

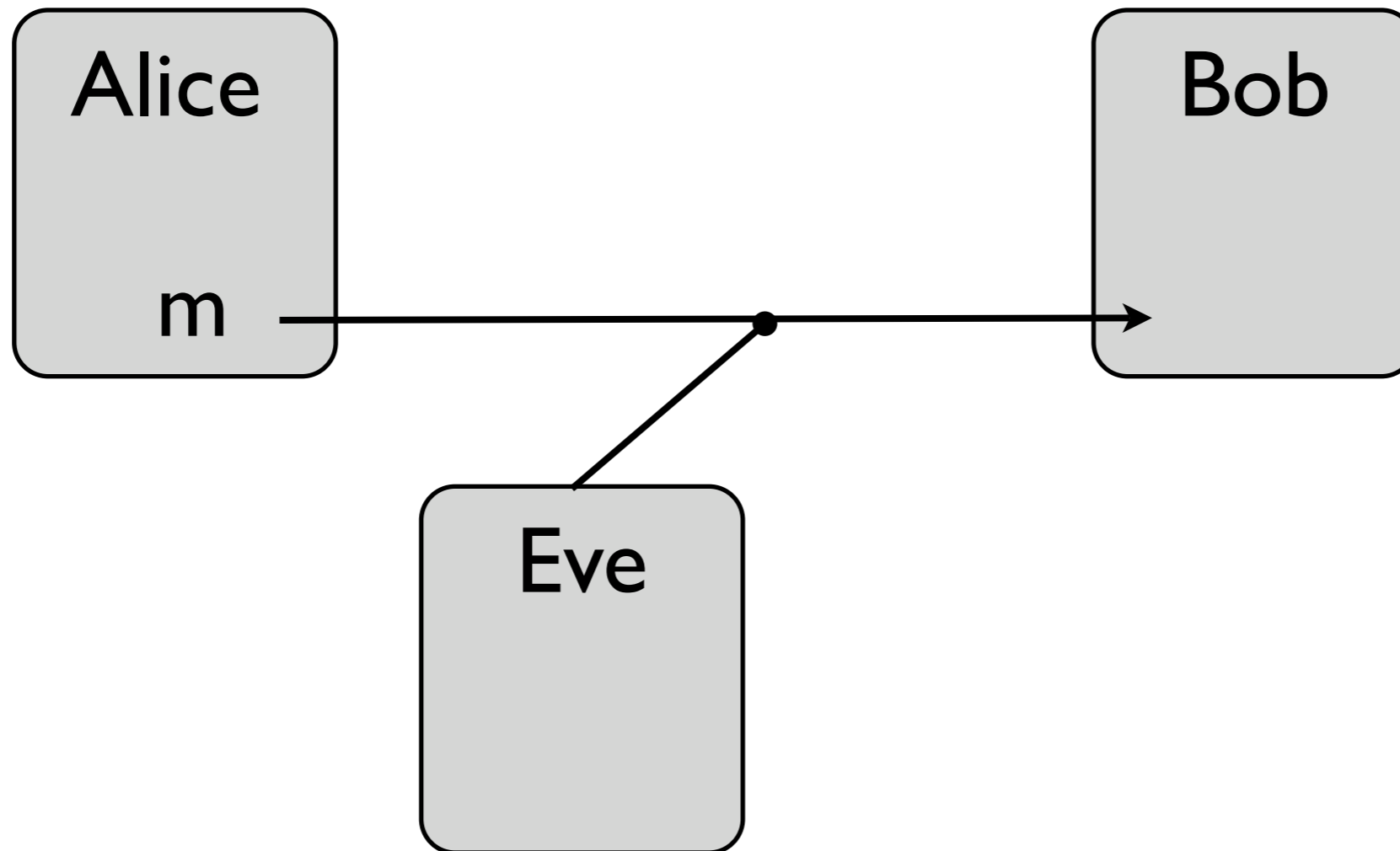
randomness

Luca Trevisan
University of California, Berkeley

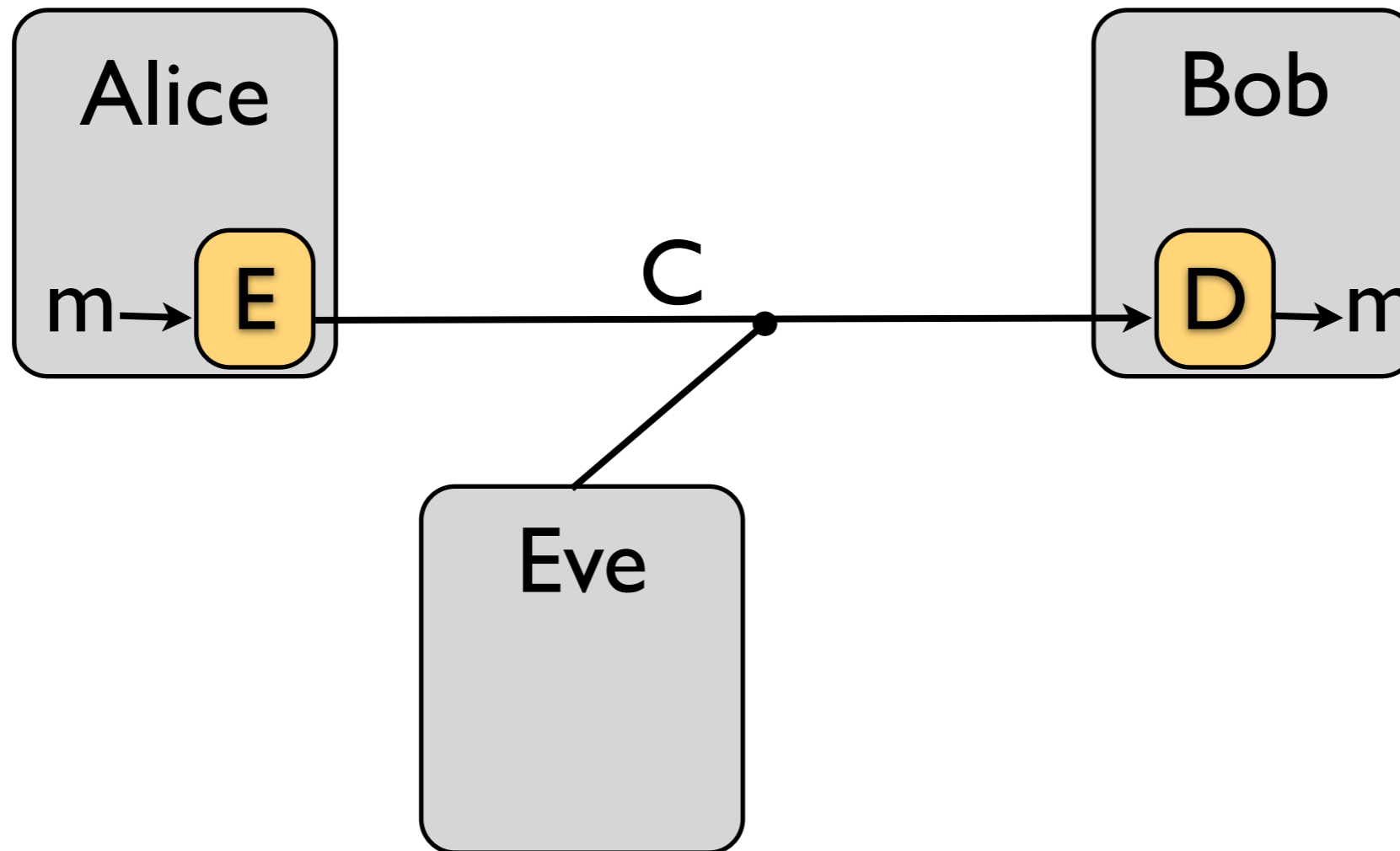
plan

- today: algorithms
 - applications that require randomness: cryptography, property testing, streaming algorithms
 - randomness extraction
 - randomized algorithms
- tomorrow: complexity
 - derandomization
 - average-case complexity

