Randomized Encoding of Functions

Yuval Ishai

Technion and UCLA

Overview

- Can we make a computation simpler by just encoding the output?
- Question originally motivated by secure computation
- Answers have found applications in other areas of cryptography and elsewhere

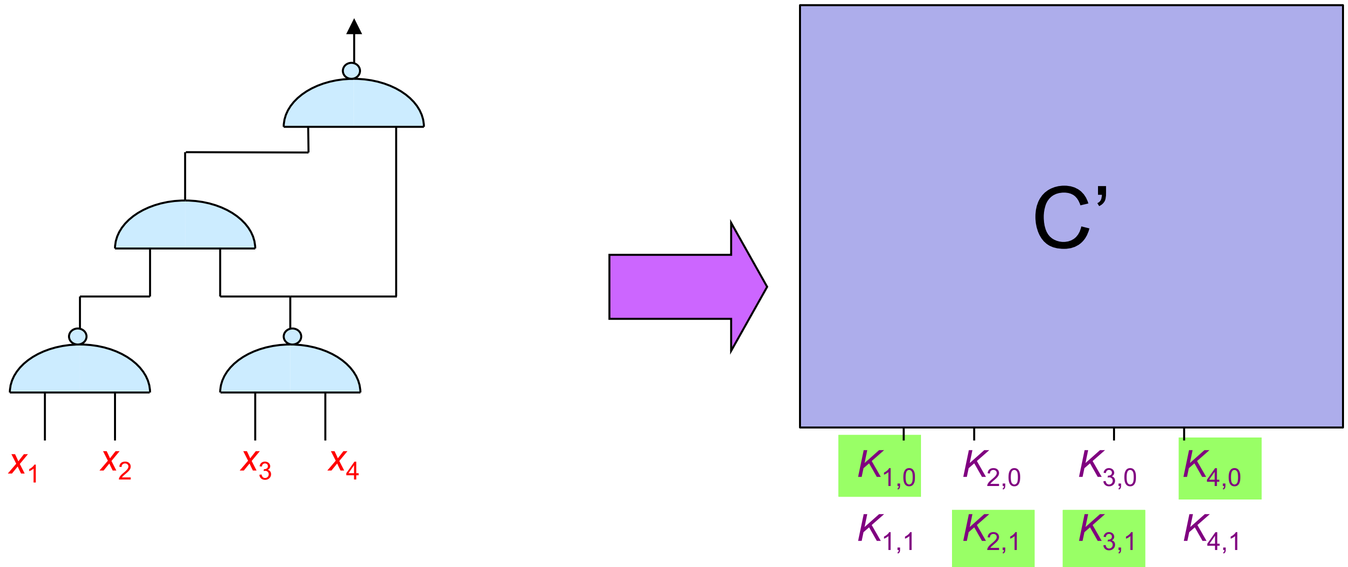
Garbled Circuit Construction

Yao, 1986

Garbled Circuit Construction

Circuit C

Garbled circuit C’



