Randomization Techniques for Secure Computation and Parallel Cryptography

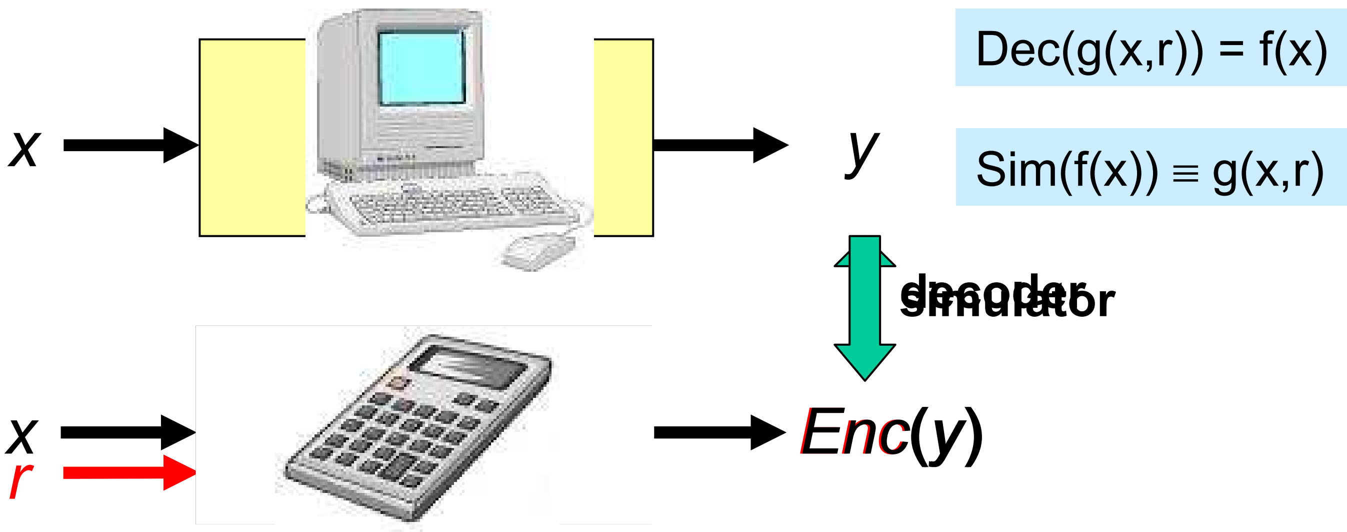
Eyal Kushilevitz (Technion)

Course based on joint works with

Yuval Ishai and Benny Applebaum

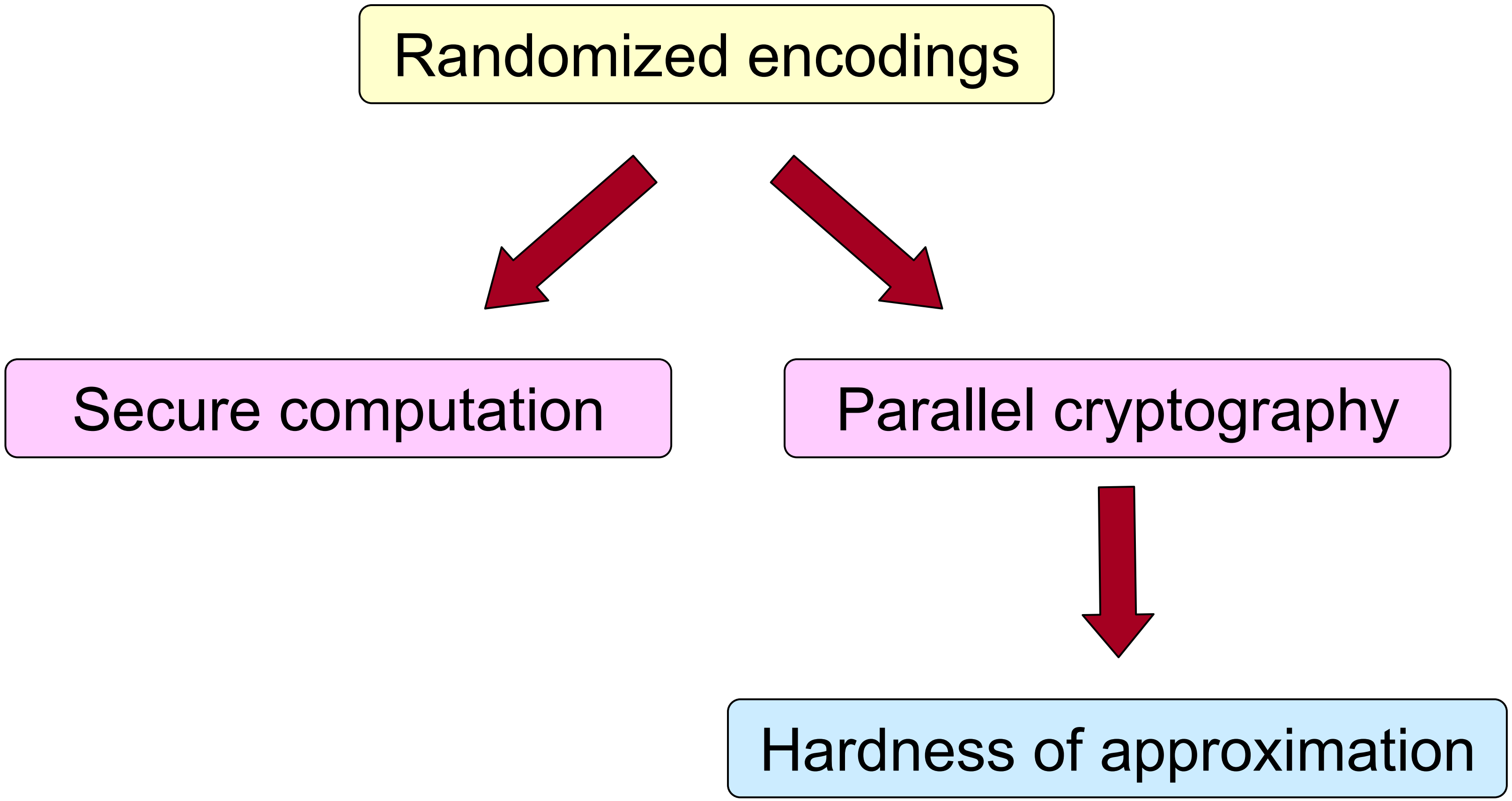
(…….,FOCS 00, ICALP 02, FOCS 04, CCC 05, RANDOM 06, CRYPTO 07)