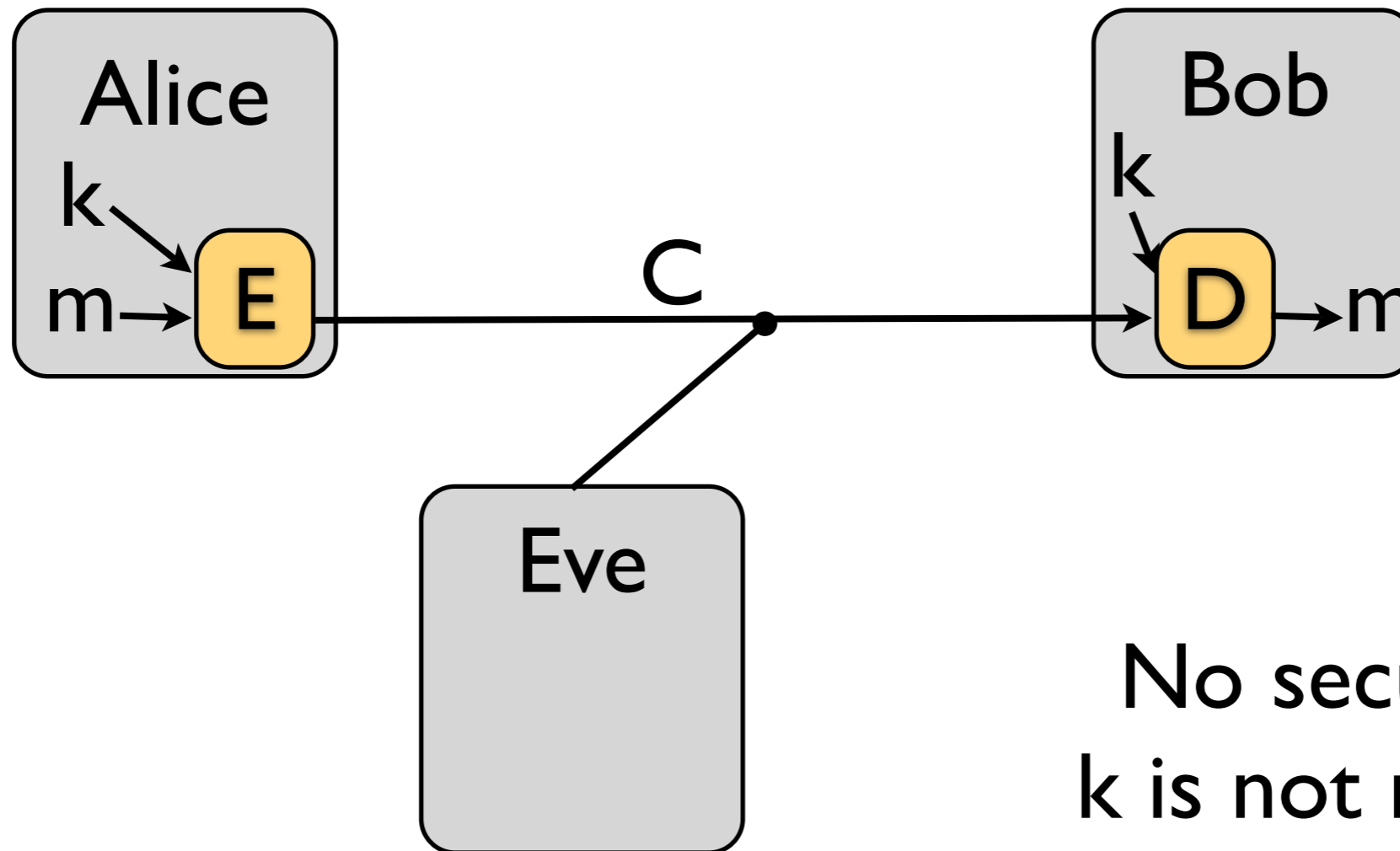
encryption



encryption

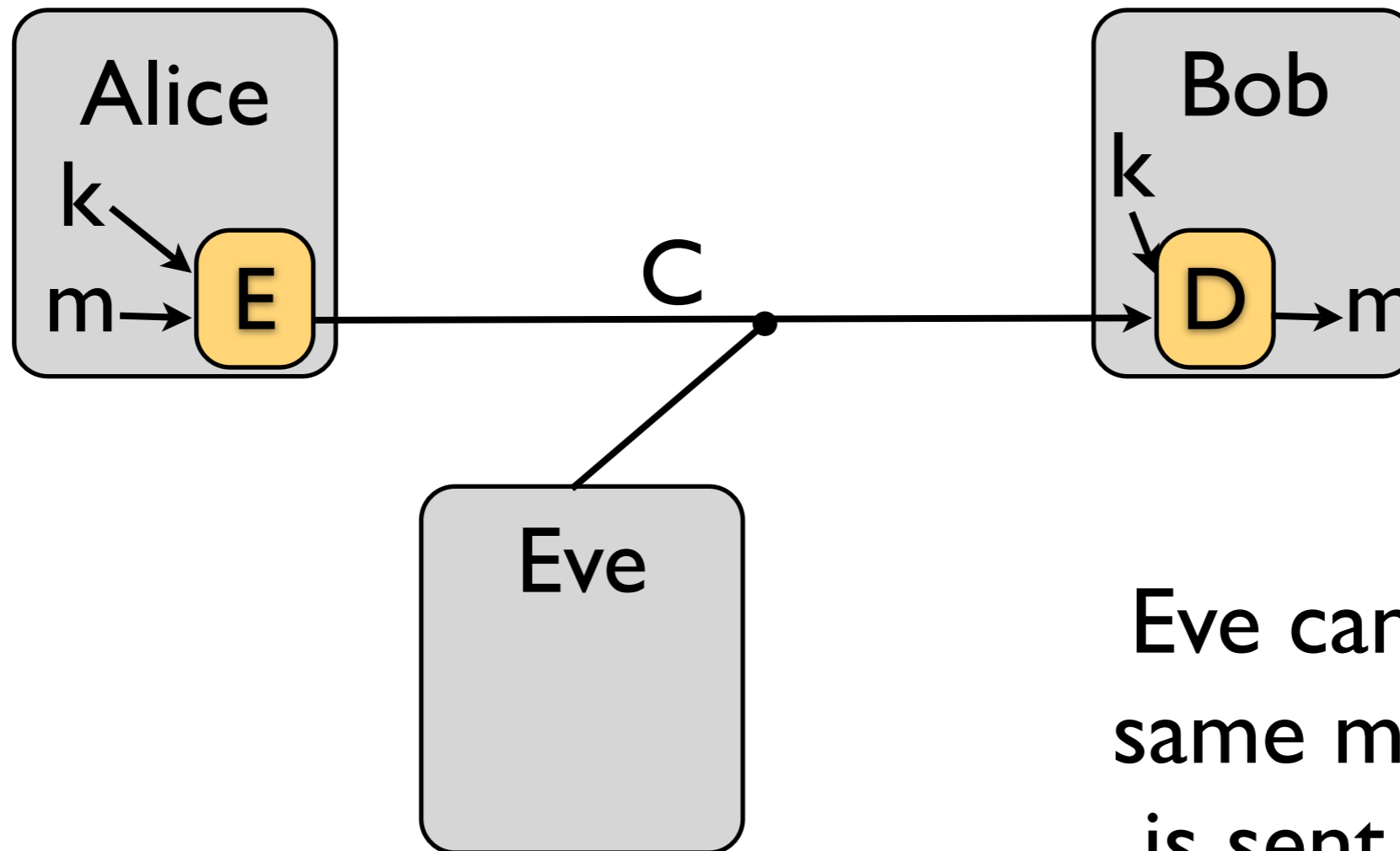


encryption



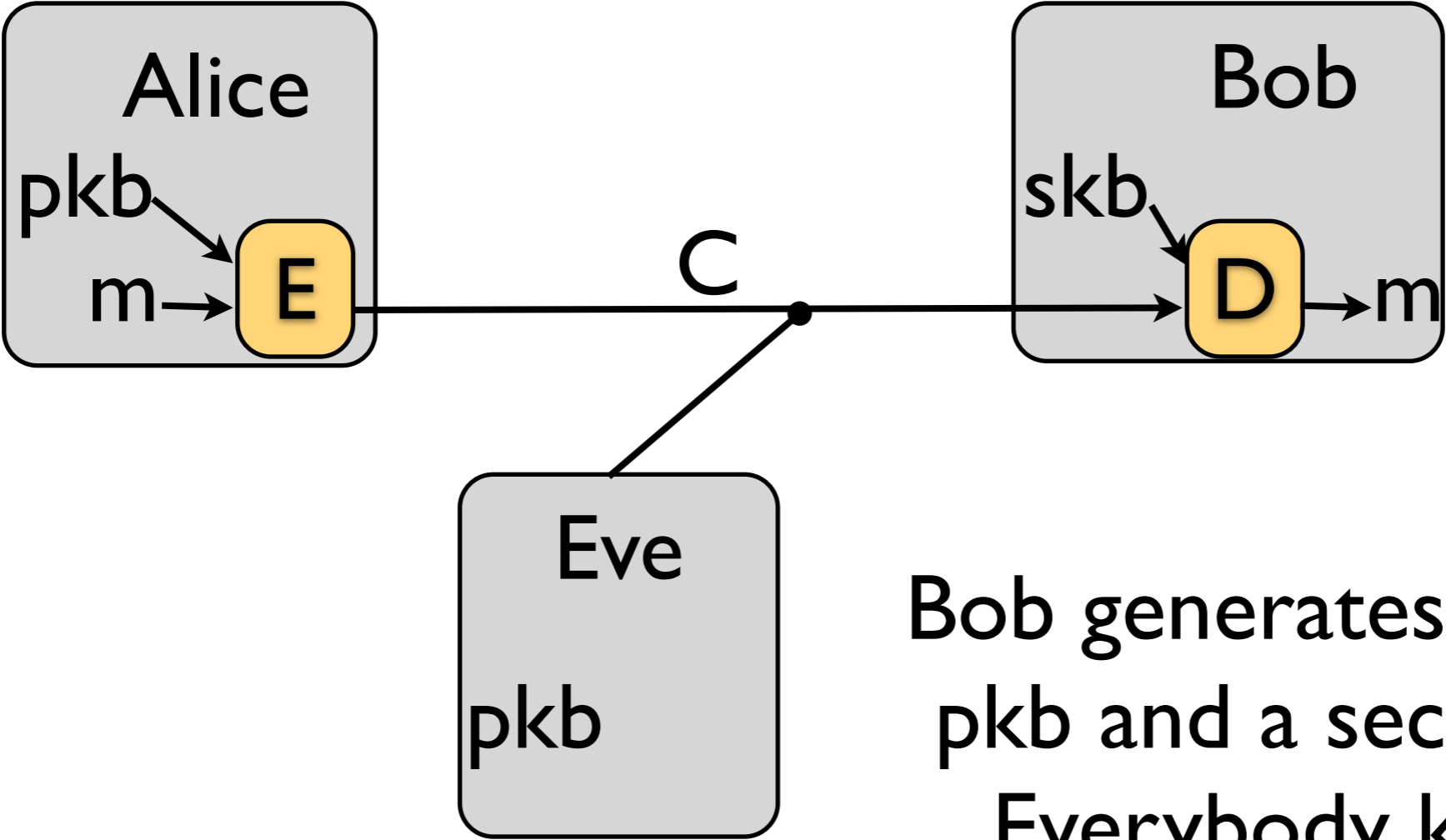
No security if
k is not random

encryption



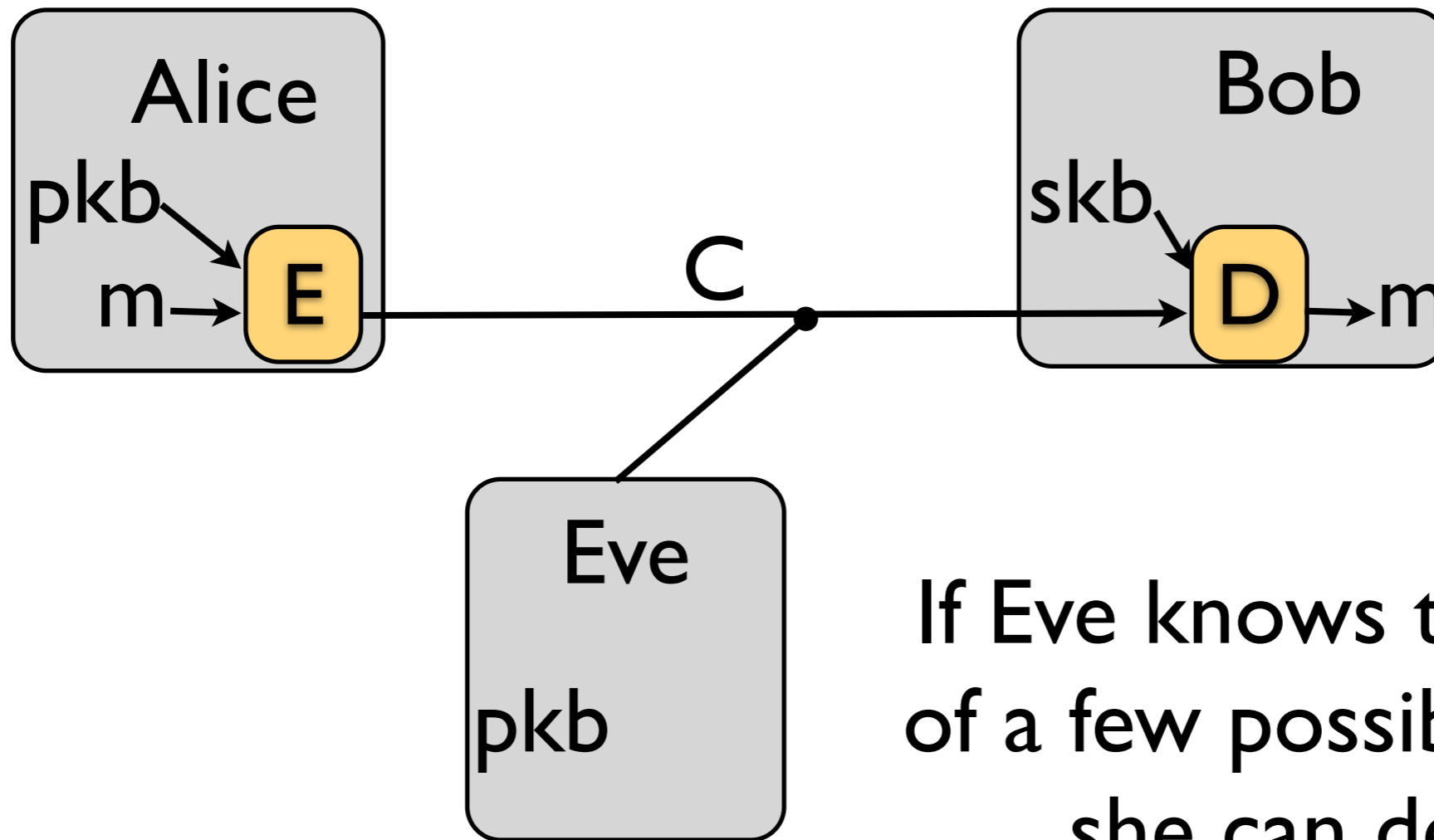
Eve can tell if
same message
is sent twice

public key encryption



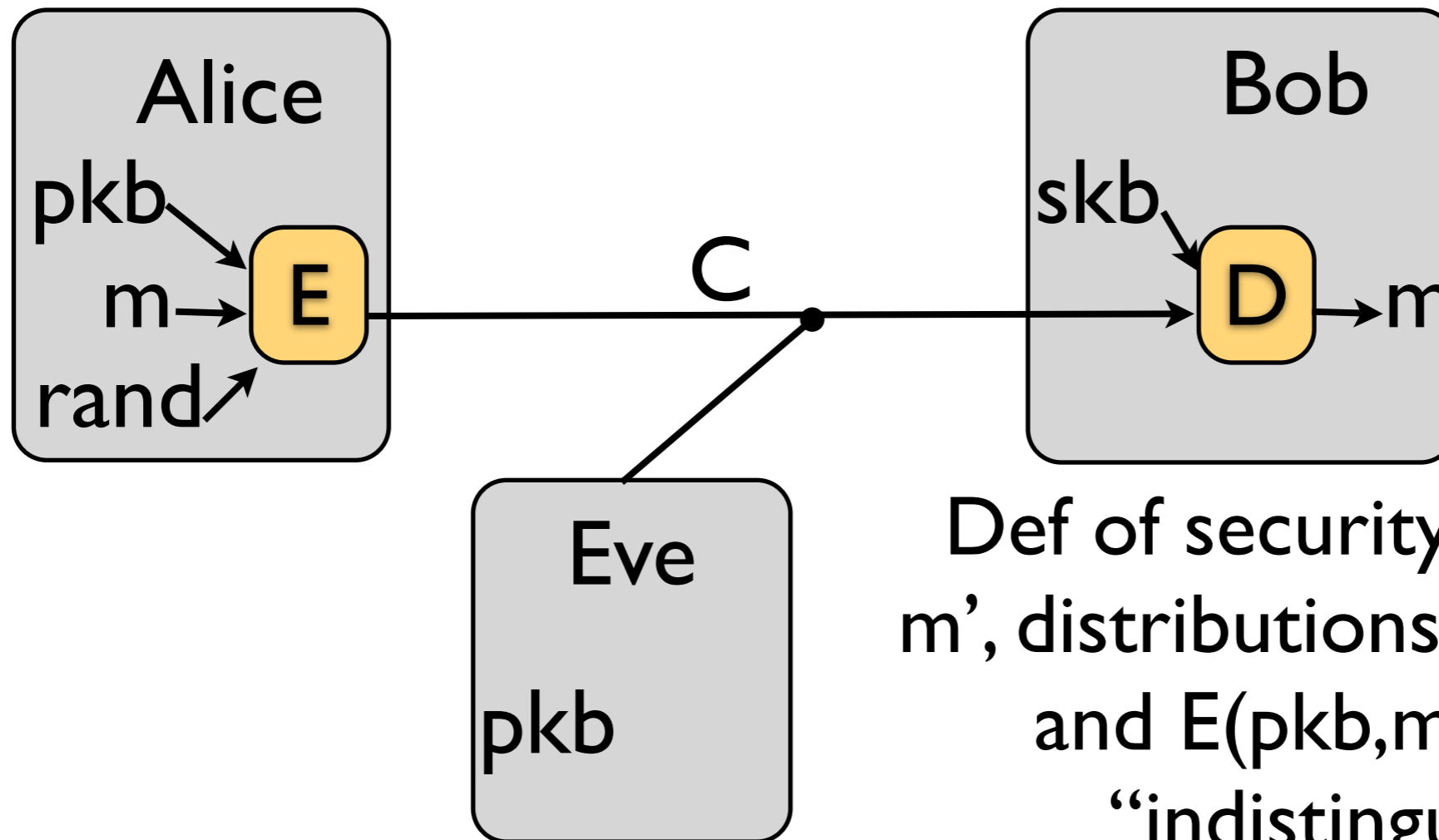
Bob generates a public key pkb and a secret key skb. Everybody knows pkb

public key encryption



If Eve knows that m is one of a few possible messages, she can decrypt C

secure public key encryption



Def of security: for every m , m' , distributions $E(pk_b, m, rand)$ and $E(pk_b, m', rand)$ are “indistinguishable”
[Goldwasser-Micali]

sampling

How many Americans approve of Obama?