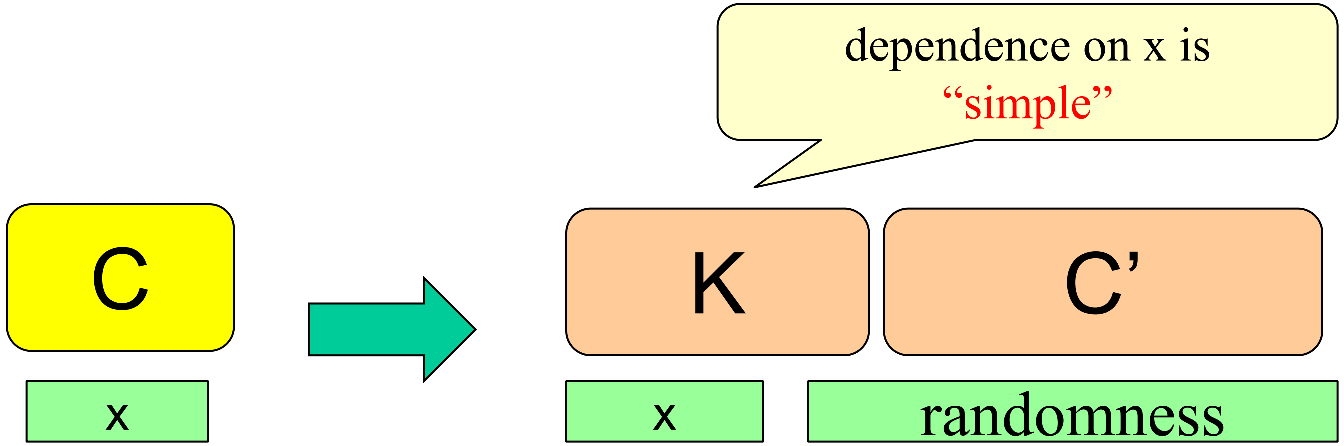
Pairs of short keys

$C(x)$ ⟷ C', K_{i,x_i}

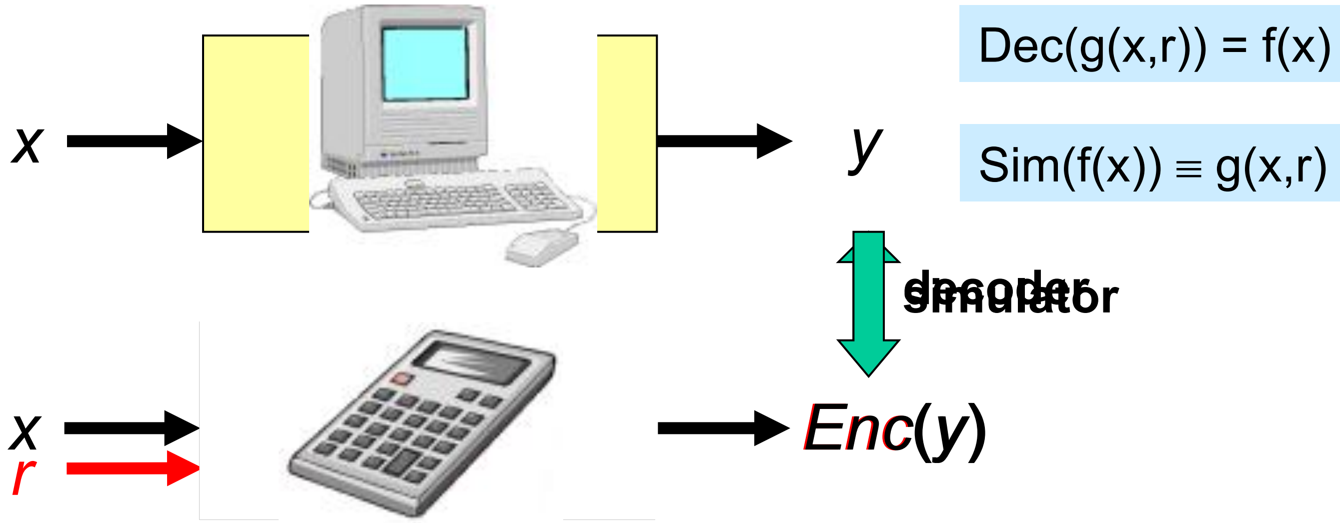
simulator

decoder

Even more abstractly…



The General 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*

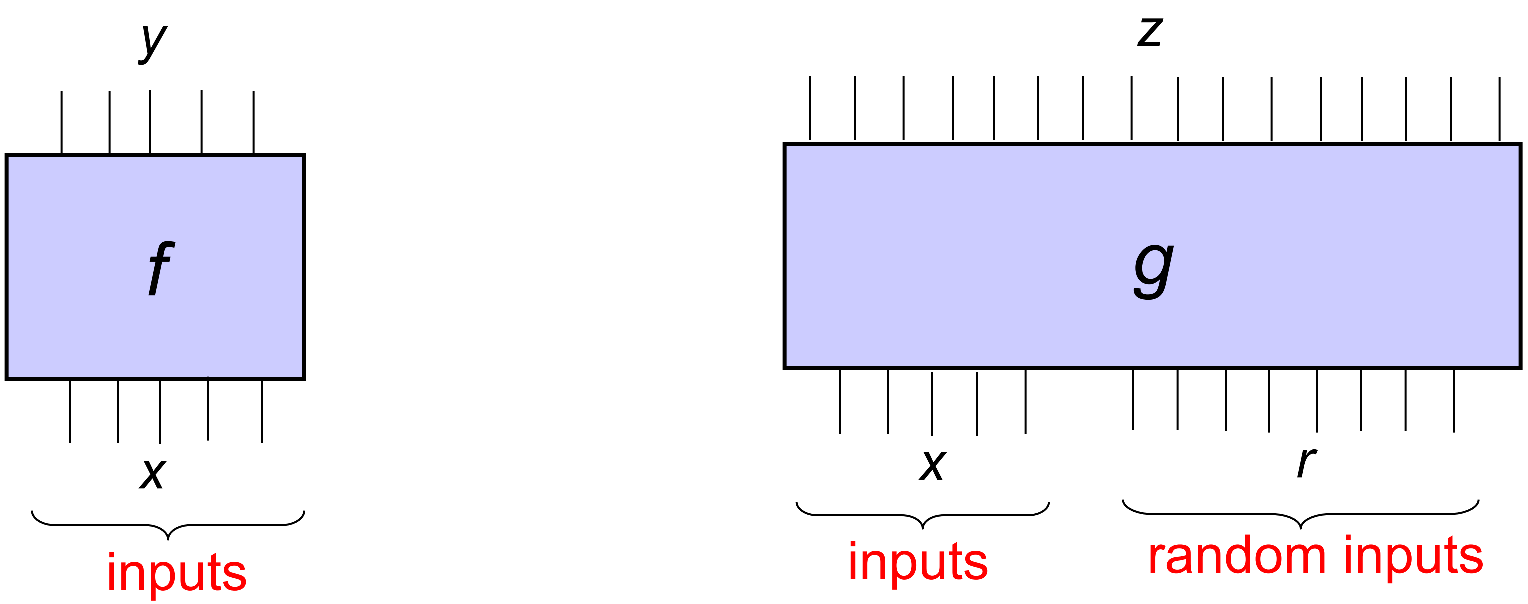
Applications

- Secure computation [Yao82…]
- Parallel cryptography [AIK04…]
- One-time programs [GKR08…]
- KDM-secure encryption [BHHI10…]
- Verifiable computation [GGP10…]
- Functional encryption [SS10…]
- …

Rest of Tutorial

- Constructions of randomized encodings
  - Different notions of simplicity
  - Different mathematical tools
    - Finite groups
    - Linear algebra
    - Number theory
  - Focus on information-theoretic security
    - Not in this tutorial: “succinct” and “reusable” variants
- Applications
  - Secure multiparty computation
  - Parallel cryptography

Randomized Encoding - Syntax

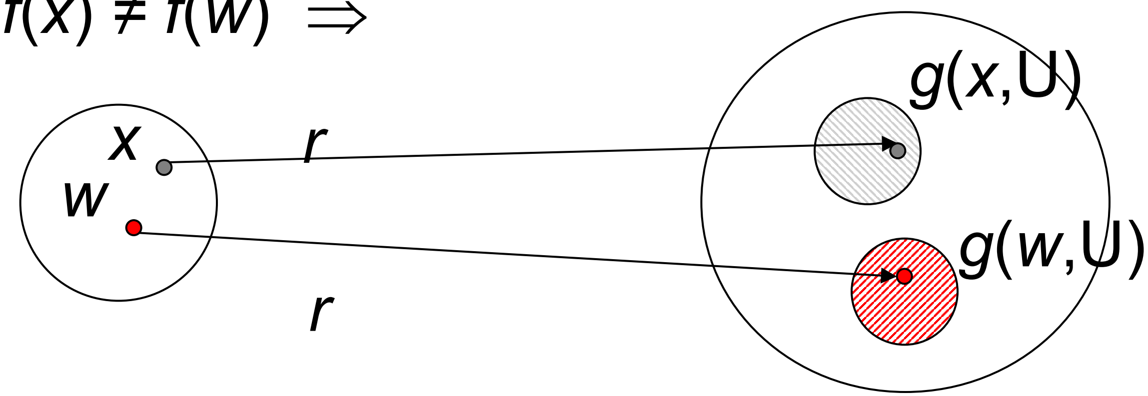


f(*x*) is encoded by *g*(*x*,*r*)

Randomized Encoding - Semantics

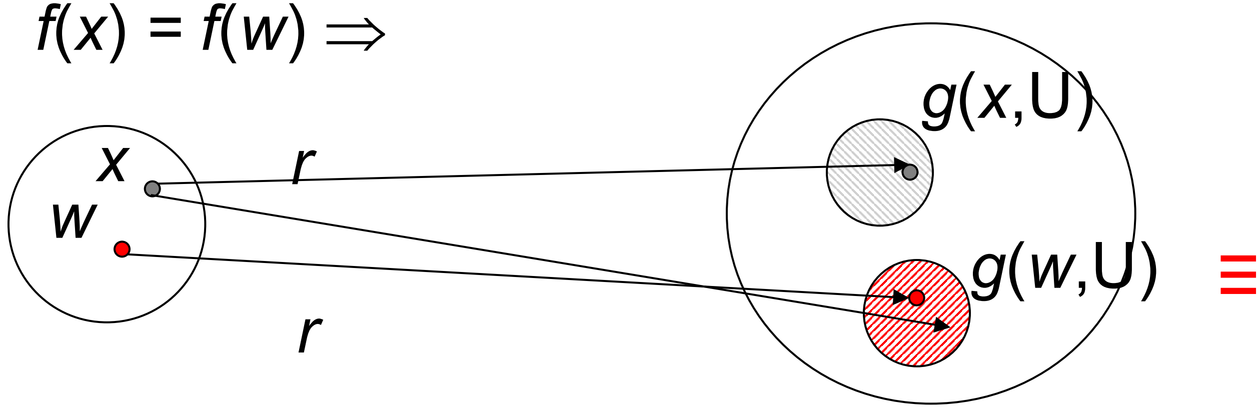
- Correctness: $f(x)$ can be efficiently decoded from $g(x,r)$.

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



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

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

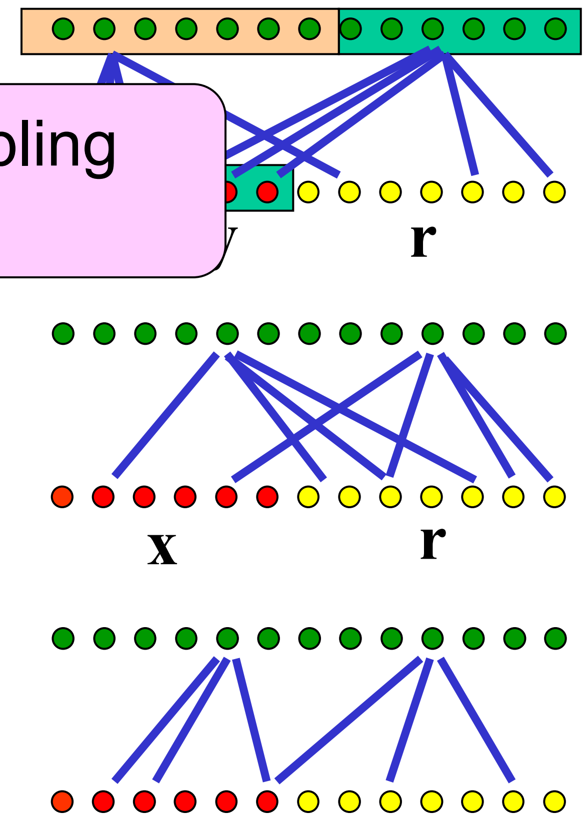


Notions of Simplicity

2-Decomposable encoding

$g((x,y),r)=(g_x(x,r),g_y$

AKA: projective garbling scheme [BHR12]



