k] \).

Then, verifier accepts if

\[
\sum_{i \in [k]} \alpha_i \cdot \langle q_i, \Pi_i \rangle = \sum_{i \in [k]} \alpha_i \cdot q_i, \Pi_{k+1} > 0
\]

\[
\Leftrightarrow \sum_{i \in [k]} \alpha_i \cdot (\langle q_i, \Pi_i \rangle - \langle q_i, \Pi_{k+1} \rangle) = 0
\]

must be non-zero for some \( i \) where \( \langle q_i, \Pi_i \rangle \neq \langle q_i, \Pi_{k+1} \rangle \)

\[
\Rightarrow \text{since } \alpha_i \in F \text{ independently of } \Pi_k, \ldots, \Pi_{k+1}, \text{ this relation is satisfied with probability at most } \frac{1}{|F|} \text{ [Shwartz-Zippel Lemma]}
\]
Putting it all together:

**Setup:**
1. Generate \( pk, sk \) for additively homomorphic encryption scheme
2. \( q_1, \ldots, q_k, \gamma_{LCP} \leftarrow \text{Verifier} \cdot \text{GenQueries}(\cdot) \)
3. Let \( x, \ldots, A \in \mathbb{F}^k \) and \( q_{tk} = \Sigma x_i q_i \)
4. Encrypt queries \( q_1, \ldots, q_{tk} \) component-wise to obtain \( \{ C_{t}x_i \}_{x_i \in \mathbb{Z}_p} \)
5. Output \( \text{CRS} = (pk, \{ C_{t}x_i \}), \gamma = (\gamma_{LCP}, sk, x_1, \ldots, x_k) \)

**Prover \((x, w, \text{crs})\):**
1. Construct linear PCP proof \( \Pi \in F^\gamma \) from \( (x, w) \)
2. Homomorphically compute \( C't = \text{Encrypt}(sk, \gamma_{LCP}, \Pi) \)
3. Output \( \Pi_{\text{SNARG}} = (C't, \ldots, C't_{tk+1}) \)

**Verifier \((x, \Pi_{\text{SNARG}}, \gamma)\):**
1. Decrypt ciphertexts in \( \Pi_{\text{SNARG}} \) to \( x, \ldots, x_{tk+1} \)
2. Check linear consistency \( x_{tk+1} = \Sigma x_i q_{tk+1} \)
3. Run \( \text{FinalCheck}(\gamma_{LCP}, x, \gamma, \ldots, \gamma_{tk+1}) \)

Completeness: by correctness of encryption scheme and completeness of linear PCP

Soundness:
- Prover applies linear functions to queries \( \text{Consistent linear function for all queries} \)
- Soundness of linear PCP \( \text{Linear function independent of queries} \)
- Soundness: \( \text{linear-only encryption} \)
- Soundness: \( \text{random linear check} \)
- Soundness: \( \text{semantic security} \)

Succinctness:
- \( \text{Proof is} \ (k+1) \cdot \text{encrypted field elements} + O(1) \)
- \( \text{Verifier runtime:} \ k+1 \text{decryptions is} \ O(1F + A) \)
- \( \text{Final check is} \ O(1F) \)

Pok: This SNARG is also a proof-of-knowledge!

ZK: Easy to add if the encryption scheme is “re-randomizable”

Given \( ct = \text{Enc}(m, k) \), anyone can produce \( ct' \) that is indistinguishable from a fresh re-encryption of \( m \).
Instantiations:

* linear-only encryption from Pallier, ElGamal, Regev, ...
  
  $\rightarrow$ "Designated-verifier" SNARGs
  (the verifier generates the CRS and only the verifier can verify proofs)

* pairing-based linear-only encodings
  
  $\rightarrow$ "publicly verifiable" SNARGs (anyone can verify proofs)

High-level idea:
  1. "encode" queries as $g^{q_i}$ in the CRS
  2. Prover can compute $g^r = g^{c_i,T_r}$ (also need to prevent prover from computing other low functions)
  3. Verifier computes the quadratic final check "in the exponent" using the pairing

$\rightarrow$ Very efficient schemes [Groth16]: Proofs are 3 group elements for any statement
  
  $\rightarrow$ currently used in Zcash

$\rightarrow$ downsides: not post-quantum, trusted-setup

$\rightarrow$ Open problem: SNARK with similar performance as [Groth16] \{without trusted setup, with post-quantum security\}

Somehow has to generate the CRS

If they "cheat" (e.g. if they save the queries $q_i$ in the clear) they can break soundness?