Approaches:

- **Deterministic.** if I ask my friends and friends of friends, unlikely to have representative answers
- **Exact.** without asking every American, impossible to have exact count
- **Randomized and approximate.** with probability $1-\delta$, possible to get count with error ϵ , after asking $O(\epsilon^{-2} \log \delta^{-1})$ random people

sampling

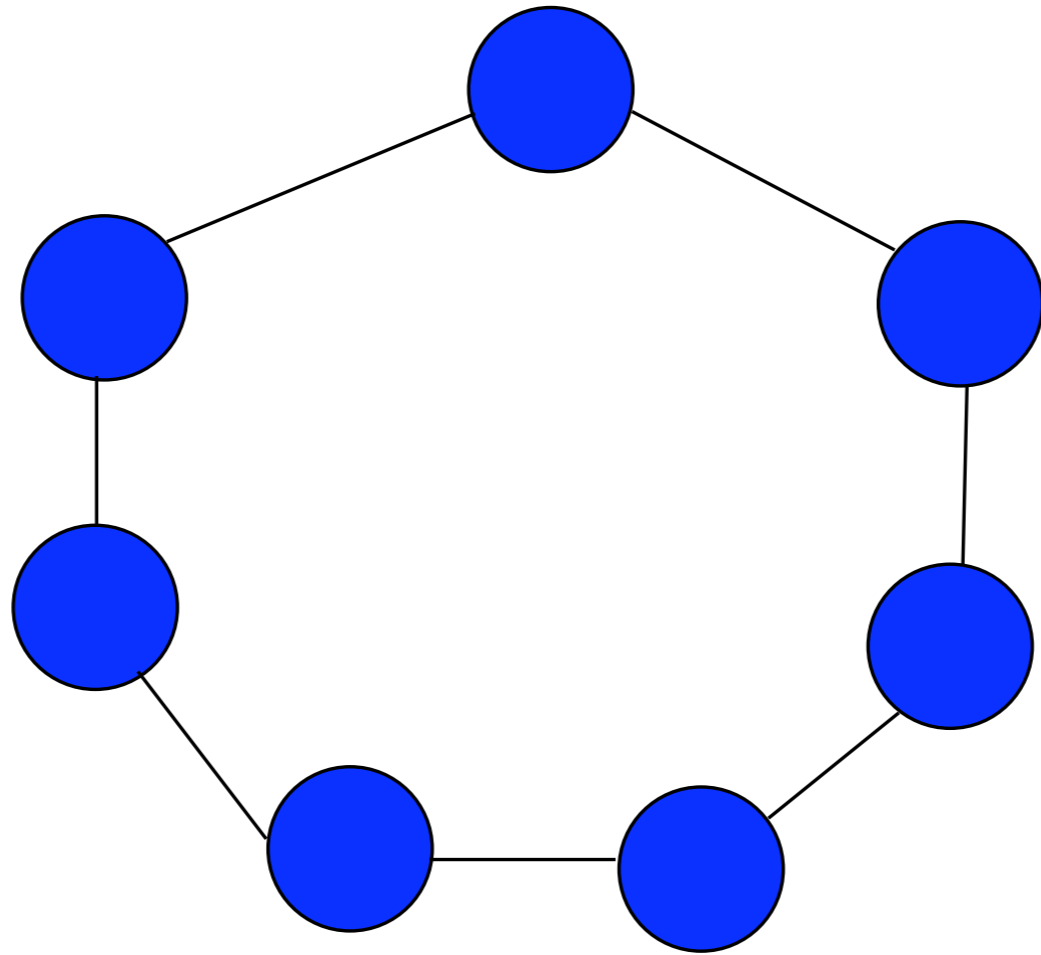
How many Americans approve of Obama?

- **Randomized and approximate.** with probability $1-\delta$, possible to get count with error ϵ , after asking $O(\epsilon^{-2} \log \delta^{-1})$ random people
- Complexity dependent only on desired quality of approximation and success probability
- Independent of # of Americans

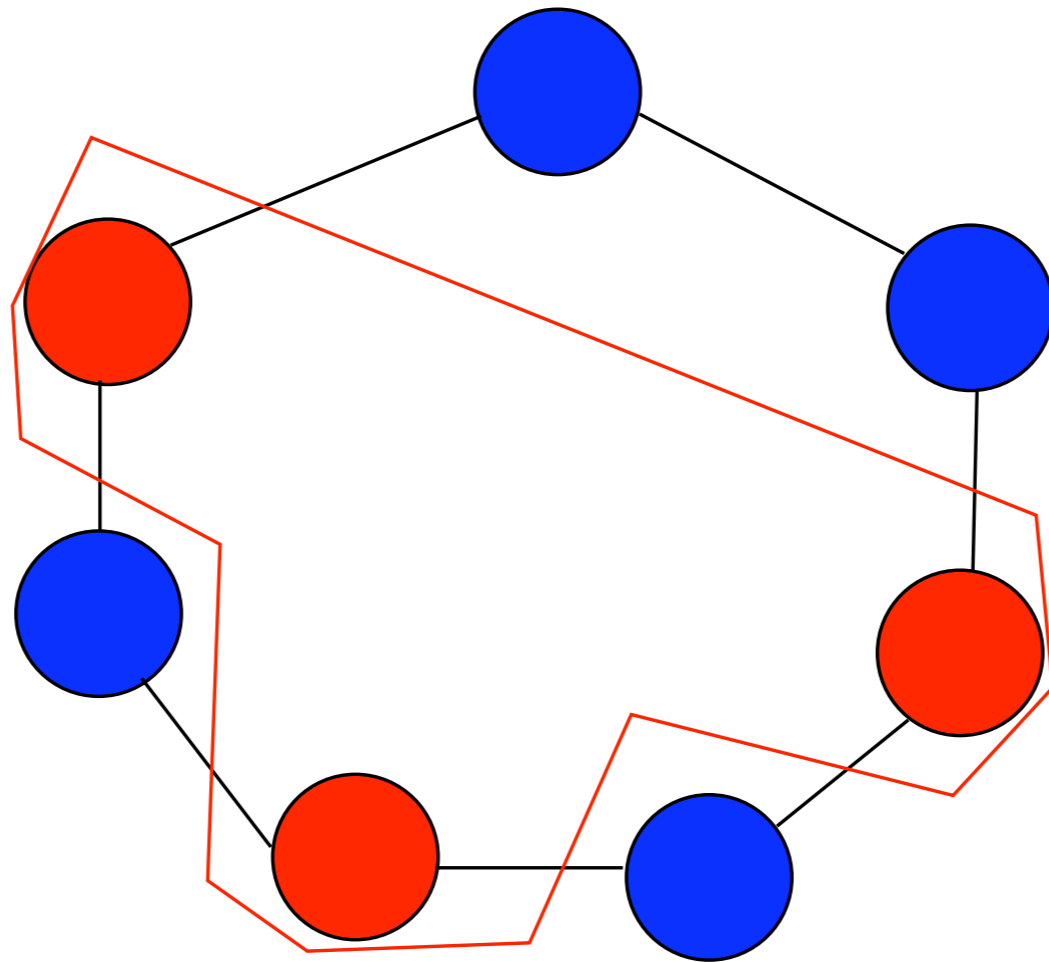
property testing & sublinear time algs

- Max Cut: given undirected graph $G=(V,E)$, partition V into (A,B) , so that max # of edges have one endpoint in A and one endpoint in B

property testing & sublinear time algs



property testing & sublinear time algs



property testing & sublinear time algs

- Max Cut: given undirected graph $G=(V,E)$, partition V into (A,B) , so that max # fraction of edges have one endpoint in A and one endpoint in B
- [Goldreich-Goldwasser-Ron]
 - pick random set S of $O(\varepsilon^{-O(1)})$ vertices
 - find Max Cut in graph induced by S
 - with high probability Max Cut of G is same $\pm\varepsilon$

property testing & sublinear time algs

[GGR, Alon-Krivelevich] Given $G=(V,E)$ undirected:

- Pick set S of $\tilde{O}(\epsilon^{-2})$ vertices
- Accept iff graph induced by S is bipartite
- **G bipartite** \Rightarrow algorithm accepts with prob 1
- **G ϵ -far from bipartite** \Rightarrow algorithm rejects with prob 99%
 - ϵ -far: need to remove $> \epsilon n^2$ edges to make graph bipartite

property testing

- General definition [Rubinfeld-Sudan,GGR]
- algorithm T is a tester for property P with accuracy ϵ if
- Given input I with property P , T accepts with prob $> 99\%$
- Given I ϵ -far from property P , T accepts with prob $< 1\%$
- Definition of “ ϵ -far” depends on input representation
e.g. having to add/remove $> \epsilon|V|^2$ edges in graph represented as adjacency matrix

examples

- Is a given graph k -colorable? (fixed k)
adjacency matrix representation
complexity $\exp(\varepsilon^{-O(1)})$ [GGR]
- Is a given boolean function GF(2)-linear?
truth-table representation
complexity $O(\varepsilon^{-O(1)})$ [Blum-Luby-Rubinfeld]
(extended to other fields, and to low-degree polys)
- ...
(is a given list sorted, is a given function a “junta”, are two given distributions statistically close, ...)

testing bipartiteness in sparse graphs

- Given $G=(V,E)$, max degree d , represented via adjacency list, test if it is bipartite or ϵ -far from bipartite
- ϵ -far: need to remove $> \epsilon d|V|$ edges to make graph bipartite

testing bipartiteness in sparse graphs

- Given $G=(V,E)$, max degree d , represented via adjacency list, test if it is bipartite or ε -far from bipartite
 - ε -far: need to remove $> \varepsilon d|V|$ edges to make graph bipartite
 - Lower bound: need to look at $\Omega(\sqrt{|V|})$ vertices
 - Algorithm: complexity $\tilde{O}(\varepsilon^{-O(1)} \sqrt{|V|})$ is sufficient
- [Goldreich-Ron]

streaming algorithms

Model:

- Input is a (HUGE) sequence of data items (e.g. numbers)
- Algorithm makes only one sequential pass through data
- Complexity: memory used by algorithm; time per item (special focus on memory)
- Motivation: data generated on-the-fly in scientific applications, e-commerce, network administration, sensor networks, ...