Decomposable encoding

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

NC^0 encoding

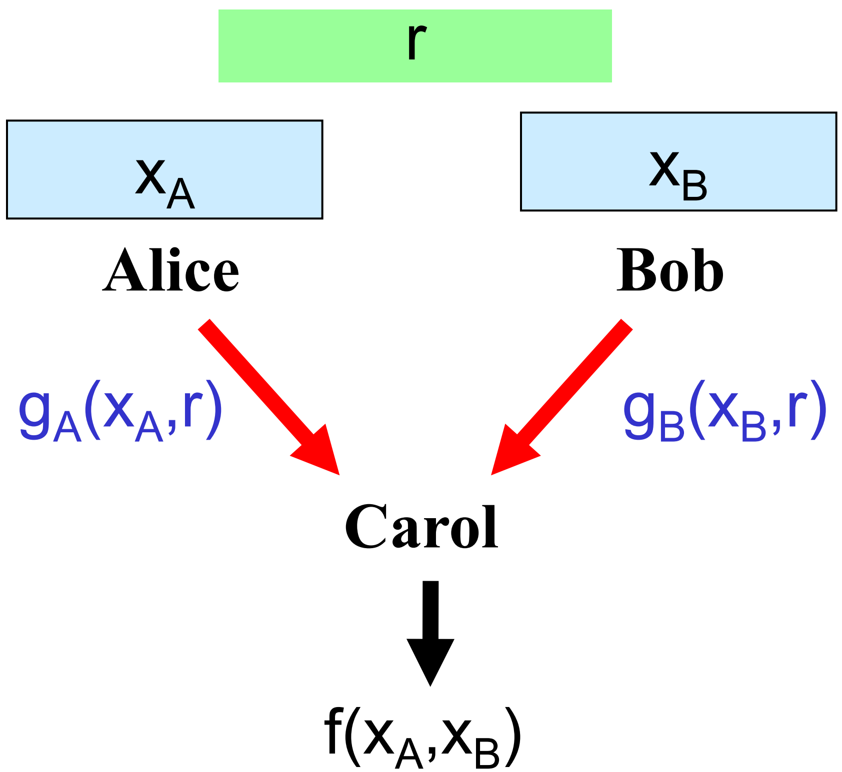
Output locality c

Low-degree encoding

Algebraic degree d over F

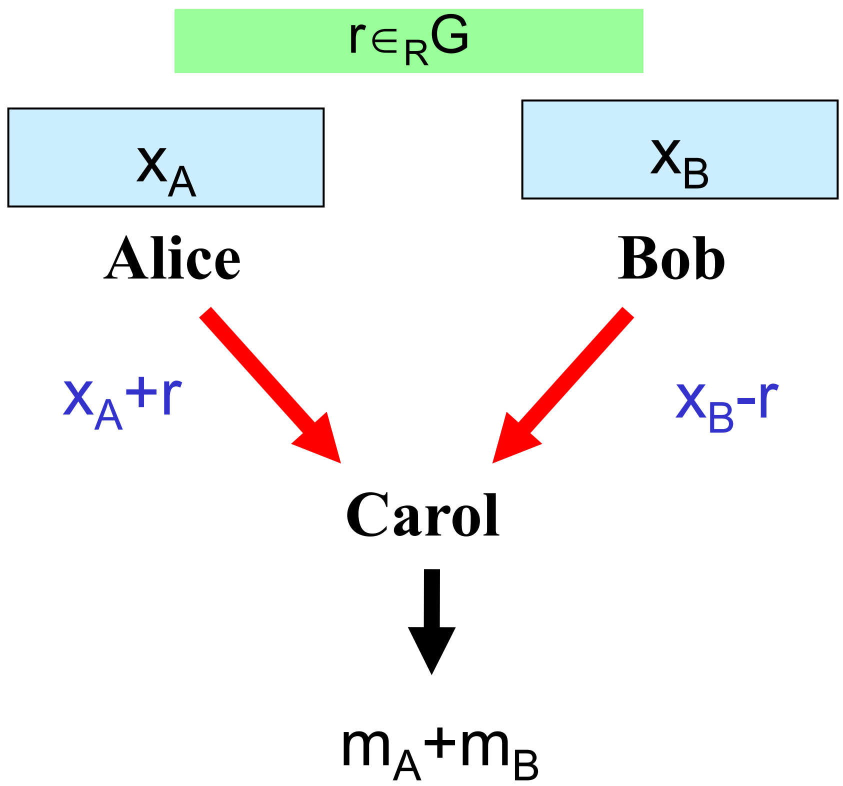
2-Decomposable Encodings

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



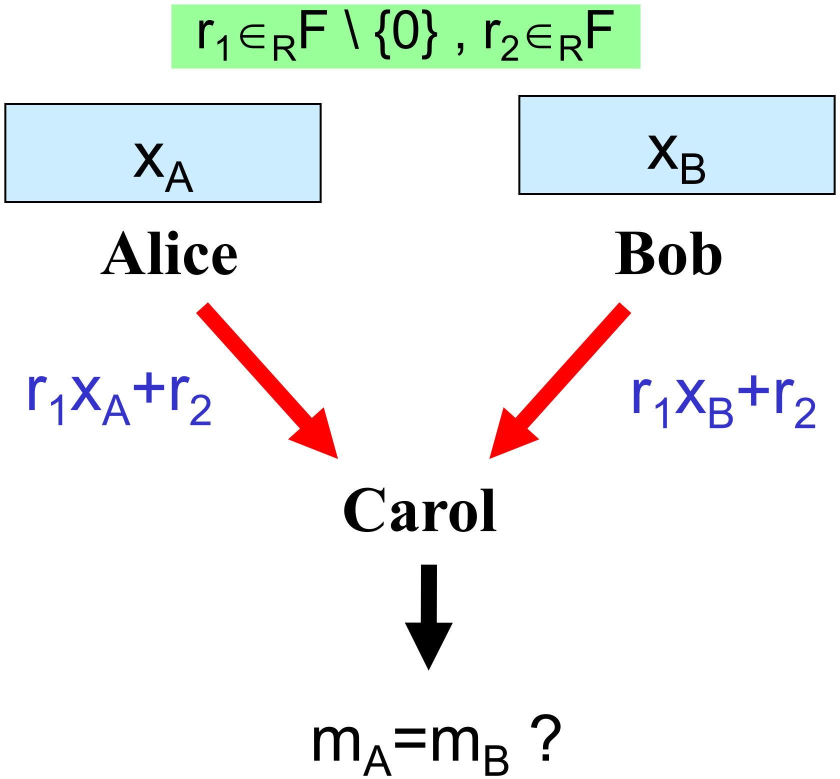
Example: sum

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