The Basic Question

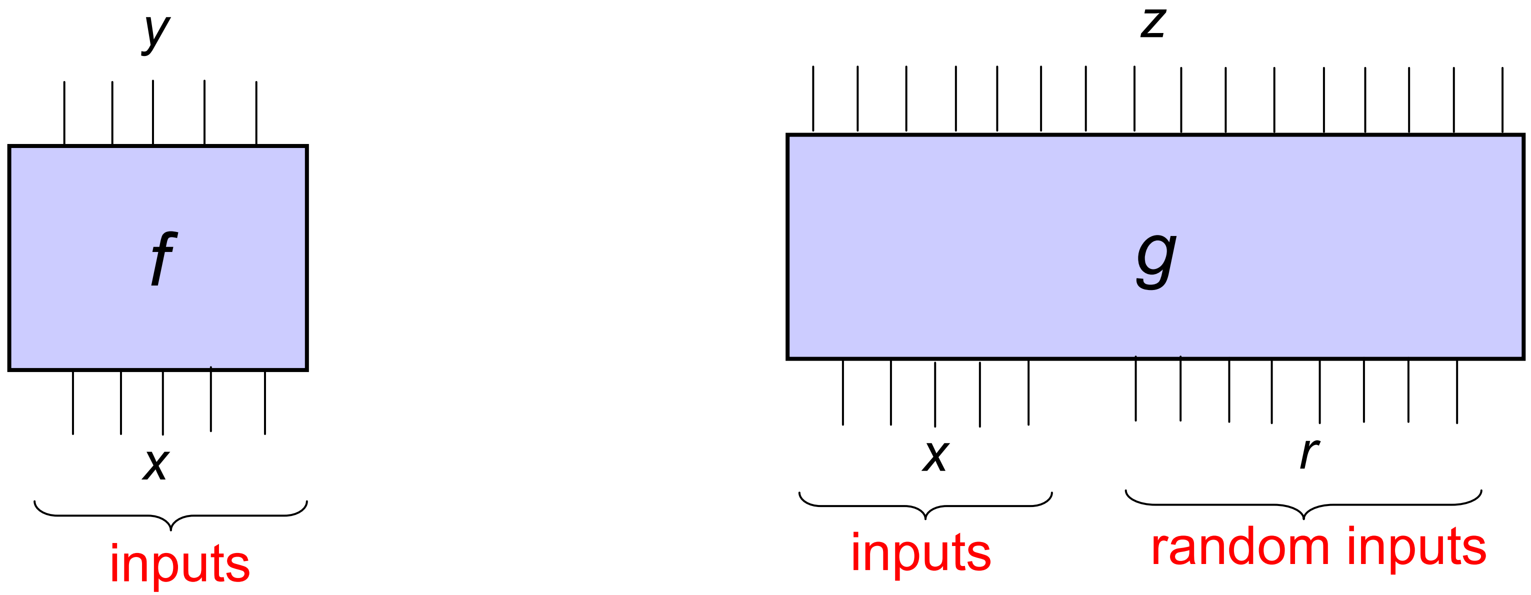


- *g* is a "randomized encoding" of *f*
  - Nontrivial relaxation of computing *f*
- Hope:
  - *g* can be "simpler" than *f*
    (meaning of "simpler" determined by application)
  - *g* can be used as a substitute for *f*
  - *g* inherits security properties of *f*

Applications at a Glance



Randomized Encoding - Syntax

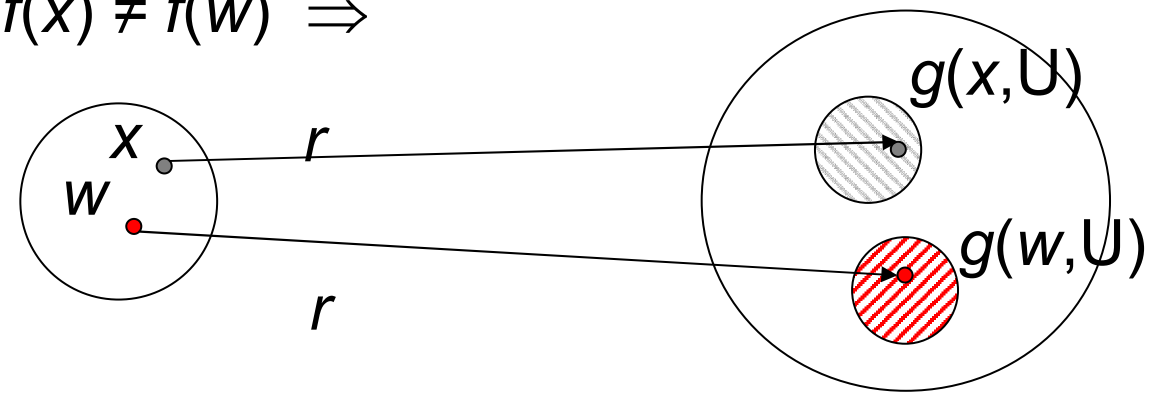


$f(x)$ is encoded by $g(x,r)$

Randomized Encoding - Semantics

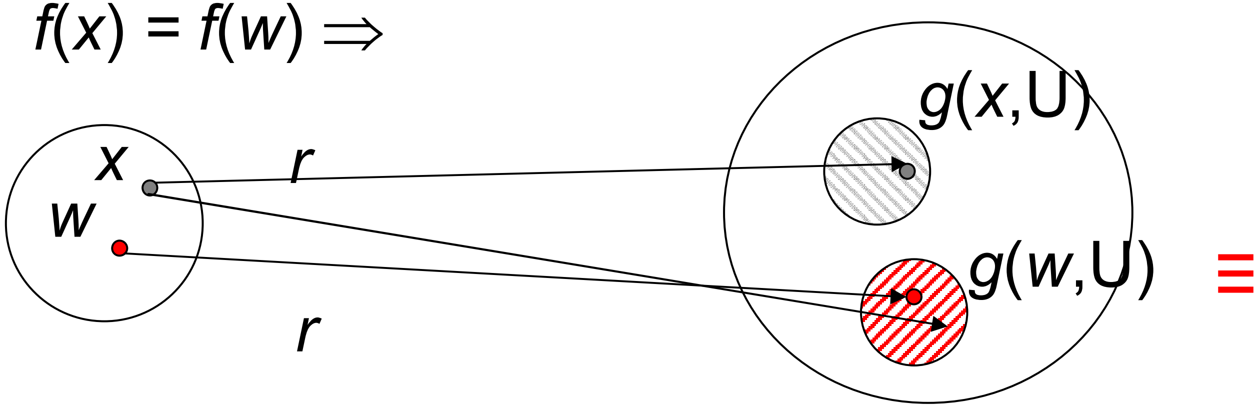
- Correctness: $f(x)$ can be efficiently decoded from $g(x,r)$.