streaming algorithms

- Deterministic, exact algorithms: finite state automata.
Easy to prove strong lower bounds for almost any task
- Randomized, approximate: very interesting (and useful) algorithms and theory
- heavily studied not just by theoreticians but also by DB and networking researchers

counting distinct elements

- Given sequence of items
- Compute the number of distinct items
 - e.g. number of unique readers of a web page given log of IP addresses from which the page was accessed
- Use bounded memory in the streaming model

counting distinct elements

- Deterministic algorithms: need linear memory even for an approximation
- Exact algorithms: need linear memory even for randomized algorithms
- Randomized and approximate?

counting distinct elements

[Alon Matias Szegedy] Given sequence x_1, \dots, x_n

- Pick random function $h: U \rightarrow [0, 1]$
- Find minimum m of $h(x_1), \dots, h(x_n)$
- Output $1/m$

(It's enough to pick h randomly from a family of “pairwise independent hash function”, range can be discretized, h needs only $(\log n)$ bits of storage, and so does n)

counting distinct elements

- If x_1, \dots, x_n contain k distinct elements, then
- Then $h(x_1), \dots, h(x_n)$ is a sequence of k random elements of $[0, 1]$, possibly with some elements repeated
- On average, min of k random numbers in $[0, 1]$ is $1/k$
- It is between $2/k$ and $1/2k$ with “high” probability
- Algorithm has “high” probability of being a 2-approximation

how to generate random bits

Typical approach:

1. observation of an unpredictable physical phenomenon
2. mapping of measurements to unbiased random bits

Issues:

- which phenomena have enough entropy?
- how to do the mapping?

random sources

with no extra hardware

- current time, current process ID
- read-write hard disk, measure latency
- user's time intervals between pressing keys; user's mouse movements

with extra hardware

- thermal noise

from high-entropy source to random bits

How to convert a high-entropy source into random bits?

- [Von Neumann] if source is a sequence of **independent** biased bits with same (unknown) bias:
 - parse stream of bits into pairs
 - discard 00 and 11
 - map 01 to 0; 10 to 1
 - e.g. 10010001011010

from high-entropy source to random bits

How to convert a high-entropy source into random bits?

- [Von Neumann] if source is a sequence of **independent** biased bits with same (unknown) bias:
 - parse stream of bits into pairs
 - discard 00 and 11
 - map 01 to 0; 10 to 1
 - e.g. 10 01 00 01 01 10 10

from high-entropy source to random bits

How to convert a high-entropy source into random bits?

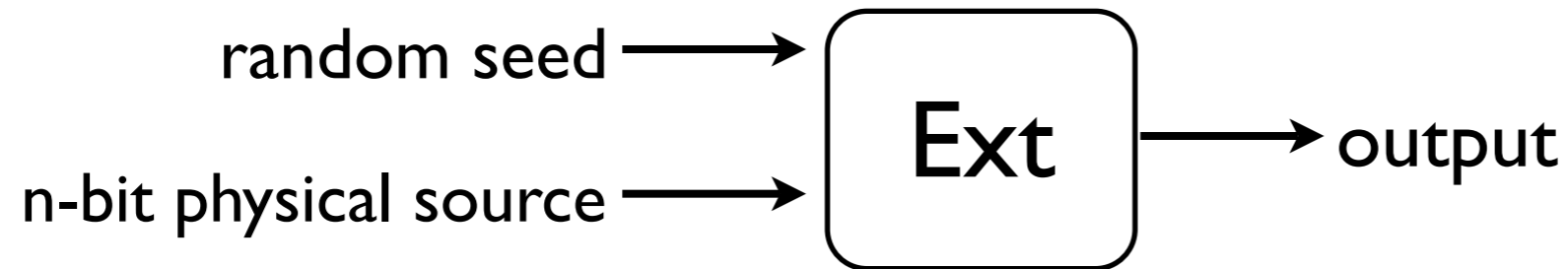
- [Von Neumann] if source is a sequence of **independent** biased bits with same (unknown) bias:
 - parse stream of bits into pairs
 - discard 00 and 11
 - map 01 to **0**; 10 to **1**
 - e.g. 10 01 00 01 01 10 10 → **100011**

from high-entropy source to random bits

How to convert a high-entropy source into random bits?

- Problem studied very intensely in the 1980s and 1990s
- For models allowing **dependencies** in the stream of bits, impossibility results
- Zuckerman introduces a theory of randomness extraction with auxiliary randomness for arbitrary random sources

(n,k) seeded randomness extractor



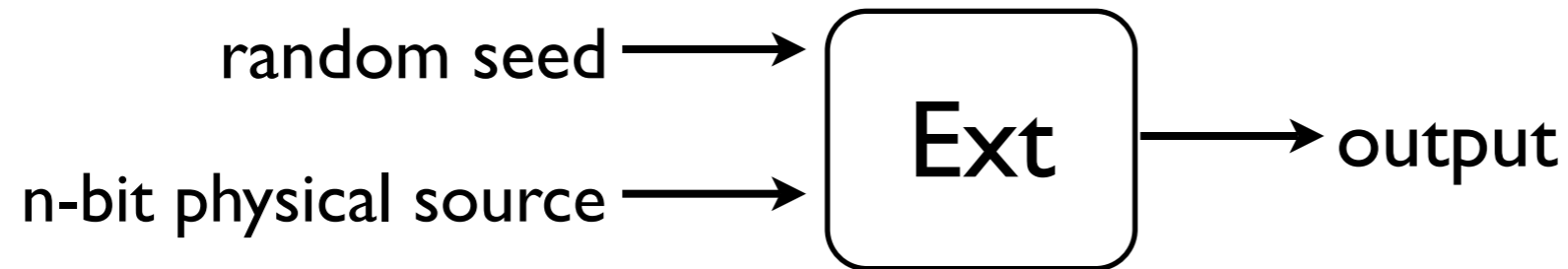
random seed: perfect random bits

random source: distribution of observations with arbitrary biases;
only assumption: entropy $> k$

output: perfect random bits

Meaningful if output is much longer than seed

(n,k) seeded randomness extractor



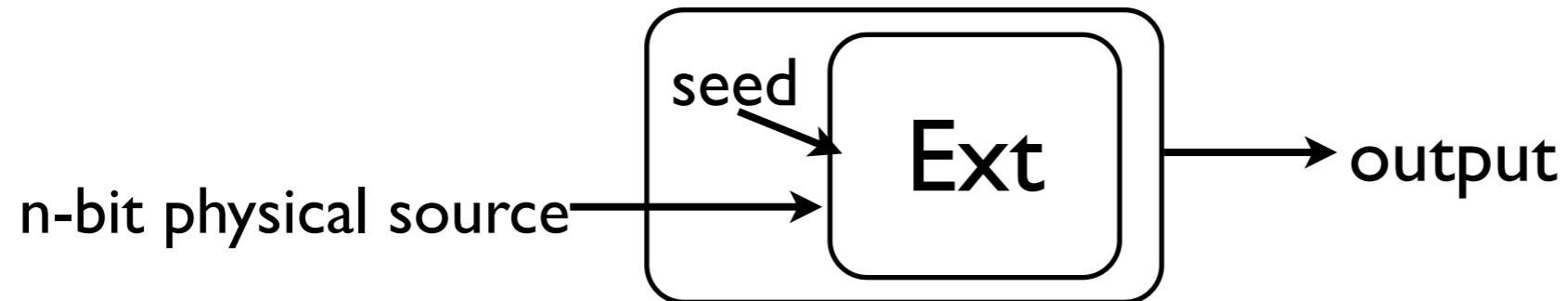
Current constructions:

- Seed is $O(\log n)$ bits — optimal
- Output is about k bits — optimal
- Work of several people over 20+ years including: Guruswami
Nisan Reingold Safra Shaltiel Ta-Shma Trevisan Umans Vadhan
Zuckerman Wigderson

seeded randomness extractors

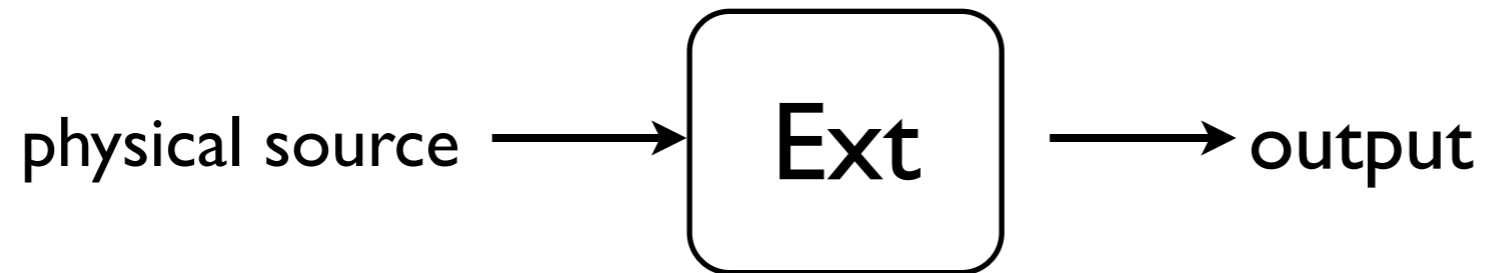
- Major area in complexity theory and combinatorics
- Relations to:
 - Expander graphs
 - Error-correcting codes
 - Hash functions
 - Pseudorandom Generators
 - ...
 - Randomness extraction?

dealing with the extra seed



- Hard-wire it as part of the specification of the extraction function [Barack et al.]