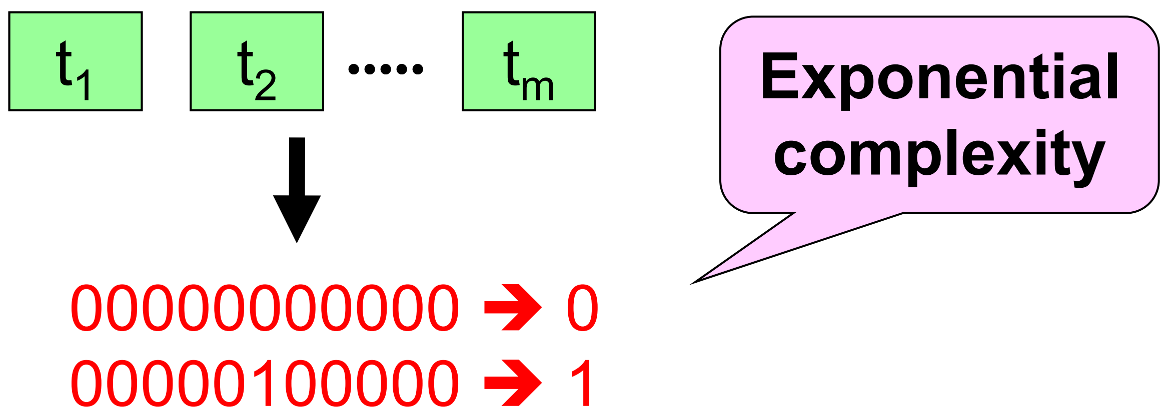
Example: equality

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



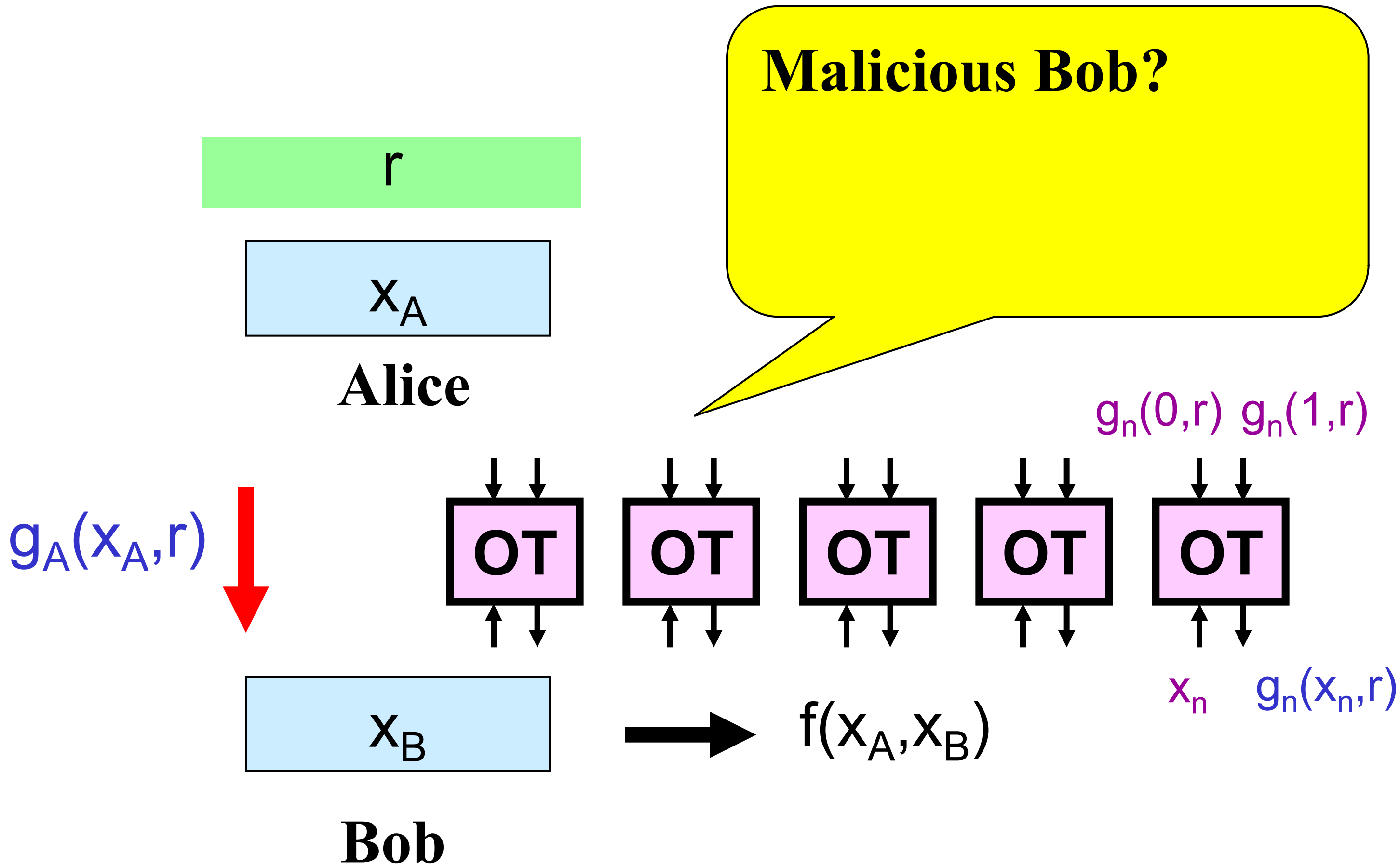
Example: ANY function

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



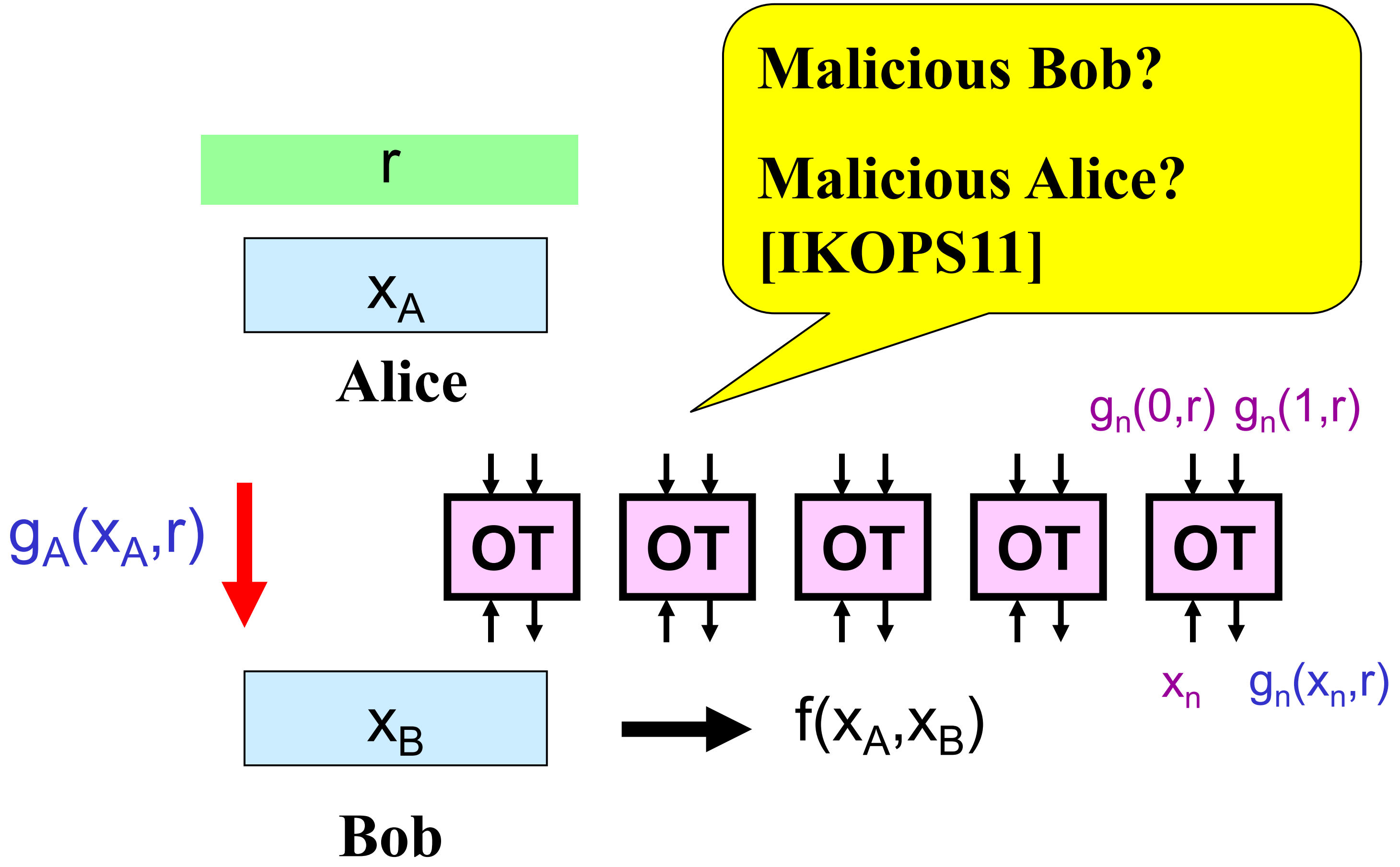
Decomposable Encodings

- Decomposability: $g((x_1,\ldots,x_n),r)=(g_1(x_1,r),\ldots,g_n(x_n,r))$
  - Application: Basing 2-PC on OT [Kilian 88, ...]