  $f(x) \neq f(w) \Rightarrow$



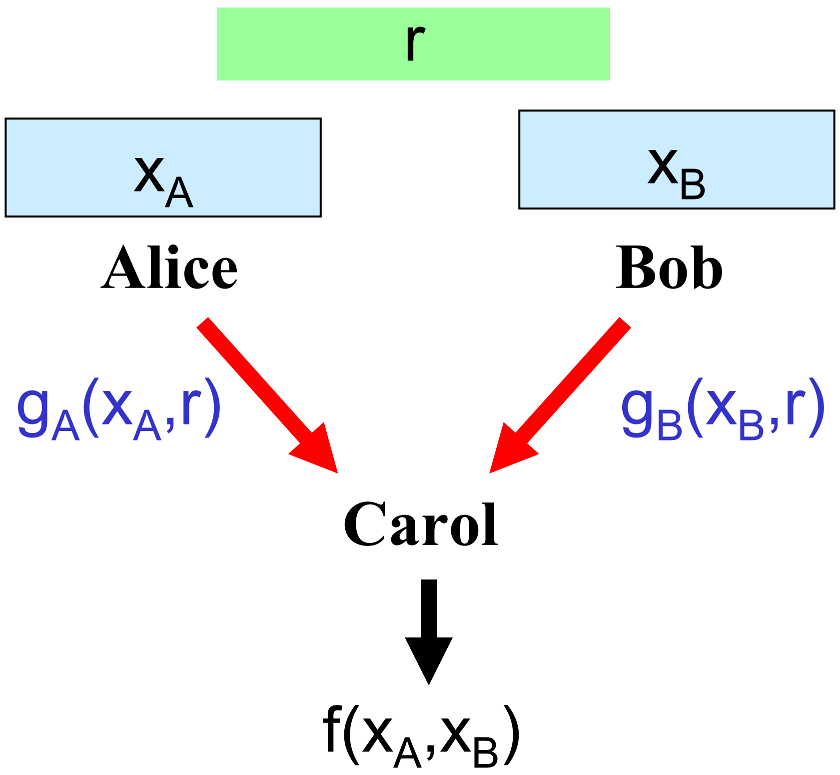
- Privacy: $\exists$ efficient simulator Sim such that Sim($f(x)$) $\equiv g(x,\mathrm{U})$
  - $g(x,\mathrm{U})$ depends only on $f(x)$

  $f(x) = f(w) \Rightarrow$



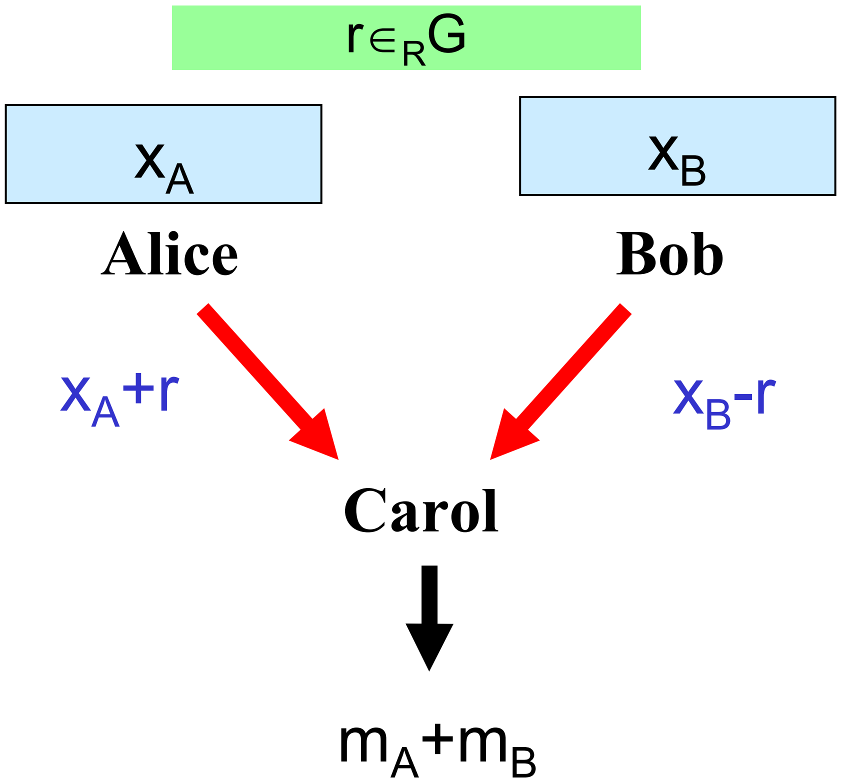
Notions of Simplicity - I

- Application: "minimal model for secure computation" [Feige-Kilian-Naor 94, …]
- 2-decomposability: $g((x_A,x_B),r)=(g_A(x_A,r),g_B(x_B,r))$