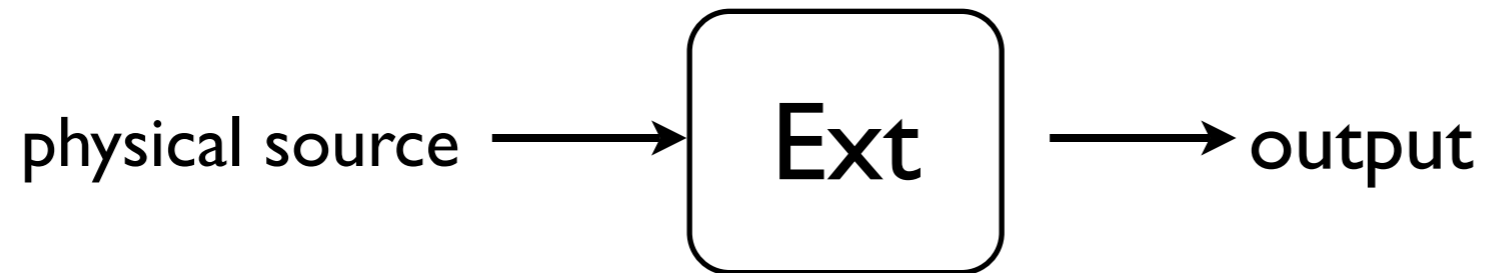
seedless extractors



- Impossible:
If physical source has entropy $> k$, then output is random
- Desired:
If physical source has entropy $> k$ and property P , then output is random

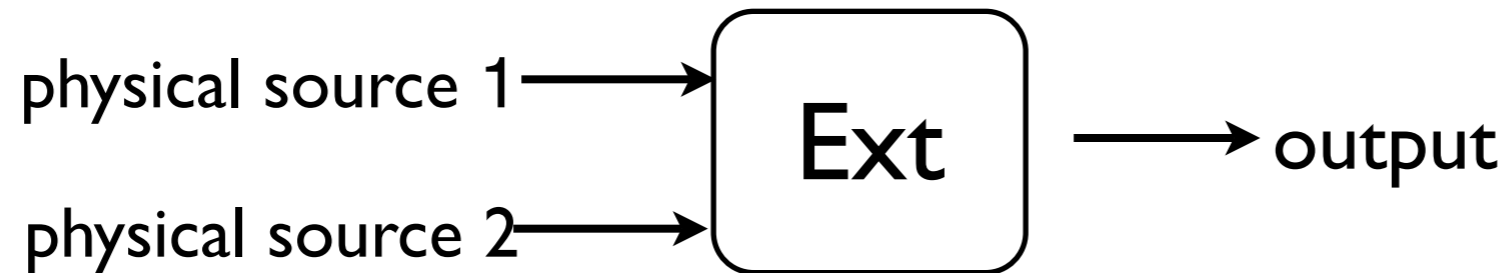
Provided property P is natural and presumably true of sources occurring in practice

seedless extractors



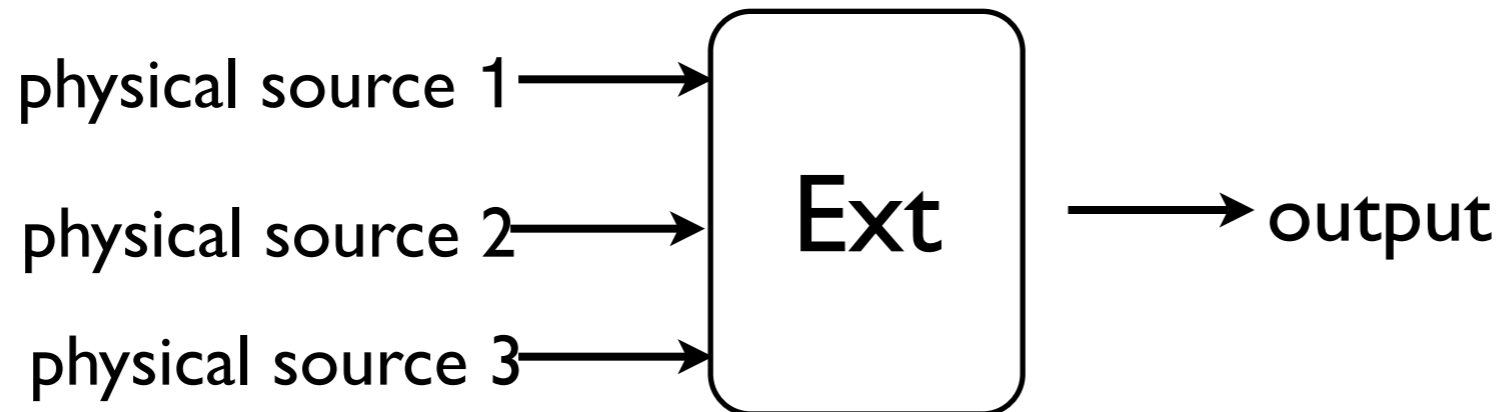
- [Chor Goldreich] suppose physical source is a pair of n -bit strings, whose distributions are arbitrary but **independent**
- They show how to extract randomness provided total entropy is $> n$

seedless extractors



- [Chor Goldreich '88] To produce one bit:
- Let physical sources be $x_1 \dots x_n$ and $y_1 \dots y_n$
- Output $x_1 y_1 + x_2 y_2 + \dots + x_n y_n \pmod{2}$
- No further progress until [Bourgain '05]

seedless extractors



- Major recent progress on extracting from 3 or more independent sources [Barak Impagliazzo Wigderson Rao ...]
- Intel Random Chip has two generators of thermal noise whose measurements are combined

randomness in algorithms

- Given N , is it prime?
- Randomized polynomial time (and efficient) algorithms in the 1970s (Miller, Rabin)
- Later, polynomial time (not efficient) algorithm [Agrawal Kayal Saxena 2002]

the Miller-Rabin algorithm

- Given N , composite,
- pick a at random in $\{ 2, \dots, N-1 \}$
- with probability at least $1/2$, at least one of the following holds:
 - $a^{N-1} \neq 1 \pmod{N}$ impossible if N is prime
 - $\gcd(a, N) \neq 1$ impossible if N is prime
 - for some k , and for $m := (N-1)/2^k$:
 - $a^m \neq \pm 1 \pmod{N}$
 - $a^{2m} = 1 \pmod{N}$impossible if N is prime

the AKS algorithm

- [Agrawal Biswas] Different randomized algorithm:
 - Given N , construct polynomial p_N of degree d such that
 - If N is prime, p_N is always zero
 - If N is composite, p_N is not always zero
 - Evaluate p_N at a random element from a set of size $2d$
($\log d$ is polynomial in $\log N$)
- [Agrawal Kayal Saxena] If p_N is not always zero, then it's non-zero somewhere in a fixed set of size polynomial in $\log N$

undirected connectivity

- Given $G=(V,E)$ undirected and vertices s,t , is there a path between s and t ?
- Linear time, linear space algorithms: BFS, DFS
- **Constant** space?
(that is, $O(\log n)$ bits?)

random walk algorithm

- start at s , then repeatedly move to a random neighbor
 - if t is reached then accept
 - if t is not reached within a certain number of steps, reject
- [Aleliunas Karp Lipton Lovasz Rackoff 1979] If s and t are connected, t is reached with high probability in $O(|V|^3)$ steps
- Memory: counter, pointers to current and next vertex

undirected connectivity

- [AKLLR 1979]
 - $O(\log |V|)$ bits of memory, $O(|V|^3)$ time
 - pick a walk at random, accept if t is reached from s
- [Reingold 2005]
 - $O(\log |V|)$ bits of memory, polynomial in $|V|$ time
 - try each walk from a certain carefully constructed set
accept if t is reached from s in one of them

conclusions (for now)

- In cryptography, property testing and streaming algorithms, randomness is provably necessary
- There are rigorously motivated ways to create the required random bits from physical sources
- In two early major applications of randomness to algorithm design, randomness has been shown to give, at best, a polynomial speed-up
- Does randomness ever give more than a polynomial speed-up in algorithm design?

day 2: is randomness necessary in algorithm design?

Pseudorandomness

Average-case Complexity

recall

- Given integer N , is it prime?
 - 1970s, polynomial (in $\log N$) time randomized algorithms
 - 2002, polynomial time deterministic algorithm
- Given undirected $G=(V,E)$, are vertices s,t connected?
 - 1979, polynomial time, $O(\log n)$ -bit memory randomized alg
 - 2005, polynomial time, $O(\log n)$ -bit memory deterministic alg

approximate counting

- Given undirected $G=(V,E)$, how many perfect matchings, if any, does it have?
- Given DNF formula f , how many satisfying assignments does it have
- Given $G=(V,E)$, how many valid 3-colorings, if any, does it have
- All #P-complete (give poly-time alg for one problem, we get poly-time algs for all problems of counting witness for any NP problem)

approximate counting

1. Given undirected $G=(V,E)$, how many perfect matchings, if any, does it have?
2. Given DNF formula f , how many satisfying assignments does it have
3. Given $G=(V,E)$, how many valid 3-colorings, if any, does it have

Approximating (3) is NP-hard (distinguish 0 from > 0)

Approximating (1) and (2)?

approximate counting

Idea of Jerrum and Sinclair:

- suppose we can approximately *sample* from the uniform distribution of perfect matchings of G
- then we can approximately *count* the number of perfect matchings

Approach: pick an edge (u,v) , approximate fraction of matchings that do and do not include (u,v) , recursively approximate number of matchings in case of higher probability

approximate counting

Idea of Jerrum and Sinclair:

- suppose we have a counting problem in $\#P$ that is “downward self-reducible” and such that we can approximately sample from the set of witnesses for a given instance
- then we can approximately solve the $\#P$ problem