Decomposable Encodings

- Decomposability: $g((x_1,\ldots,x_n),r)=(g_1(x_1,r),\ldots,g_n(x_n,r))$
  - Application: Basing 2-PC on OT [Kilian 88, ...]



Example: iterated group product

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

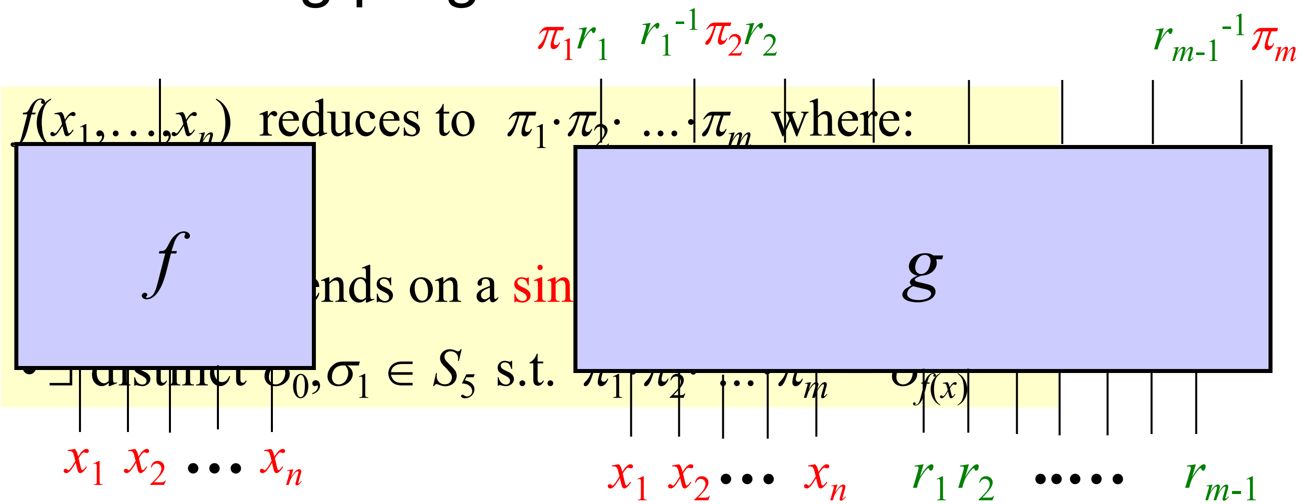
$$x_1+r_1 \quad x_2+r_2 \quad \ldots \quad x_{n-1}+r_{n-1} \quad x_n-r_1-\ldots-r_{n-1}$$

- General case [Kilian 88]
  - $f(x_1,\ldots,x_n)=x_1 x_2 \cdots x_n$
  - $g(x, (r_1,\ldots,r_{n-1})) =$

$$x_1 r_1 \quad r_1^{-1} x_2 r_2 \quad r_2^{-1} x_2 r_3 \quad \ldots \quad r_{n-2}^{-1} x_{n-1} r_{n-1} \quad r_{n-1}^{-1} x_n$$

Example: iterated group product

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



Encoding iterated group product

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

- Every output bit of g depends on just a single bit of x
  - ➢ Efficient decomposable encoding for every f∈NC[1]

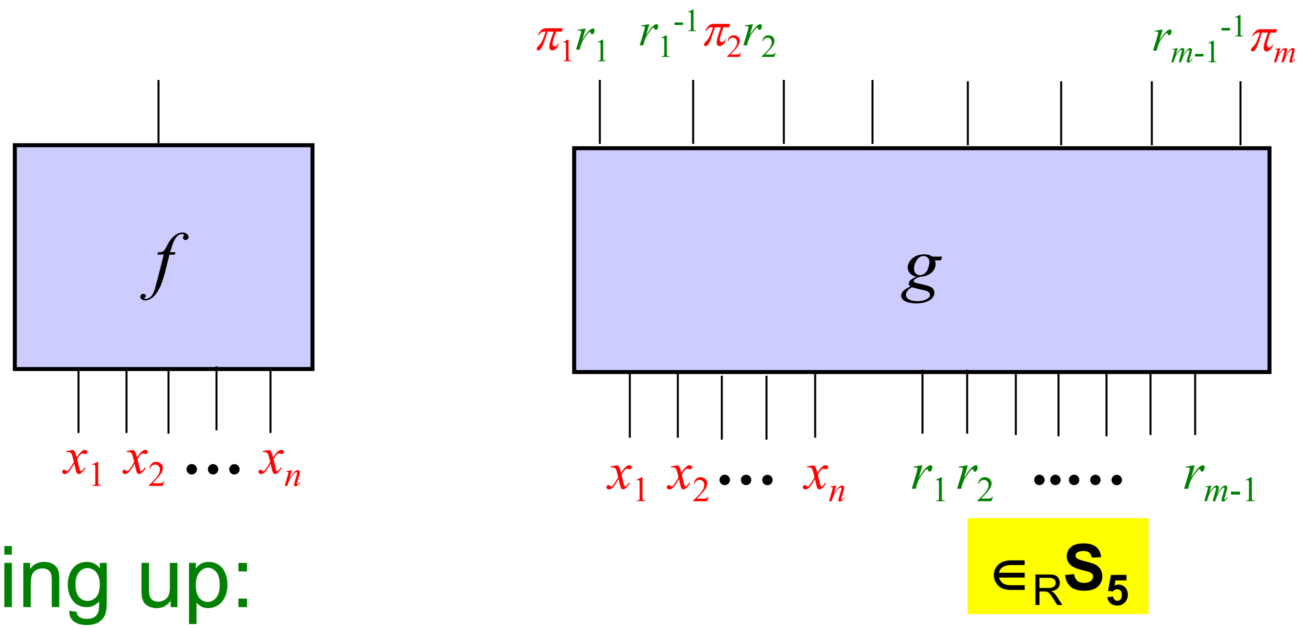
Low-Degree Encodings

- Low degree: g(x,r) = vector of degree-d poly in x,r over F
  - aka "Randomizing Polynomials" [I-Kushilevitz 00,…]
  - Application: round-efficient MPC

- Motivating observation:
  Low-degree functions are easy to distribute!
  - Round complexity of MPC protocols [GMW87,BGW88,CCD88,…]
    - Semi-honest (passive) adversary:
      - $t<n$ using ideal OT ➔ O(log d) rounds
      - $t<n/d$ ➔ 2 rounds
      - $t<n/2$ ➔ multiplicative depth + 1 = $\lceil \log d \rceil$+1 rounds
    - Malicious (active) adversary:
      - Optimal t ➔ O(log d) rounds

Examples

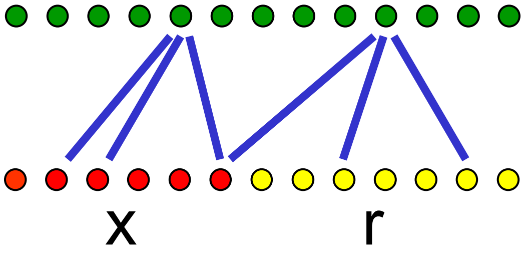
- What's wrong with previous examples?
  - Great degree in x (deg_x=1), bad degree in r



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

Local Encoding

- Small output locality:



  – Application: parallel cryptography!