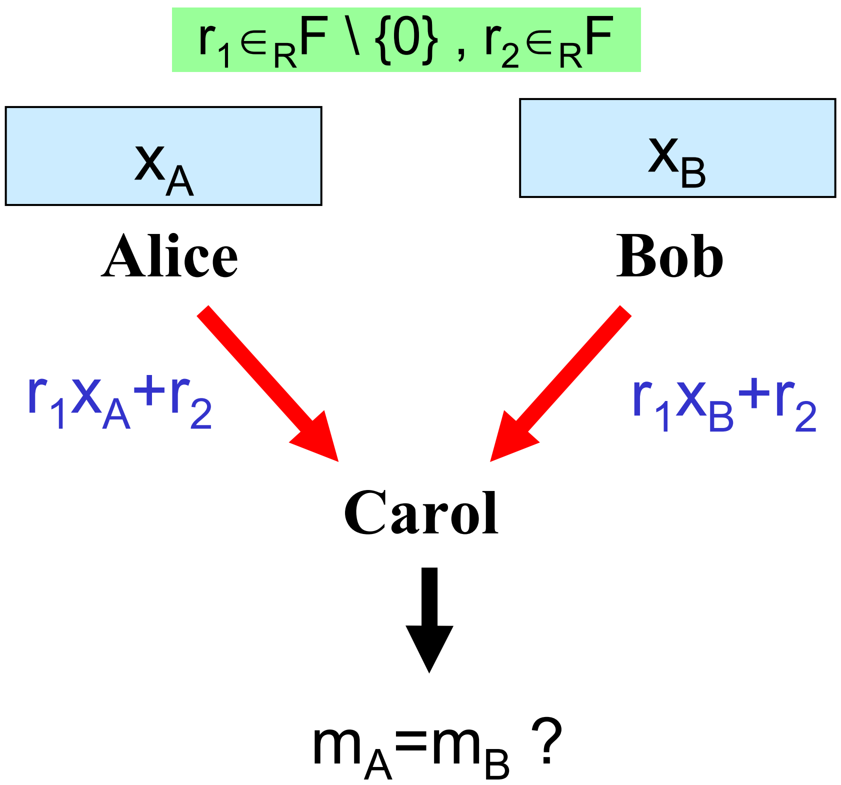
Example: sum

- $f(x_A,x_B) = x_A+x_B$ ($x_A,x_B \in$ finite group G)



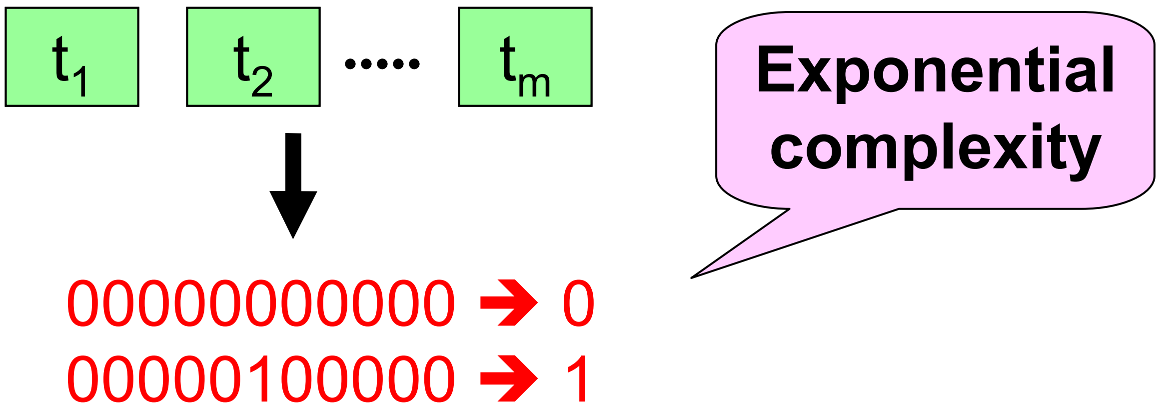
Example: equality

- $f(x_A,x_B)$ = equality ($x_A,x_B \in$ finite field F)



Example: ANY function

- $f(x_A,x_B) = x_A \wedge x_B$ $(x_A,x_B \in \{0,1\})$
  - Reduction to equality: $x_A \rightarrow 0/1$, $x_B \rightarrow 2/1$
- General boolean f: write as **disjoint** 2-DNF
  - $f(x_A,x_B) = \vee_{(a,b):f(a,b)=1} (x_A=a \wedge x_B=b) = t_1 \vee t_2 \vee \ldots \vee t_m$

