Outline

1. (Long) Introduction
2. Randomized Polynomials (w/applications to round-efficient MPC)
3. Randomized Encodings w/applications to NC$^0$ Cryptography
4. Constant Input Locality
5. Computational Randomized Encodings (w/applications)
6. NC$^0$ Linear Stretch PRG (w/applications)
Parallel Cryptography

How low can we get?

<table>
<thead>
<tr>
<th>Complexity Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>poly-time</td>
</tr>
<tr>
<td>NC</td>
</tr>
<tr>
<td>log-space</td>
</tr>
<tr>
<td>NC^1</td>
</tr>
<tr>
<td>AC^0</td>
</tr>
<tr>
<td>NC^0</td>
</tr>
</tbody>
</table>

Diagram showing a network of connections between the complexity classes.
**Caveats:**
- We get PRG with sub-linear stretch.
- Decryption/verification not in NC^0.
  - In fact, impossible to decrypt/verify.
  - But: can commit in NC^0 with decommit in NC^0.

**Sufficient Assumptions for Crypto in NC^0**

- OWF
- PRG
- Hash
- Sym-Enc
- PK-Enc
- Signature
- Commit
- NIZK

- Assuming min-PRG in NC^1

**Diagram:**

<table>
<thead>
<tr>
<th>P</th>
<th>NC^1</th>
<th>NC^0_4</th>
<th>OWF</th>
<th>PRG</th>
<th>Hash</th>
<th>Sym-Enc</th>
<th>PK-Enc</th>
<th>NI-Com</th>
<th>Sign</th>
<th>NIZK</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>factoring</td>
<td>factoring</td>
<td>factoring</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Factors, discrete-log/DDH, lattices, …
Main Primitives

**OWF**

\[ U_{in} \xrightarrow{f} y = f(U_{in}) \]

find \( x \in f^{-1}(y) \) poly-time

**PRG**

\[ U_{in} \xrightarrow{f} f(U_{in}) \]

\[ U_{out} \]

Pseudorandom or Random?

poly-time
Thm. \( f(x) \) is a OWF \( \Rightarrow g(x,r) \) is a OWF

Proof: inverter B for g \( \Rightarrow \) inverter A for f

- A succeeds whenever B succeeds
  - Follows from perfect correctness of decoder

- A generates a correct input distribution for B
  - Follows from correctness of simulator
Encoding a PRG

- **Want:** $f(x)$ is a PRG $\Rightarrow g(x, r)$ is a PRG
- **Problems:**
  - output of $g$ may not be pseudorandom
  - $g$ may *shrink* its input
- **Solution:** “perfect” randomized encoding
  - respects pseudorandomness, additive stretch, …
  - stretch of $g$ is typically sublinear even if that of $f$ is superlinear
  - most (not all) known constructions give perfectness for free
From Low Degree to Low Locality

• Locality Reduction:
  degree 3 boolean function $\Rightarrow$ locality 4

\[
f(x) = T_1(x) + T_2(x) + \ldots + T_k(x)
\]

\[
g(x, r) = -r_1 + T_2(x) + r_2 + \ldots + -r_k + T_k(x)
\]

\[
-r_1 + s_1 \quad -s_1 - r_2 + s_2 \quad \ldots \quad -s_{k-1} - r_k
\]
Wrapping Up

Composition Lemma:

\[ f \quad g \text{ encodes } f \quad h \text{ encodes } g \]

Concatenation Lemma:

\[ g^{(1)} \text{ encodes } f^{(1)} \quad \ldots \quad g^{(l)} \text{ encodes } f^{(l)} \quad \rightarrow \quad g \text{ encodes } f \]
From Branching Programs to Locality 4

poly-size BPs

BP encoding

composition

degree 3

locality reduction

concatenation

NC$^0_4$

locality 4