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

Parallel Cryptography

How low can we get?

poly-time

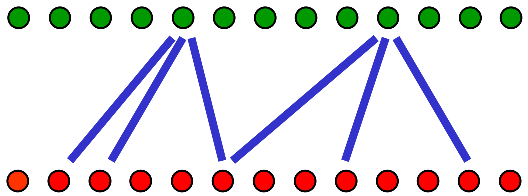
NC

log-space

NC^1

AC^0

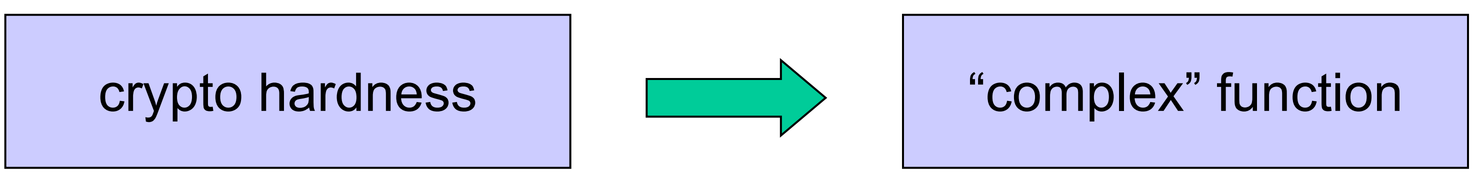
NC^0



Cryptography in NC^0?

- Real-life motivation: fast cryptographic hardware

- Tempting conjecture:

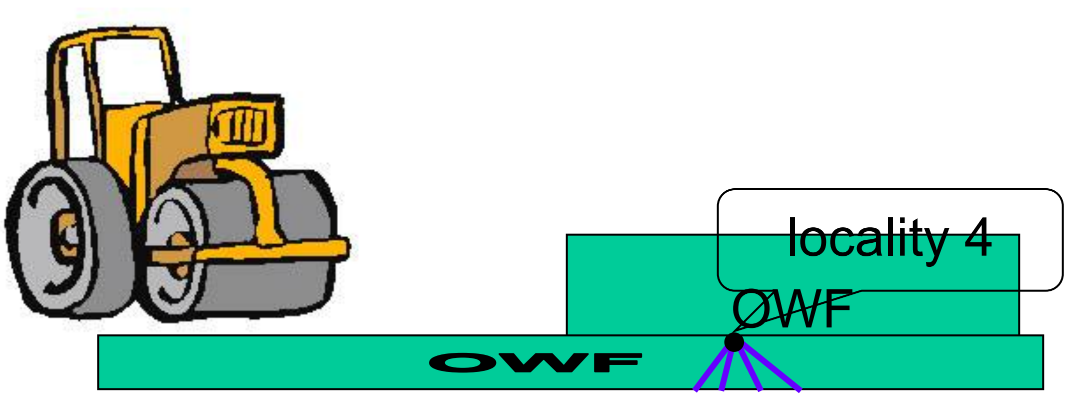


Surprising Positive Result [AIK04]

Compile primitives in a "relatively high" complexity class (e.g., NC^1, NL/poly, ⊕L/poly) into ones in NC^0.

NC^1 cryptography implied by factoring, discrete-log, lattices…

⇒ essentially settles the existence of cryptography in NC^0



Remaining Challenge

Coming up…

How to encode “complex” f by g ∈

- Observation: enough to obtain const. degree encoding

- Locality Reduction:
  degree 3 poly over GF(2) ⇒ locality 4 rand. encoding

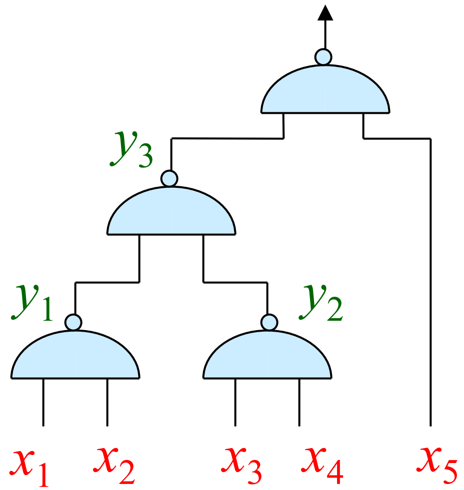
$$f(x) = T_1(x) + T_2(x) + \ldots + T_k(x)$$

$$g(x,r,s) = \quad T_1(x)+r_1 \qquad T_2(x)+r_2 \qquad \cdots \qquad T_k(x)+r_k$$

$$-r_1+s_1 \qquad -s_1-r_2+s_2 \qquad \cdots \qquad -s_{k-1}-r_k$$

3 Ways to Degree 3

1. Degree-3 encoding using a circuit representation



$f(x)=1 \quad \Leftrightarrow \quad$

$\exists\, y_1, y_2, y_3$

$y_1 = \text{NAND}(x_1, x_2) = x_1(1-x_2)+(1-x_1)x_2+(1-x_1)(1-x_2)$

$y_2 = \text{NAND}(x_3, x_4)$

$y_3 = \text{NAND}(y_1, y_2)$

$1 = \text{NAND}(y_3, x_5)$

Note: $\quad \Rightarrow \quad \exists!\, y_1, y_2, y_3$

Using circuit representation (contd.)

$$\left.\begin{array}{c} q_1(x,y)=0 \\ q_2(x,y)=0 \\ \dots \\ q_s(x,y)=0 \end{array}\right\} \text{deg.-2}$$

$$g(x, y,r)=\Sigma\, r_i\, q_i(x,y) \quad \} \text{ deg.-3}$$

$f(x)=0 \;\Rightarrow\; g(x,y,r)$ is uniform

$f(x)=1 \;\Rightarrow\; g(x,y,r) \equiv 0$ given $y=y_0$, otherwise it is uniform

Statistical distance amplified to 1/2 by $2^{\Theta(s)}$ repetitions.

•works over any field

•complexity exponential in circuit size

2. Degree-3 encoding using quadratic characters

Fact from number theory:

$$\forall N \ \ \forall \text{bit-sequence} \ \ b \in \{0,1\}^N$$

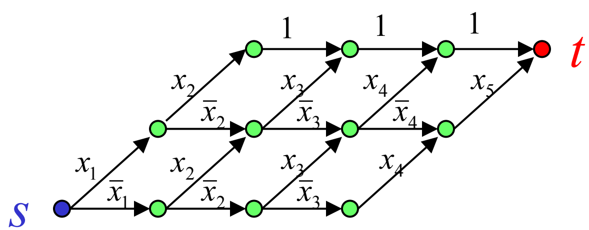
$$\exists \text{prime} \ \ q(=2^{O(N)}) \ \ \exists d > 0 \ \text{ such that } \ b = \chi_q(d)\chi_q(d+1)\cdots\chi_q(d+N-1)$$

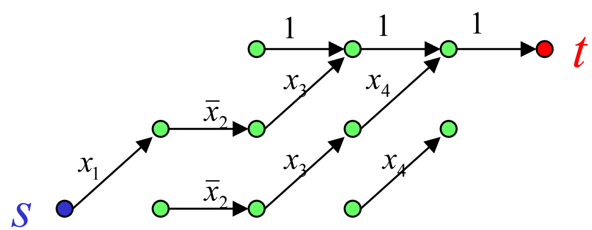
- Let $N=2^n$, b = length-N truth-table of f, F=GF(q)
- Define $p(x_1,\ldots,x_n, r) = \left(d + \sum_{i=1}^{n} 2^{i-1} x_i\right) \cdot r^2$