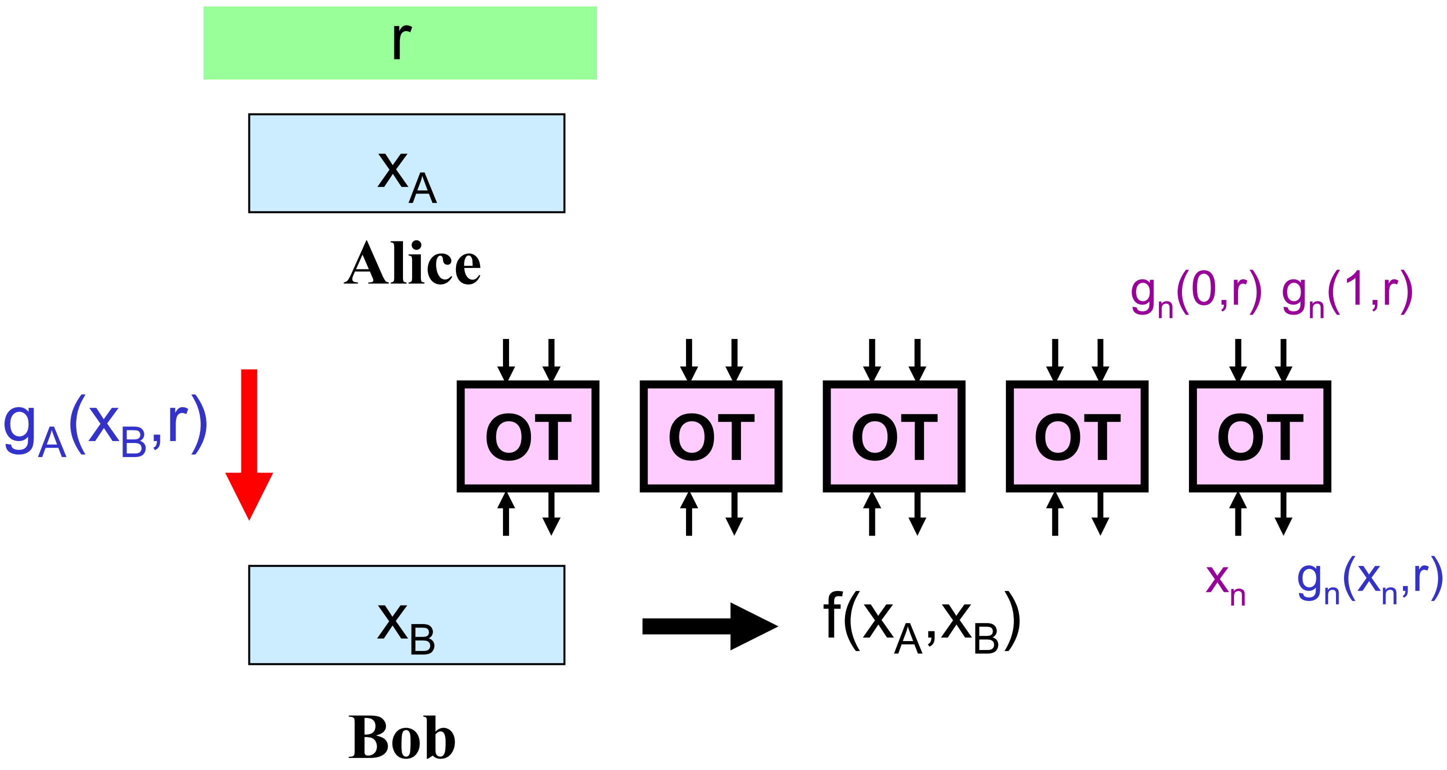


Notions of Simplicity - II

- Full decomposability:

  $g((x_1,\ldots,x_n),r)=(g_1(x_1,r),\ldots,g_n(x_n,r))$

  – Application: Basing SFE on OT [Kilian 88, ...]



Example: iterated group product

- Abelian case
  - $f(x_1,\ldots,x_n)=x_1+x_2+\ldots+x_n$
  - $g(x, (r_1,\ldots,r_{n-1})) =$

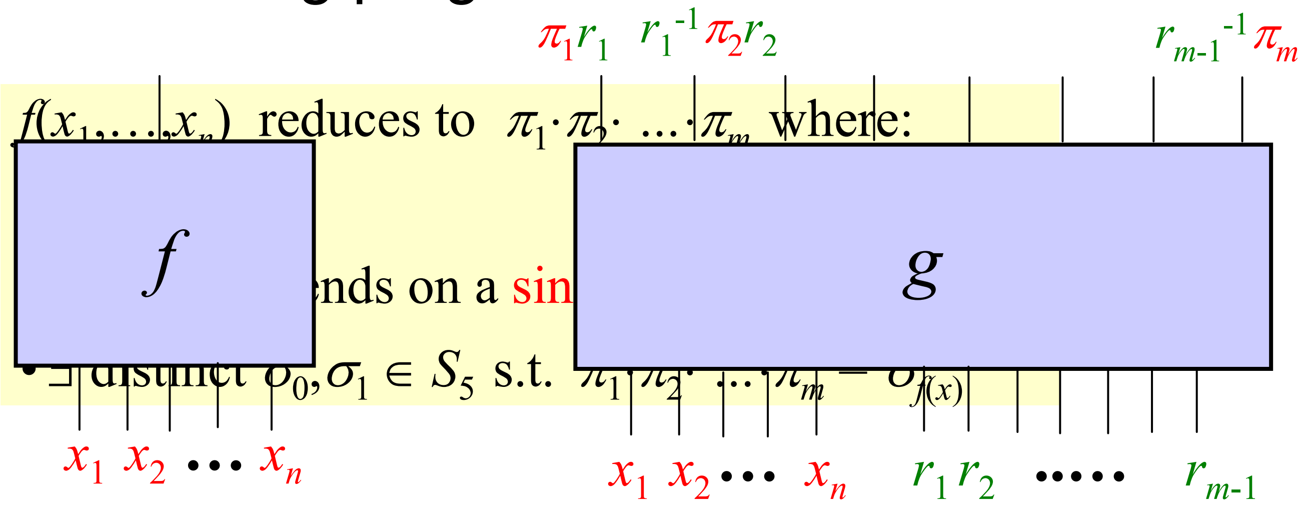
x_1+r_1 | x_2+r_2 | $\ldots$ | $x_{n-1}+r_{n-1}$ | $x_n-r_1-\ldots-r_{n-1}$

- General case [Kilian 88]
  - $f(x_1,\ldots,x_n)=x_1x_2\cdots x_n$
  - $g(x, (r_1,\ldots,r_{n-1})) =$

x_1r_1 | $r_1^{-1}x_2r_2$ | $r_2^{-1}x_2r_3$ | $\ldots$ | $r_{n-2}^{-1}x_{n-1}r_{n-1}$ | $r_{n-1}^{-1}x_n$

Example: iterated group product

Thm [Barrington 86]
Every boolean $f \in NC^1$ can be computed by a poly-length, width-5 branching program.



Encoding iterated group product

- Every output bit of g depends on just a single bit of x

$\pi_1\cdot\pi_2\cdot\pi_3\cdot\ldots\cdot\pi_m \Rightarrow \pi_1 r_1 \quad r_1^{-1}\pi_2 r_2 \quad r_2^{-1}\pi_3 r_3 \quad \ldots \quad r_{m-1}^{-1}\pi_m$