When this approach works, it *always* yield randomized alg

how to sample

- Suppose $\mathbf{G}=(\mathbf{V},\mathbf{E})$ is a regular connected graph
- Take a random walk for t steps in \mathbf{G} starting from an arbitrary vertex
- If t is large enough, the distribution of final vertex reached in the walk is close to uniform over \mathbf{V}

how to sample

- Given $G=(V,E)$ we want to sample from the uniform distribution over its set of perfect matchings
- Define graph $\mathbf{G}=(\mathbf{V},\mathbf{E})$ where \mathbf{V} is set of perfect matchings and \mathbf{E} is a *regular* set of edges $(M1,M2)$ such that $M2$ is derived from $M1$ via “local changes”
- Start from a fixed matching and do a random walk for t steps
- If $t=|V|^{O(1)}$ is sufficient to reach uniform matching, we are done

bounding the mixing time

- The number of perfect matchings of G (= the number of vertices of \mathbf{G}) can be exponential in the size of G
- For the algorithm to be efficient, a random walk in \mathbf{G} must reach a nearly uniform vertex in a number of steps polylogarithmic in the size of \mathbf{G}
- Various techniques to prove such bounds
- Jerrum-Sinclair-Vigoda recently proved it in this case

randomized → deterministic

- Given integer N , is it prime?
 - 1970s, polynomial (in $\log N$) time randomized algorithms
 - 2002, polynomial time deterministic algorithm
- Given undirected $G=(V,E)$, are vertices s,t connected?
 - 1979, polynomial time, $O(\log n)$ -bit memory randomized alg
 - 2005, polynomial time, $O(\log n)$ -bit memory deterministic alg
- Given undirected $G=(V,E)$, approx how many perfect matchings?
 - 1980s-2000s polynomial time randomized alg
 - deterministic?

derandomization

- Suppose there is a randomized polynomial time algorithm A that solves computational problem X
- Is there always deterministic polynomial time A' that solves X ?
- Maybe sub-exponential A' ?

derandomization

- If $P=NP$: YES
- If $P \neq NP$: also YES!

derandomization

- If $P=NP$: YES
- If “ $P \neq NP$ ”: also YES!

derandomization

If $P=NP$ then

- In polynomial time can find strings with a given property if they exist
(Gives derandomization of “one-sided error” algorithms)
- Harder but known since 1980s: can also approximately count the number of strings with a given property
(Derandomizes all randomized algorithms)

derandomization

- [Nisan-Wigderson, Babai-Fortnow-NW] Suppose there is a problem in EXP (e.g. a problem in NP) that cannot be solved by *polynomial size circuits*
- Then every problem solvable in randomized polynomial time can also be solved in deterministic sub-exponential time $\exp(n^{o(1)})$

derandomization

- [Nisan-Wigderson, Impagliazzo-Wigderson] Suppose there is a problem in E (e.g. SAT) that cannot be solved by *circuits of size $\exp(o(n))$*
- Then every problem solvable in randomized polynomial time can also be solved in deterministic polynomial time

derandomization

- [Nisan-Wigderson, Impagliazzo-Wigderson] Suppose there is a problem in E (e.g. SAT) that cannot be solved by *circuits of size $s(n)$*
- Then every problem solvable in randomized polynomial time can also be solved in deterministic time $\exp(s^{(-1)}(n))$

How to prove the Nisan-Impagliazzo-Wigderson Theorem

pseudorandom generator

Def [Blum-Micali, Goldwasser-Micali, Yao 1982]:

- length-increasing deterministic procedure;
- if input is uniform random bits, output “looks like” uniform random bits

011011001 → G → 1101000011011001

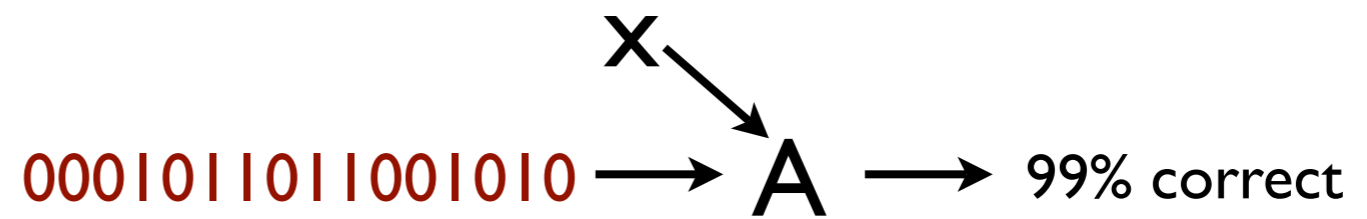
pseudorandom generator

- Generator G is ϵ -pseudorandom against circuits of size s if for every test T computable in size s

0001011011001010 \longrightarrow T \longrightarrow outputs 1 prob p

011011001 \longrightarrow G \longrightarrow 1101000011011001 \longrightarrow T \longrightarrow outputs 1 prob $p \pm \epsilon$

p.r.g. to reduce randomness

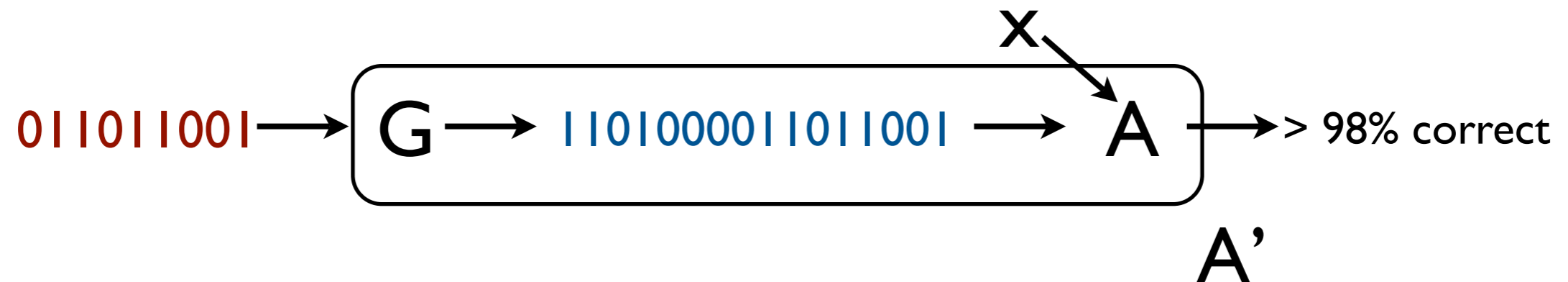
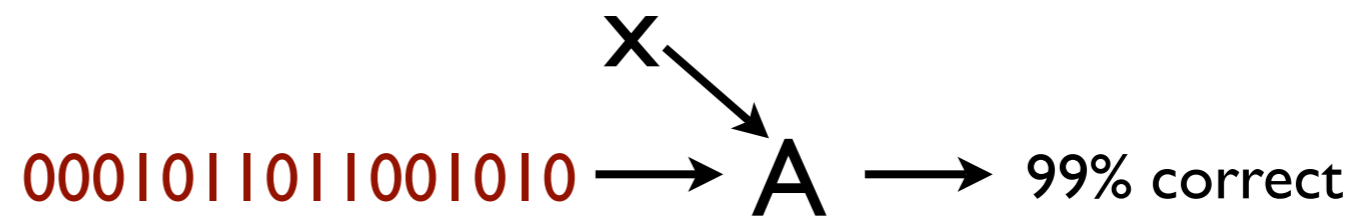


p.r.g. to reduce randomness

0001011011001010 → ^x A → 99% correct

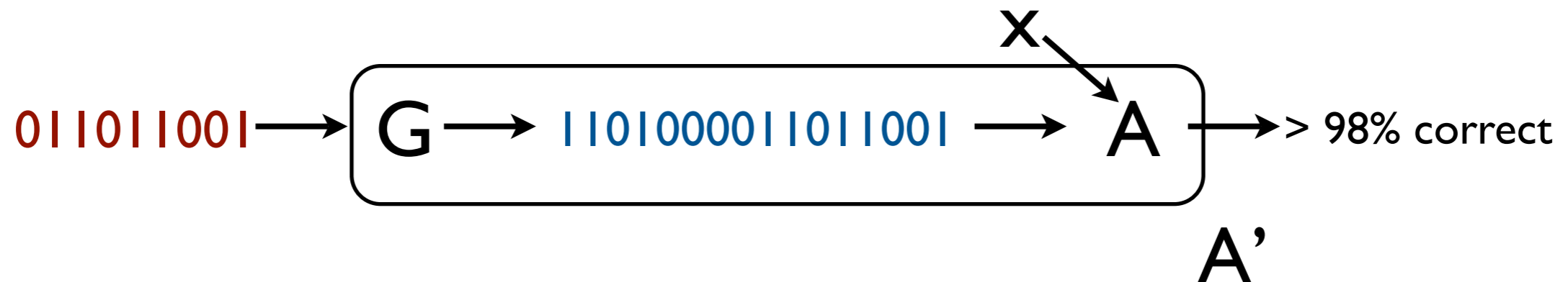
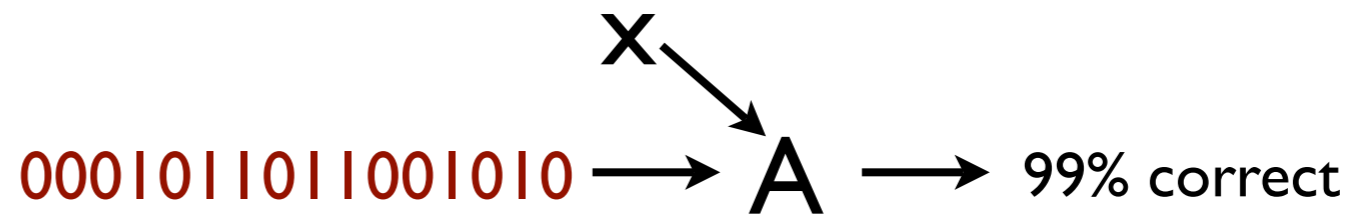
011011001 → G → 1101000011011001 → ^x A → > 98% correct

p.r.g. to reduce randomness



p.r.g. to eliminate randomness

Suppose A' runs in $\text{poly}(n)$ time and uses $O(\log n)$ random bits. Then bit can be made deterministic by enumerating random bit strings



also for numerical problems

- Jerrum-Sinclear-Vigoda algorithm approximates # of perfect matchings $\pm \delta$ with prob $> 99\%$
- Replacing random input with p.r.g. is such that approximation $\pm \delta$ with prob $> 98\%$
- Enumerating all random inputs and outputting *median* gives deterministic approximation $\pm \delta$ in time exponential in the seed length of the generator