- one polynomial
- huge field size

3. Perfect Degree-3 Encoding from Branching Programs

BP=(G, s , t, edge-labeling) G_x=subgraph induced by x



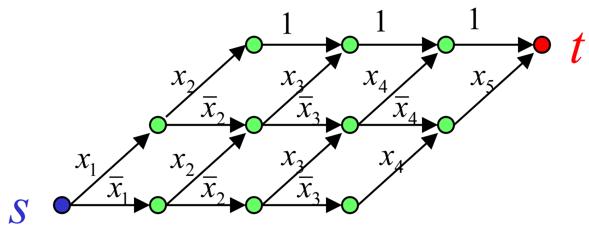


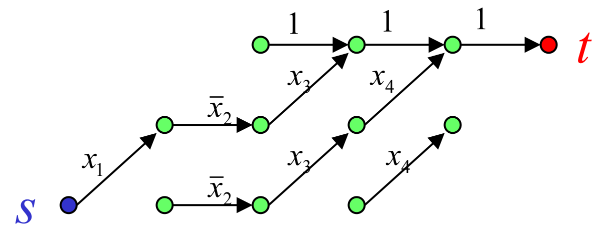
mod-q NBP: $f(x)$ = # s-t paths in G_x (mod q)

- size = # of vertices
- circuit-size ≤ BP-size ≤ formula-size
- Boolean case: q=2.
  - Captures complexity class ⊕L/poly

3. Perfect Degree-3 Encoding from Branching Programs

BP=(G, s , t, edge-labeling) G_x=subgraph induced by x





- BP(x)=det(L(x)), where L is a degree-1 mapping which outputs matrices of a special form.

$$\begin{matrix} * & * & * & * \\ -1 & * & * & * \\ 0 & -1 & * & * \\ 0 & 0 & -1 & * \end{matrix}$$

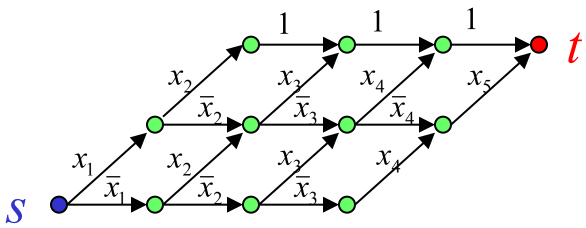
- Encoding:

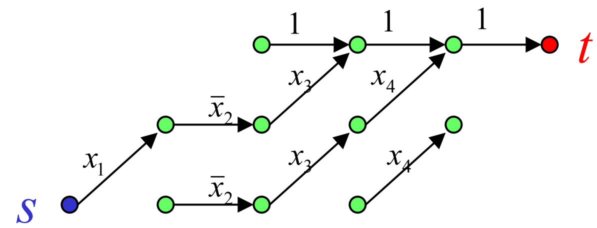
$$g(x,r_1,r_2)= R_1(r_1)\cdot L(x)\cdot R_2(r_2)$$

$$R_1(r_1): \begin{matrix} 1 & \$ & \$ & \$ \\ 0 & 1 & \$ & \$ \\ 0 & 0 & 1 & \$ \\ 0 & 0 & 0 & 1 \end{matrix} \qquad L(x): \begin{matrix} * & * & * & * \\ -1 & * & * & * \\ 0 & -1 & * & * \\ 0 & 0 & -1 & * \end{matrix} \qquad R_2(r_2): \begin{matrix} 1 & 0 & 0 & \$ \\ 0 & 1 & 0 & \$ \\ 0 & 0 & 1 & \$ \\ 0 & 0 & 0 & 1 \end{matrix}$$

Perfect Degree-3 Encoding of BPs

BP=(G, s, t, edge-labeling) G_x=subgraph induced by x





Encoding based on Lemma: $g(x,r_1,r_2)= R_1(r_1)\cdot L(x)\cdot R_2(r_2)$

mod-q BP: $f(x) = \#\ s\rightarrow t$ paths in G_x mod q.

$$\begin{pmatrix} 1 & \$ & \$ & \$ \\ 0 & 1 & \$ & \$ \\ 0 & 0 & 1 & \$ \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \begin{pmatrix} * & * & * & * \\ -1 & * & * & * \\ 0 & -1 & * & * \\ 0 & 0 & -1 & * \end{pmatrix} \quad \begin{pmatrix} 1 & 0 & 0 & \$ \\ 0 & 1 & 0 & \$ \\ 0 & 0 & 1 & \$ \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

size(BP)

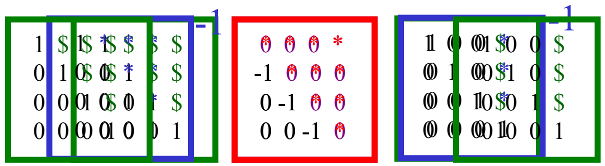
Lemma: $\exists$ degree-1 mapping $L : x \rightarrow \begin{pmatrix} * & * & * & * \\ -1 & * & * & * \\ 0 & -1 & * & * \\ 0 & 0 & -1 & * \end{pmatrix}$ s.t. $\det(L(x))= f(x)$.

Correctness: $f(x)=\det g(x,r_1,r_2)$

det $L(x)$ ($= f(x)$)

Privacy: $\begin{pmatrix} 1 & * & * & * \\ 0 & 1 & * & * \\ 0 & 0 & 1 & * \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} * & * & * & * \\ -1 & * & * & * \\ 0 & -1 & * & * \\ 0 & 0 & -1 & * \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & * \\ 0 & 1 & 0 & * \\ 0 & 0 & 1 & * \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & * \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \end{pmatrix}$

$g(x,r_1,r_2) \equiv$



Proof of Lemma

Lemma: $\exists$ degree-1 mapping $L : x \rightarrow \begin{pmatrix} * & * & * & * \\ -1 & * & * & * \\ 0 & -1 & * & * \\ 0 & 0 & -1 & * \end{pmatrix}$ s.t. $\det(L(x)) = f(x)$.

Proof:

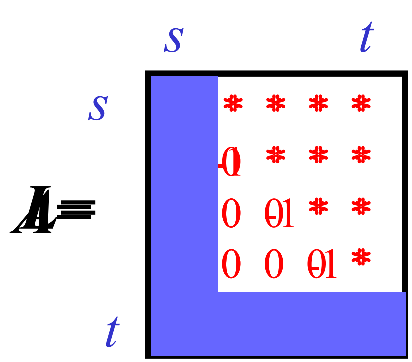
$A(x)$= adjacency matrix of G_x (over F=GF(q))

$A^* = I + A + A^2 + \ldots = (I - A)^{-1}$

$A^*_{s,t} = \ (-1)^{s+t} \cdot det\ (I-A)|_{t,s} \,/\, det\ (I-A)$

$= det\ (A-I)|_{t,s}$

$L(x) = (A(x)-I)|_{t,s}$



Thm. size-s BP $\Rightarrow$ degree 3 encoding of size $O(s^2)$

- perfect encoding for mod-q BP (capturing $\oplus$L/poly for q=2)
- imperfect for nondeterministic BP (capturing NL/poly)

The secure evaluation of an arbitrary function can be reduced to the secure evaluation of degree-3 polynomials.

Round complexity of information-theoretic MPC in semi-honest model:

- How many rounds for maximal privacy? — 3 rounds suffice
- How much privacy in 2 rounds? — $t<n/3$ suffices

- perfect privacy + correctness
- complexity $O(\text{BP-size}^2)$

Is 3 minimal?