   ➢ Efficient fully decomposable encoding for every $f \in NC^1$

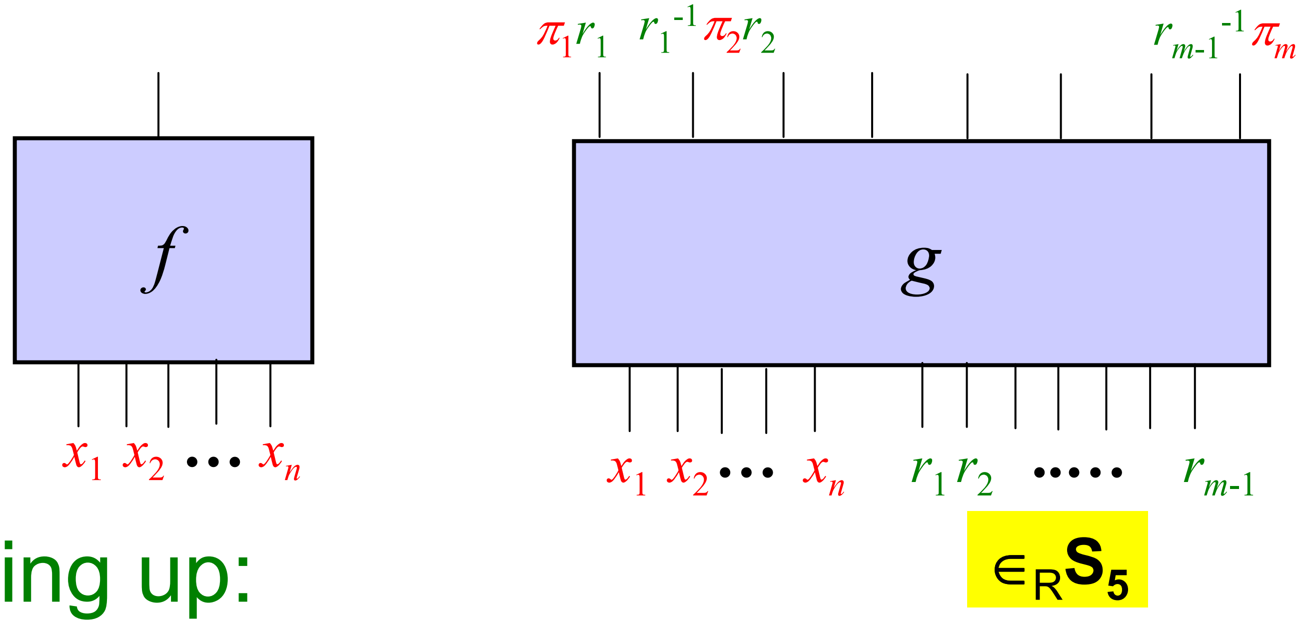
Notions of Simplicity - III

- Low degree: $g(x,r)$ = vector of degree-d poly in x,r over F
  - aka "Randomizing Polynomials" [IK00,…]
  - Application: round-efficient MPC

- Motivating observation:
  Low-degree functions are easy to distribute!
  - Round complexity of MPC protocols [BGW88,CCD88,CDM00,…]
    - Semi-honest model
      - $t<n/d$ ➔ 2 rounds
      - $t<n/2$ ➔ multiplicative depth + 1 = $\lceil \log d \rceil$+1 rounds
    - Malicious model
      - Optimal t ➔ $O(\log d)$ rounds

Examples

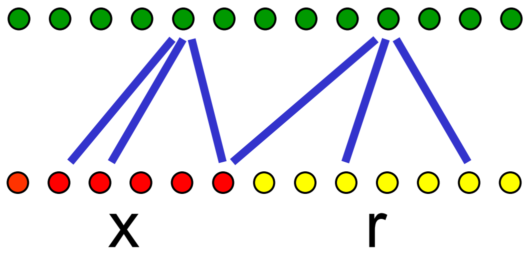
- What's wrong with previous examples?
  - Great degree in x (deg_x=1), bad degree in r



- Coming up:
  - Degree-3 encoding for every f
  - Efficient in size of branching program

Notions of Simplicity - IV

- Small locality:



  – Application: parallel cryptography!

[AIK04,AIK05,AIK07…]

- Coming up: encodings with locality 4
  – degree 3, fully decomposable
  – efficient in size of branching program

Parallel Cryptography

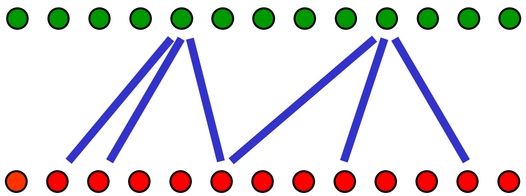
How low can we get?

poly-time
NC
log-space
NC^1
AC^0
NC^0

"Simplicity" of Cryptographic Primitives

- Can cryptographic primitives be computed by **very simple** functions?

NC^0_3

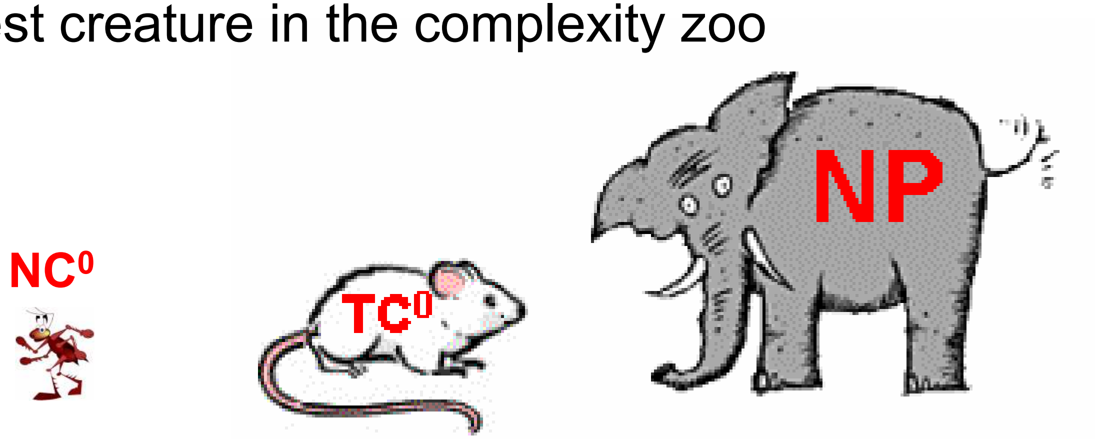


Simple = each output bit depends on $O(1)$ input bits

= const. depth circuits with bounded fan-in

= **NC^0**

- Currently the smallest creature in the complexity zoo



Cryptography in NC^0?