Nisan-Impagliazzo-Wigderson restated

- Suppose there is a (family of) boolean function(s)
 $f_k: \{0,1\}^k \rightarrow \{0,1\}$
computable by a uniform algorithm in time $2^{O(k)}$
but not computable by circuits of size $2^{o(k)}$
- Then $\forall c,d$, there is a $\text{poly}(n)$ -time computable p.r.g. G
that maps $O(\log n)$ bits into n^c bits
and is 1% pseudorandom against circuits of size n^d

Nisan-Impagliazzo-Wigderson restated

- Suppose there is a (family of) boolean function(s)
 $f_k: \{0,1\}^k \rightarrow \{0,1\}$
computable by a uniform algorithm in time $2^{O(k)}$
but not computable by circuits of size $s(k)$
- Then there is a $2^{O(k)}$ -time computable p.r.g. G
that maps $O(k)$ bits into $s(k)^{\Omega(1)}$ bits
and is 1% pseudorandom against circuits of size $s(k)^{\Omega(1)}$

Nisan-Impagliazzo-Wigderson restated

- Suppose there is a (family of) boolean function(s)
 $f_k: \{0,1\}^k \rightarrow \{0,1\}$
computable by a uniform algorithm in time $2^{O(k)}$
but not computable by circuits of size $2^{k/50}$
- Then there is a $2^{O(k)}$ -time computable p.r.g. G
that maps $2,500 \cdot k$ bits into $2^{k/2,500}$ bits
and is 1% p.r. against ckts of size $2^{k/2,500}$.
Take $k = 10,000 \log n$

average-case complexity \rightarrow p.r.g.

[Nisan-Wigderson 1988]

- Suppose there is $f_k : \{0,1\}^k \rightarrow \{0,1\}$ computable in time $2^{O(k)}$ and $s(k) = 2^{\Omega(k)}$ such that for all A of size $< s(k)$
 $\Pr [A(x) = f(x)] < 1/2 + 1/s(k)$
(NW assumption)
- Then there is $\text{poly}(n)$ time computable generator with $O(\log n)$ seed p.r. against circuits of size $\text{poly}(n)$

worst-case complexity \rightarrow average-case

[... Impagliazzo-Wigderson 1997]

- Suppose there is $f_k : \{0,1\}^k \rightarrow \{0,1\}$ computable in time $2^{O(k)}$ and not computable by circuits of size $s(k) = 2^{\Omega(k)}$
- Then there is $f'_k : \{0,1\}^k \rightarrow \{0,1\}$ computable in time $2^{O(k)}$ and $s'(k) = 2^{\Omega(k)}$ such that for all A of size $< s'(k)$
 $\Pr [A(x) = f'(x)] < 1/2 + 1/s'(k)$
(NW assumption)
- Then there is $\text{poly}(n)$ time computable generator with $O(\log n)$ seed p.r. against circuits of size $\text{poly}(n)$

average-case complexity \rightarrow p.r.g.

[Nisan-Wigderson 1988]

- Suppose there is $f_k : \{0,1\}^k \rightarrow \{0,1\}$ computable in time $2^{O(k)}$ and $s(k) = 2^{\Omega(k)}$ such that for all A of size $< s(k)$
 $\Pr [A(x) = f(x)] < 1/2 + 1/s(k)$
(NW assumption)
- Then
 $x \rightarrow x, f(x)$
is p.r.g. (but only maps k bits into $k+1$)

average-case complexity \rightarrow p.r.g.

[Nisan-Wigderson 1988]

- Suppose there is $f_k : \{0,1\}^k \rightarrow \{0,1\}$ computable in time $2^{O(k)}$ and $s(k) = 2^{\Omega(k)}$ such that for all A of size $< s(k)$
 $\Pr [A(x) = f(x)] < 1/2 + 1/s(k)$
(NW assumption)

- Then

$$x \rightarrow f(x|_{S_1}), f(x|_{S_2}), f(x|_{S_3}), \dots, f(x|_{S_n})$$

is p.r.g.

worst-case complexity → average-case

- Def: *permanent* of square matrix M :
$$\text{perm}(M) := \sum_{\pi} \prod_i M_{i,\pi(i)}$$
- e.g. is M adjacency matrix of bipartite graph, $\text{perm}(M)$ is the number of perfect matchings
- Lipton 1990
If *permanent* is hard to compute in worst case, then also hard on average

permanent

- $\text{perm}(M)$ is a degree- n polynomial in the n^2 variables $M_{i,j}$
- Abstract problem: $p(x_1, \dots, x_m)$ is a degree- d polynomial in m variables over finite field \mathbf{F} . We have algorithm A that makes at most a $1/10d$ fraction of errors in computing $p()$ over \mathbf{F}^m . Given \mathbf{x} , find $p(\mathbf{x})$

error-correction for polynomials

- Abstract problem: $p(x_1, \dots, x_m)$ is a degree- d polynomial in m variables over finite field \mathbf{F} . We have algorithm A that makes at most a $1/10d$ fraction of errors in computing $p()$ over \mathbf{F}^m . Given \mathbf{x} , find $p(\mathbf{x})$
- Sol: pick random line through \mathbf{x} :
$$l(t) := \mathbf{x} + t\mathbf{y}$$
for random \mathbf{y}

error-correction for polynomials

- pick random line through \mathbf{x} :
 $l(t) := \mathbf{x} + t\mathbf{y}$
for random \mathbf{y}
- the function $q(t) := p(l(t))$ is univariate poly of degree d
- given $d+1$ values of $q()$, can interpolate, find q ,
find $q(0)=p(\mathbf{x})$
- with high probability,
 $A(\mathbf{x}+t\mathbf{y}) = p(\mathbf{x}+t\mathbf{y}) = q(t)$
for $d+1$ non-zero values of t

conclusion

- Suppose we have efficient algorithm that computes $\text{perm}(M)$ correctly on at least $1 - 1/10n$ fraction of matrices in $\mathbf{F}^{n \times n}$
- Then we can also get an efficient algorithm that computes $\text{perm}(M)$ correctly on all of $\mathbf{F}^{n \times n}$

extensions

- Similar ideas for all problems complete for PSPACE, EXP, $E=DTIME(2^{O(n)})$, ...
- Give average-case complexity of the form: “every efficient algorithm makes at least a $1/\text{poly}(n)$ fraction of mistakes”
- We need: “every efficient algorithm makes at least a $1/2 - 2^{-\Omega(n)}$ fraction of mistakes”
- Average-case hardness amplification.
Difficult work by Impagliazzo, Wigderson 1995-1997
Alternative approach: Sudan T Vadhan 1999 (ask offline)

unconditional derandomization

- Nisan-Impagliazzo-Wigderson: approach of derandomization via p.r.g. requires proving circuit lower bounds
 - For problems in E, open to prove $\Omega(n \log n)$ lower bound
- Impagliazzo-Kabanets: any proof that all randomized algorithms can be derandomized implies circuit lower bounds
 - In particular, any proof that testing polynomial identities can be derandomized implies circuit lower bounds

unconditional derandomization

- Is it possible to derandomize the Jerrum-Sinclair-Vigoda approximate counting algorithm for perfect matchings?
 - derandomization not known to imply circuit lower bounds
- Worst-case complexity / average-case complexity equivalence proved for many classes: PSPACE, EXP,...
What about NP?
 - there is evidence that it's hard to show that NP contains hard-on-average problems under the assumption it contains hard-on-worst-case problems

unconditional derandomization

- [Reingold 2005] proved that Undirected Connectivity is solvable in log space and poly time deterministically
- Open Question: are all problems solvable in randomized log space (and poly time) also solvable deterministically in log space and poly time?
- Open to construct optimal pseudorandom generators against width-3 branching programs (algorithms with $\log_2 3$ bits of memory)

conclusions

- In cryptography, property testing and streaming algorithms, randomness is provably necessary
- There are rigorously motivated ways to create the required random bits from physical sources
- In two early major applications of randomness to algorithm design, randomness has been shown to give, at best, a polynomial speed-up
- Does randomness ever give more than polynomial speed-up?
- strong enough pseudorandom generators can derandomize all algorithms with polynomial slowdown
- such generators can be constructed if problems of high average-case complexity exist
- such problems exist if problems of high worst-case complexity exist
- general results can only be proved under complexity assumptions
- specific derandomization questions: counting problems, space-bounded algorithms