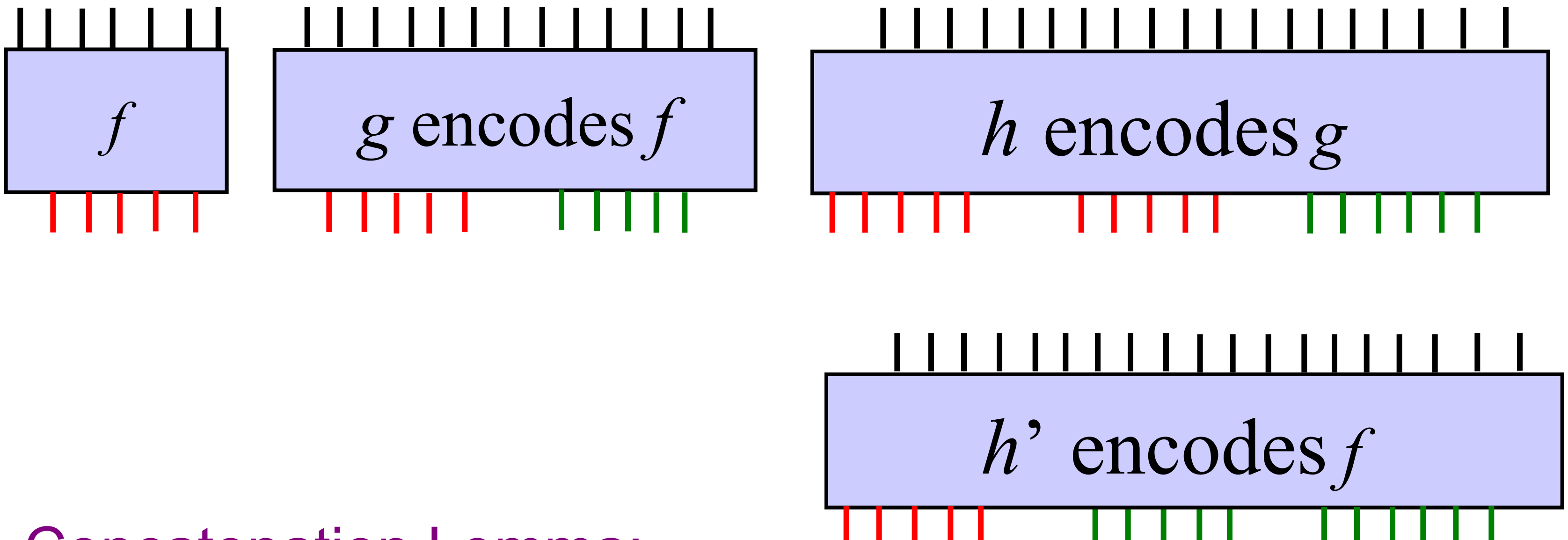
Thm. [IK00]

A boolean function f admits a *perfectly private* degree-2 encoding over F if and only if *either:*

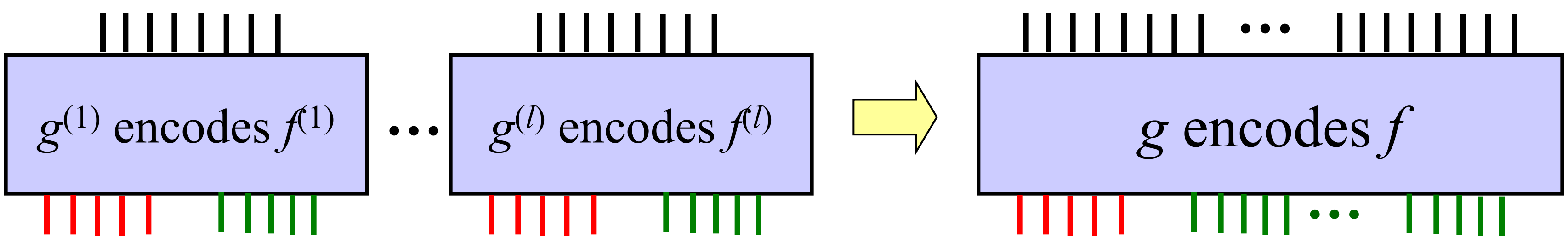
- f or its negation test for a linear condition $Ax=b$ over F;
- f admits standard representation by a degree-2 polynomial over F.

Wrapping Up

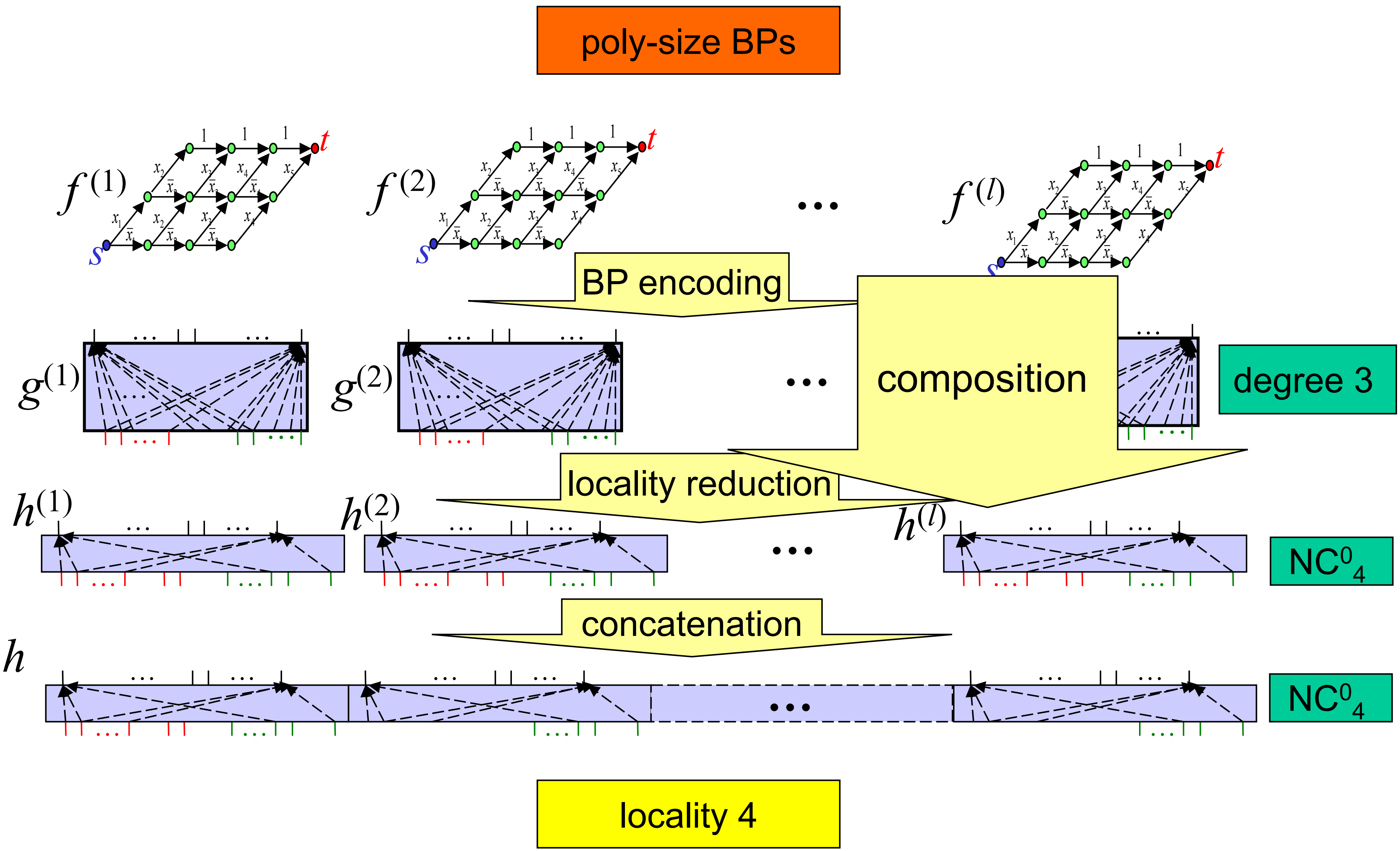
Composition Lemma:



Concatenation Lemma:



From Branching Programs to Locality 4



Summary

- Different flavors of randomized encoding
  - Motivated by different applications
- "Simplest" encodings: outputs of form $x_i r_j r_k + r_h$
  - Efficient perfect/statistical encodings for various complexity classes (NC^1, NL/poly, mod$_q$L/poly)
  - (Efficient computationally private encodings for all P, assuming "Easy PRG".)

Open Questions

Randomized encoding	MPC	Parallel crypto
poly-size NC^0 encoding for every $f \in P$?	Unconditionally secure constant-round protocols for every $f \in P$?	$\exists$OWF ➔ $\exists$OWF in NC^0?
locality 3 for every f?	maximal privacy with minimal interaction?	$\exists$OWF in NC^1 ➔ $\exists$OWF in NC^0_3?
better encodings?	better constant-round protocols?	practical hardware?