- Longstanding open question

  Håstad 87
  Impagliazzo Naor 89
  Goldreich 00
  Cryan Miltersen 01
  Krause Lucks 01
  Mossel Shpilka Trevisan 03

- Real-life motivation: super-fast cryptographic hardware

- Tempting conjecture:

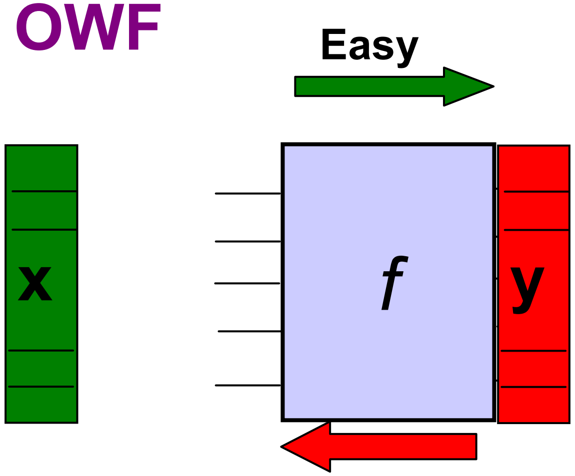
crypto hardness $\Rightarrow$ "complex" function

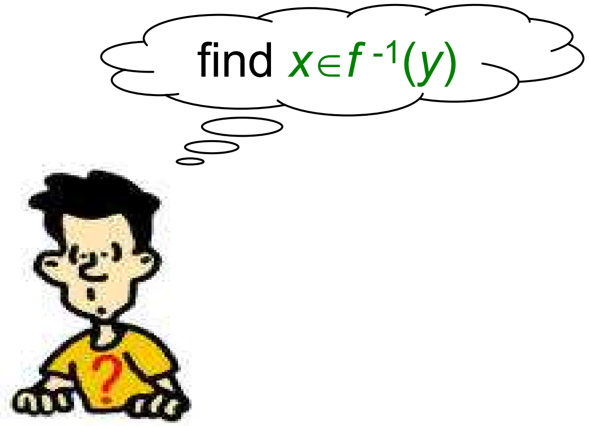
[CM]: Yes

[G]: No

Basic Primitives: One-way Function (OWF)

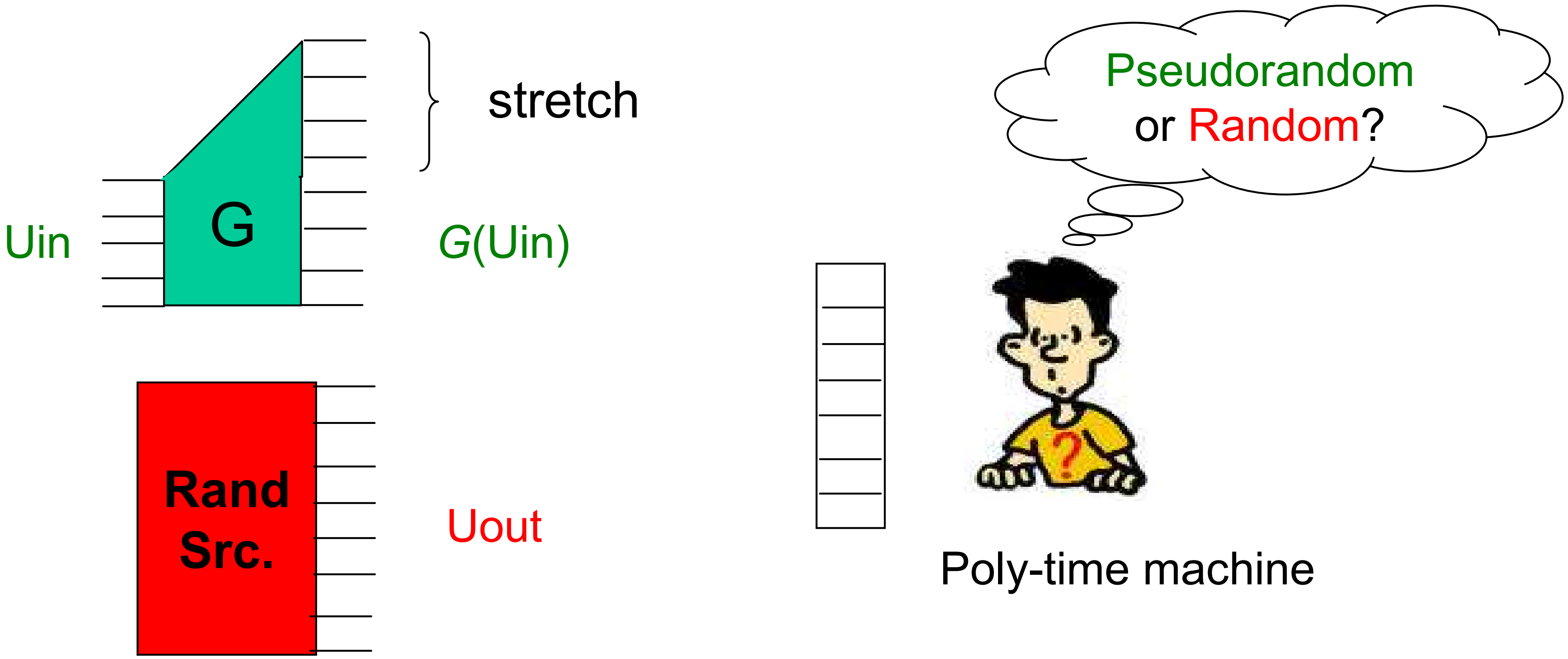
OWF





Poly-time machine

Basic Primitives: Pseudorandom Generator (PRG)



Def. PRG is minimal if stretch=1

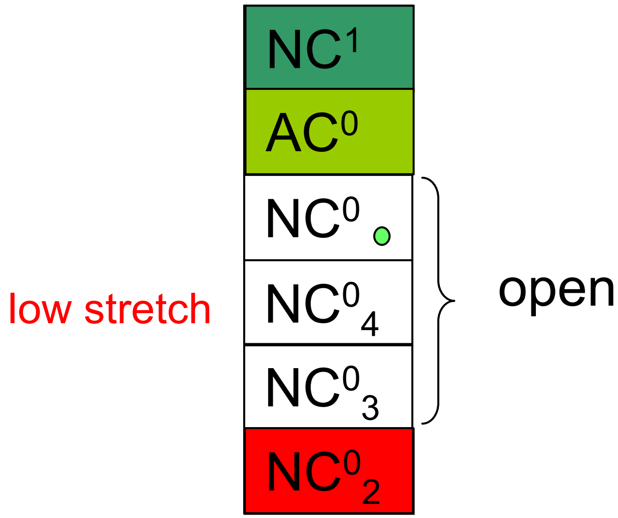
Previous Work

- Positive results
  - PRG in NC^1 from factoring, discrete-log, lattices…
  - PRF in NC^1 from factoring [Naor Reingold 97]
  - PRG (sub-lin stretch) in AC^0 from subset sum [Impagliazzo Naor 89]
- Negative results
  - No OWF in NC^0_2 [Goldreich 00, CryanMiltersen 01]
  - No PRG with large stretch in NC^0_3, NC^0_4 [CryanMiltersen01, MosselShpilkaTrevisan03]
- Permutation in NC^0 which is P-complete to invert [Håstad 87]
- Function in NC^0 which is NP-complete to invert [Agrawal Allender Rudich98]
- Heuristic construction of OWF/PRG in NC^0 [Goldreich 00 MST 03]

factoring, discrete-log, lattices, …

subset sum

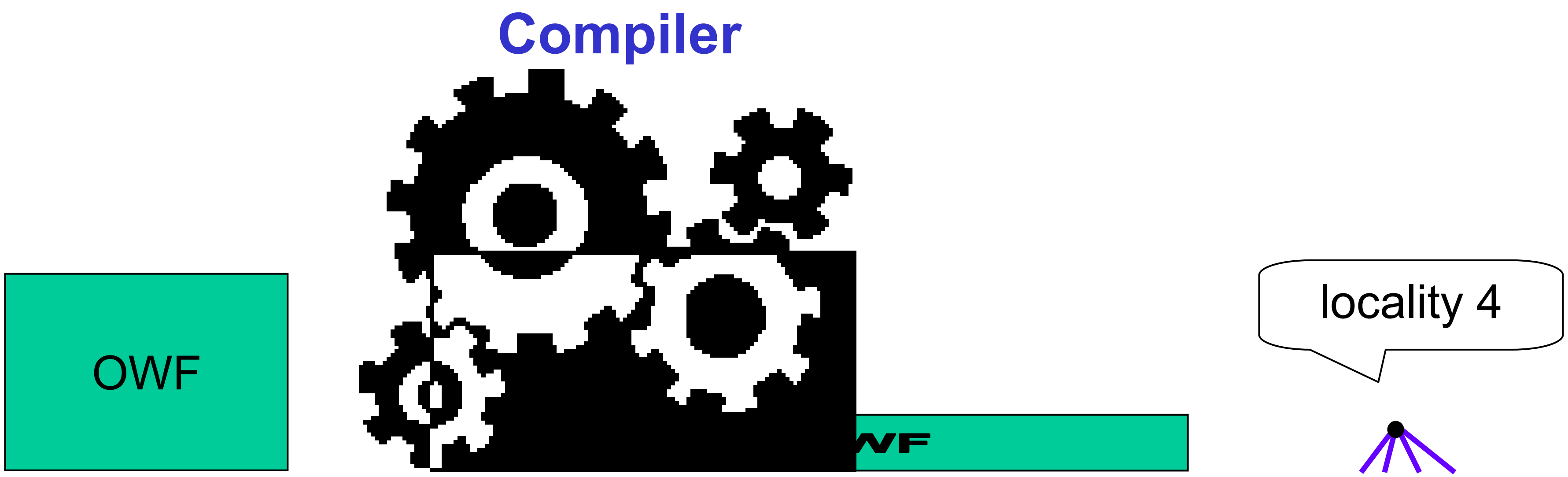
impossible



PRG / OWF

Our Approach

Compile primitives in a “relatively high” complexity class into ones in NC^0.



Sufficient Assumptions for Crypto in NC^0

Caveats:

- We g… with sub-linear st…
- decry… verification not in
  - In … ssible to decrypt/ver…
  - … commit in NC^0 with … in $NC^0[AND]$

Assuming min-PRG in NC^1

PRG
Hash

Sym-Enc
PK-Enc
Signature
Commit
NIZK

exist NC^1 $\Longrightarrow$ $\in NC^0$ [AIK04] [AIK05]

OWF
PRG
Hash

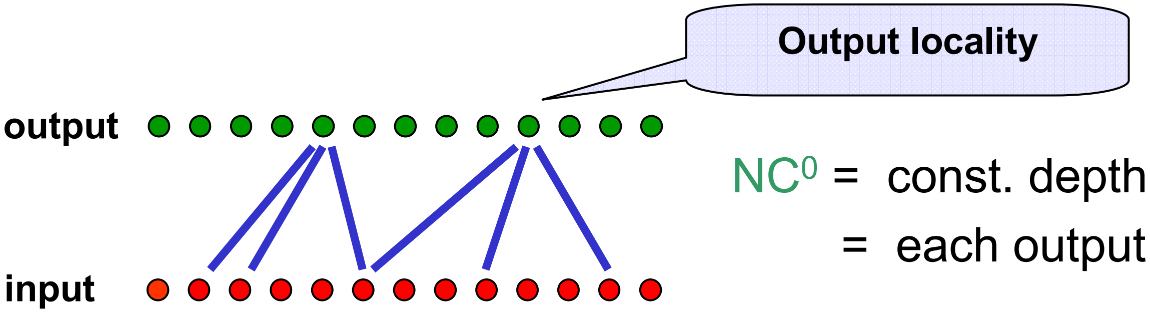
Sym-Enc
PK-Enc
Signature
Commit
NIZK

factoring, discrete-log/DDH, lattices, …

	OWF	PRG	Hash	Sym-Enc	PK-Enc	NI-Com	Sign	NIZK
P								factoring
NC^1								factoring
NC^0_4								factoring

Cryptography with Constant Input Locality

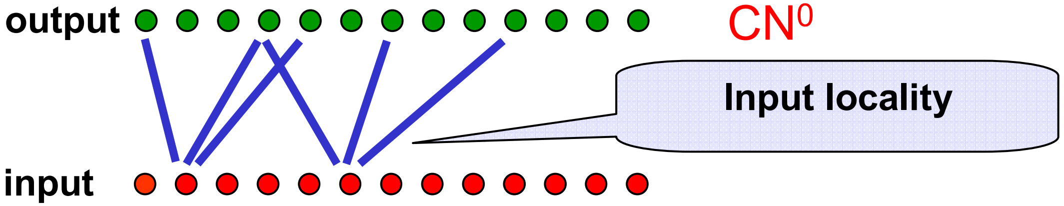
Till now we considered only NC^0 functions…



NC^0 = const. depth circuits with bounded fan-in

= each output bit depends on $O(1)$ input bits

Q: Can cryptographic primitives be realized by functions in which each input bit affects a constant number of output bits?



CN^0

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)