Outline

1. (Long) Introduction
2. Randomized Polynomials (w/applications to round-efficient MPC)
3. Randomized Encodings w/applications to NC⁰ Cryptography
4. Constant Input Locality
5. Computational Randomized Encodings (w/applications)
6. NC⁰ Linear Stretch PRG (w/applications)
Computationally Private Encodings

- **Known:** $f \in \text{NC}^1$, $\oplus L$ $\rightarrow$ encoding in $\text{NC}^0$
- **Goal:** $f \in \text{P}$ $\rightarrow$ encoding in $\text{NC}^0$

- **Idea:** relax encoding requirement

- **Thm:** $f \in \text{P}$ $\rightarrow$ computational encoding in $\text{NC}^0_4$
  - assuming "easy PRG" (min-PRG $\in \oplus L$)
  - "Easy PRG" can be based on factoring, discrete-log, lattices
App 1: Relaxed Assumptions for Crypto in $\text{NC}^0$

- Using perfect encoding:

  - Assuming "easy PRG"

  $$\exists L \in \text{NC}^0 \implies \text{exist OWF, OWP, PRG, Hash, Sym-Enc, PK-Enc, Signature, Commit, NIZK}$$
App 2: Parallel Reductions Between Primitives

- What about NC reductions?
- Much less is known....
- New

Blum Micali '82, Yao '82, Levin '85, Goldreich Krawczyk Luby '88, Håstad Impagliazzo Levin Luby '90, Goldreich Micali '84, Goldreich Goldwasser Micali

84 - Goldwasser Micali Rivest '84, Bellare Micali '88, Naor Yung '89, Rompel '90, Naor '89, Impagliazzo Luby '89, ...
In case you don’t insist on unconditional security…

• Securely evaluating an arbitrary function \( f \) *efficiently* reduces to securely evaluating deg-3 polynomials

  … assuming an “easy PRG”

• In particular:

  Basic MPC protocols (e.g., BGW88, CDM00) imply constant-round *computationally* secure MPC for every \( f \).

  Known assuming *any* PRG [BMR90,DI05]; however, current approach is simpler and can be made more efficient [DI06].
Thm. “easy PRG” $\Rightarrow$ encoding in NC$^0$ for all $f \in P$

- $f \in P$ $\Rightarrow$ Yao garbled circuit
- $g \in NC^0[\text{symmetric encryption}]$ $\Rightarrow$ one-time symmetric encryption $\in NC^0[\text{min-PRG}]$
- $g \in \oplus L$ $\Rightarrow$ easy PRG
- $h \in NC^0_4$ [AIK04]
Step 1: Min-PRG $\xrightarrow{\text{NC}_0}$ OTSE

Sol: use naïve construction in parallel to increase key size

- Naïve construction: $E_{\text{key}}(m) = G(\text{key}) \oplus m$, $D_{\text{key}}(c) = G(\text{key}) \oplus c$
- Problem: Cannot encrypt long messages, $|m| \leq |\text{key}| + 1$
- Solution: Transform G into a poly-stretch PRG + naïve
- Problem: We do not know how to do it in NC
- Solution: use naïve in parallel to increase key size
Step 2: Garbled Circuit Construction

- Pair of random colored keys for each wire
- For each input wire, key corresponding to its value is revealed
- Color semantics of output wires are revealed
- Garbled gates:
Step 2: Garbled Circuit Construction

Implementing locks
- (one-time) symmetric encryption
  - computational privacy, works for any circuit
- one-time pads
  - information-theoretic privacy, efficient only for log-depth circuits

• Garbled gates: