Lecture 3

Characterizing a source of randomness

Plan of the Lecture

Non-ideal sources and what one can do with them

1 Biased coin

2-3 Other random sources

4 Min-entropy

5 Extraction of randomness

6 **Balance of Lectures 1-3**

THE BIASED COIN

Just as useful

Definition

"Biased coin" =

- Alphabet: two values {0,1} = bit
- Single run (toss): partially unpredictable
- Several runs: uncorrelated

Single run:

$$P(0) = p > \frac{1}{2},$$

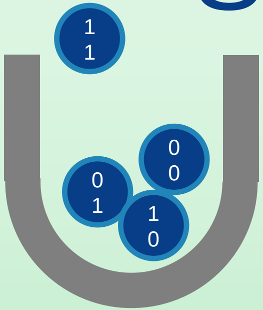
$$P(1) = 1 - p$$

Sequence of *n* runs:

Each sequence with *k* 0, *n*-*k* 1:

$$P(s_k) = p^k (1 - p)^{n-k}$$

Statistics of sequences of bits



The statistics are given by $P(k\,\mathbf{0}, n-k\,\mathbf{1}) = \binom{n}{k} p^k (1-p)^{n-k}$

$\Rightarrow P(k\,\mathbf{0}, n-k\,\mathbf{1})$ is maximum for $k \approx np$

Long sequences have almost certainly np 0 and n(1-p) 1

Nevertheless, the single most probable sequence is 00000…0, because

$$P(0000\ldots0) = p^n$$

$\Rightarrow$ If you were to bet on a *specific* sequence, this is the one to be chosen!

Extraction (von Neumann)

Can one simulate a fair coin with a biased one?

…00101001111000010010100100011000100100100111010001100100…

… 1 1 0 1 0 1 1 0 0 1 1 0 1 0 0 0 1 0 …

1. Group the bits by pairs

   The pairs 01 and 10 have the same probability $p(1-p)$

2. Replace 01→0 and 10→1; discard 00 and 11

The new list
- is shorter: average length $np(1-p)$
- is a fair coin: $P(0) = P(1) = \frac{1}{2}$

Biasing

Can one simulate a biased coin (e.g. $p = 0.75$) with a fair one?

Not valid: “toss the coin; if you see 1, discard with probability 2/3”: this assumes a biased coin with $p = 2/3$. You have <u>only</u> the fair coin.

1. Compute the binary decimal expansion $p = \sum_k q_k \frac{1}{2^k}$, $q_k = 0\ or\ 1$

   Ex: $0.75 = 1 \times \frac{1}{2} + 1 \times \frac{1}{4} + 0 \times \frac{1}{8} + 0 \times \frac{1}{16} + \ldots$ i.e. $\vec{q}(0.75) = (1,1,0,0,\ldots)$

2. Toss the coin: at the j-th toss:

   If $b_j = 0\ and\ q_j = 1$, output 0 and restart

   If $b_j = 1\ and\ q_j = 0$, output 1 and restart

   If $b_j = q_j$, move to j+1 and toss again

Prob($b_j = q_j$ for k-1 tosses)

$$\boldsymbol{P(0)} = \sum_k \boxed{\frac{1}{2^{k-1}}}\boxed{\frac{1}{2}}\boxed{q_k} \equiv \boldsymbol{p}$$

Prob($b_j \neq q_j$ for k-th toss)

Prob($q_k = 1$)

Biased vs. fair coin: summary

- Biased → fair ("extraction")
  - Possible in time $O\left(\frac{1}{p(1-p)}\right) = O(1)$.
- Fair → biased
  - Possible in time $\sum_k \frac{k^c}{2^k} = O(1)$ if the binary decimal expansion of p can be computed efficiently, i.e. q_k can be computed in time k^c (not true for all p).

⇒ The two resources are basically equivalent.

Suggested Readings

Technical references (a draft of each book can be downloaded from the links):
Salil Vadhan: Pseudorandomness
http://people.seas.harvard.edu/~salil/

Sanjeev Arora and Boaz Barak: Computational complexity
http://www.cs.princeton.edu/theory/complexity/

Other sources:

- http://en.wikipedia.org/wiki/Random_number_generation
- http://www.ams.org/samplings/feature-column/fcarc-random
- http://www.random.org/

WEAKER SOURCES OF RANDOMNESS

Information in correlations

The enemy

If fair and biased coins are equally good to generate randomness, what can possibly go wrong?

There may be correlations between the runs!

Extreme example: a process that can produce only the two sequences 00000… and 11111…

- Once you have tossed that "coin" once, its future is predictable.
- So there is at most one bit of randomness, *independently of the length n of the sequence*, instead of O(n) for uncorrelated coins (fair or biased).

Definition of correlation

Two random variables a,b are correlated if

$$P(a,b) \neq P(a)P(b)$$

or equivalently $P(a|b) \neq P(a)$, where $P(a) = \sum_b P(a,b)$ is the marginal distribution.

Applied to sequences:
A sequence of runs of a process has correlations if

$$P(b_k|b_{k-1},\dots) \neq P(b_k)$$

for some values of k.

Favorite example: Markov chain (1)

Markov chain = Correlation only with the previous draw:

$$P(b_k | b_{k-1}, \ldots) = P(b_k | b_{k-1})$$

Example:

$$P(0|0) = P(1|1) = p$$
$$P(0|1) = P(1|0) = 1 - p$$

$$\neq P(0) = P(1) = \frac{1}{2}$$

Markov chain (2)

$$P(0|0) = P(1|1) = p$$
$$P(0|1) = P(1|0) = 1 - p$$

$$p \approx 1$$

00000011111100000110000111111

Long strings of 0s and 1s

$$p \approx 0$$

01010110101010001010101001101

Very frequent alternation

$\Rightarrow$ This behavior can be detected by looking at the statistics of strings

Natural example: humans

"Can't I just generate a sequence myself? I am sure that I am not being influenced, so the process is really random".

In fact, we humans are **bad RNGs**:

- For most applications, we are slow (approx. 1 bit/sec, i.e. 1 Hz)
- And we are not very random either: after a few 0's, we "feel" that we *should* create a 1. A fair coin never feels such pressure.
  - Without warning, one can see a difference already with 3-bit sequences: 000 and 111 are less probable than the six others!
  - Now you are warned! Don't fall on the other extreme ☺

And recall: the definition of randomness is not "no external influence" or "free will", but "impossible to predict".

Perversion

One can imagine all kind of correlations:

- Every k bits, the first k-1 are fair coins, the k-th is the binary sum of the previous.
  - Hard to detect by statistical tests if k is large enough
- Toss a fair coin 1M times, then just repeat the same sequence over and over again
  - Dangerous for most applications, including Monte Carlo
- Strings of increasing length:

  011000111000001111110000000111111…

  $P(0) \approx P(1)$, but there is no randomness at all.

The message of correlations

- It's not because there are 0s and 1s that it's random.
- A *finite* battery of statistical tests detects common correlations, but may easily fail for more perverse ones.
- Perversion may not be expected in nature, but
  - We can introduce it unwillingly (see next)
  - Some scenarios are adversarial (e.g. cryptography)

Suggested Readings

- Humans are bad at generating randomness
  http://www.youtube.com/watch?v=H2lJLXS3AYM

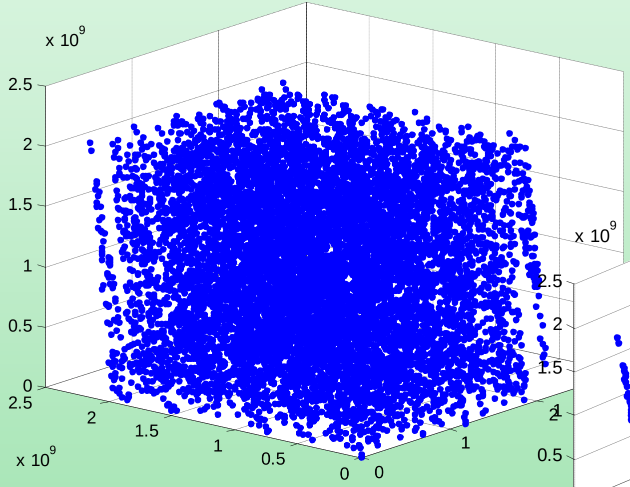
- The NIST 800-22 battery of statistical tests for random number generators
  http://csrc.nist.gov/publications/nistpubs/800-22-rev1a/SP800-22rev1a.pdf

MY FIRST RNG'S

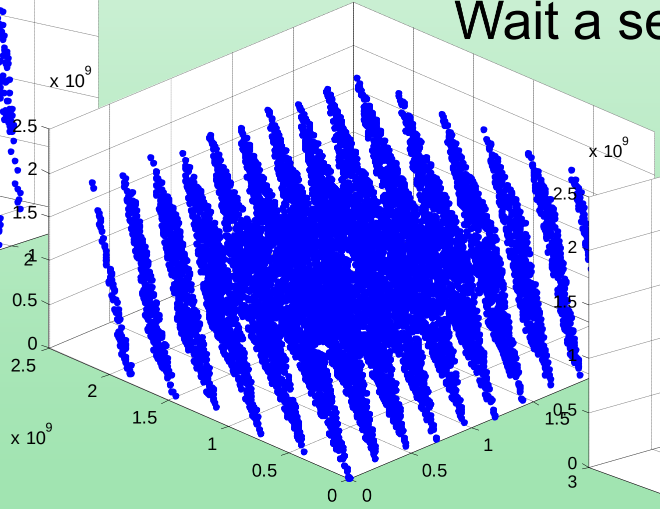
RNG = Random Number Generator

Filling space

Recall Monte Carlo software testing: sample the parameters space at random:

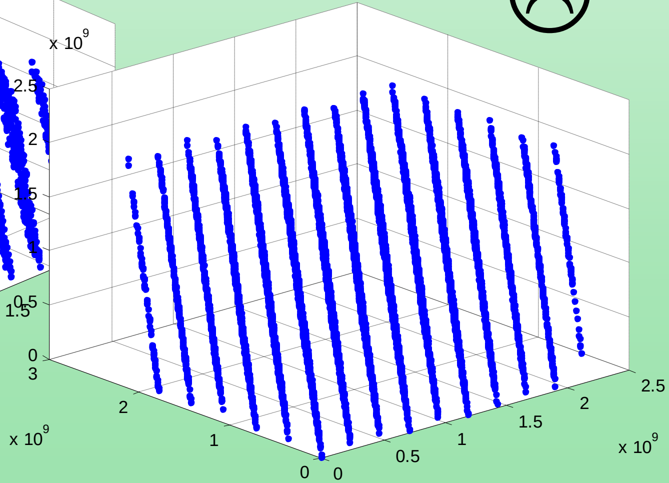


Looks good…



Wait a second…

☹



RANDU: an ill-conceived RNG

$$X_{j+1} = 65539\, X_j \; mod\; 2^{31}$$

i. Take the previous number
ii. Multiply it by 65539
iii. Take the remainder of the division by 2^{31}.

Convenient for computers with 32-bits registers

The closest prime to 2^{16}=65536

Fails the "spectral test" for D=3:

- Take triples of consecutive numbers (X_j, X_{j+1}, X_{j+2})
- Plot them in a scatter plot

⇒ they obviously don't fill the 3D space: **they define 15 2D planes**

Unwanted, "perverse" correlations

“Random numbers fall mainly in the planes” (George Marsaglia, PNAS 1968)

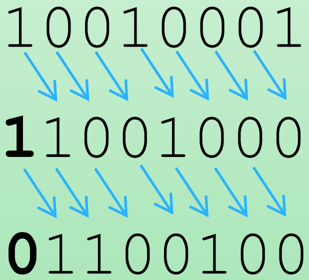
- RANDU is the bad example *par excellence[*]*, quoted in all main books on the subject.
- Widely used in the early 1970s $\Rightarrow$ some Monte Carlo optimizations computed in those years are doubtful.
- Marsaglia: consecutive random numbers fall on (maybe higher-dimensional) planes for all “*linear congruential generators*”

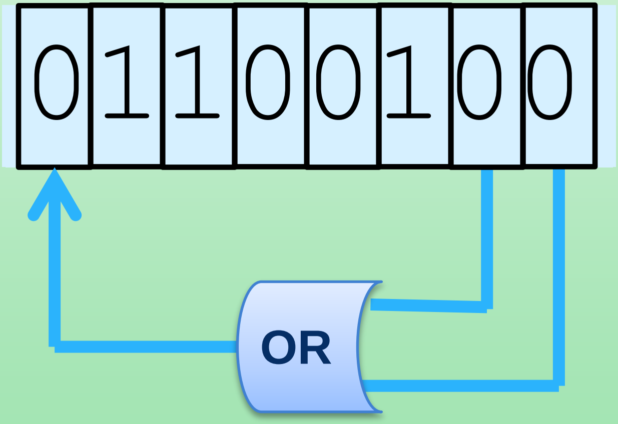
$$X_{j+1} = (aX_j + c)\ mod\ M$$

[*] The recent case of Dual-EC may make history too, for secrecy instead of Monte Carlo

Better RNG: LFSR

A better, widely use pseudo-random number generation technique uses **Linear Feedback Shift Registers**:





In practice:

- More complex calculation than OR
- 16 or 32 registers

Suggested Readings

G. Marsaglia, Random numbers fall mainly in the planes, PNAS 1968 61 (1) 25-28
http://www.pnas.org/content/61/1/25.citation

On the recent issue with Dual EC:
http://blog.cryptographyengineering.com/2013/09/the-many-flaws-of-dualecdrbg.html?

Wikipedia pages:

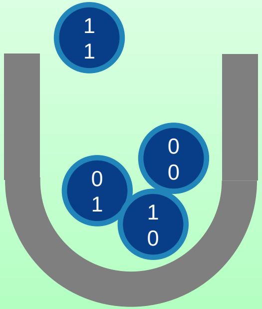
- The first (and still used) long list of random numbers:
http://en.wikipedia.org/wiki/A_Million_Random_Digits_with_100,000_Normal_Deviates
- http://en.wikipedia.org/wiki/List_of_random_number_generators
- http://en.wikipedia.org/wiki/Linear_feedback_shift_register

MIN-ENTROPY

The amount of randomness

How much randomness?

A good quantifier: guessing probability



If you are asked to bet on a particular sequence to be drawn, which one would you guess?

Obviously the best choice is **the sequence with largest probability**!

$$P_{guess} = \max_{s} P(s)$$

Min-entropy

Two unpleasant features of P_{guess} as a quantifier of randomness:

1. The larger P_{guess} , the smallest the amount of randomness (indeed $P_{guess} = 1$ means no randomness)
2. Random coin: n tosses = n random bit, but $P_{guess} = \frac{1}{2^n}$.

Amount of randomness = min-entropy:

$$H_{min} = -\log_2 P_{guess}$$
$$\equiv -\log_2 \max_s P(s)$$

Unit: bits

Name: in the family of “Renyi entropies”, it gives the minimal value H(P) for any given P

Examples

Process	P_{guess}	Sequence	H_{min}
Fair coin	$\frac{1}{2^n}$	any	n
Biased coin	p^n (*)	0000000…	$n\log_2\frac{1}{p}$
Markov example $P(0\|0) = P(1\|1) = p$	p^n (*)	0000000… or 1111111…	$n\log_2\frac{1}{p}$

<*n* for p>1/2
=*0* for p=1

$^*\, p > \frac{1}{2}$

A refinement

Randomness = ignorance

$\Rightarrow$ It is not the *observed* probability that matters, but **the probability of guessing *conditioned to someone's knowledge.***

Examples:

- The toss of the coin may be unpredictable for me, predictable for someone who can describe the motion exactly;
- In crypto, a seed should be random for Eve, not for Alice

Conditional min-entropy

$$H_{min}(P|E) = -\log_2 \max_s P(s|E)$$

Operational interpretation of H_{min}

Recall von Neumann extraction:
From a sequence of n bits of a biased coin, one can extract a sequence of O(n) bits [on average, np(1-p)] of a fair coin.

n bits with min-entropy $H_{min}(P|E)$ $\xrightarrow{\exists \text{ Extraction}}$ $m = H_{min}(P|E)$ bits of a fair coin.

- "**Leftover hash lemma**" (Impagliazzo, Levin, Luby 1989)
- In words: the min-entropy is the amount of "ideal randomness" that can be extracted out of the initial data.

LET’S DO AN EXTRACTION

What extraction?

n bits with min-entropy $H_{min}(P|E)$ $\xrightarrow{\exists \text{ Extraction}}$ $m = H_{min}(P|E)$ bits of a fair coin.

Claim: possible without any other information about the RNG.

Example: $H_{min}(P|E) = 0.1n$ could be:

- A source producing one fair-coin bit, then replicating it 9 times before tossing the fair coin again…
- A source that behaves like a fair coin 10% of the time, say in the first six minutes of each hour on the clock, then just produces 0's…
- A biased, uncorrelated coin with $p = 2^{-0.1} \approx 0.933\ldots$ Etc.

$\Rightarrow$ How to do the extraction in practice?

“Strong” extraction

Final sequence
Length m
Min-entropy m

Extractor: matrix
nxm **random** bits

Initial sequence
Length n
Min-entropy m<n

$$\begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix} = \begin{pmatrix} M_{11} & \cdots & M_{1n} \\ M_{m1} & \cdots & M_{mn} \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ \vdots \\ a_n \end{pmatrix}$$

You need a fair coin! Does this not defeat the purpose?

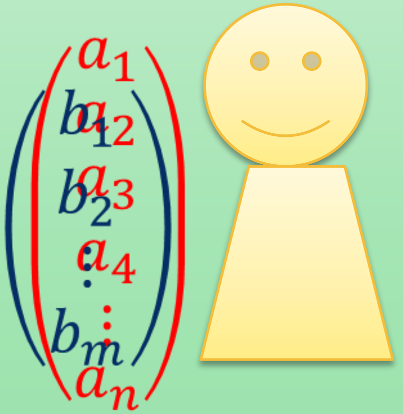
Think crypto

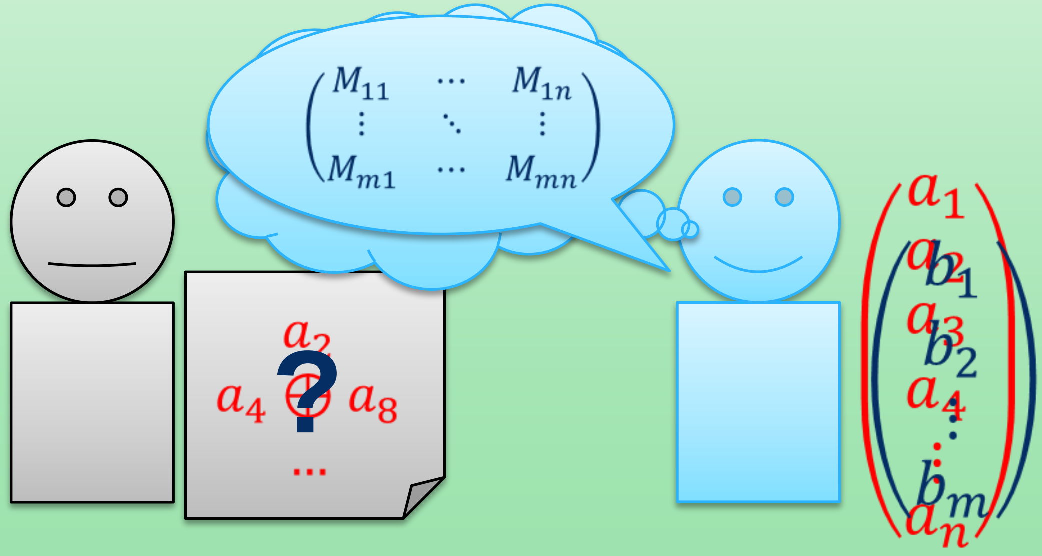
The initial list, meant to be secret, is partly compromised:

$$H_{min}(A|E) = m < n$$

Bob can generate M with *his own coin* $\Rightarrow$ it will be uncorrelated from Eve's attack. $\Rightarrow$ He can even reveal it publicly

$\Rightarrow$ Eve's information is erased.





The intuition

Suppose that Eve knows a_3 exactly, and nothing of the others:

$$\begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \end{pmatrix}$$

$\Rightarrow$ For this example, Eve knows b_k if and only if the k-th line is 001000

Recap of extraction

- Only one source of randomness $\Rightarrow$ need to know its structure
  - Example: uncorrelated $\Rightarrow$ von Neumann, for whatever p
- Add a coin $\Rightarrow$ extract from any source knowing only H_{min}
  - Crypto: extract your secret using a coin
  - Non-crypto, only Bob: the nxm bits of the matrix are not spoiled, you can still use them, and you got m more.
- Remark: the rigorous results are *statistical:*
  - If you can afford failing with probability ε, you can extract

$$m = H_{min} - \log\frac{1}{\varepsilon}$$

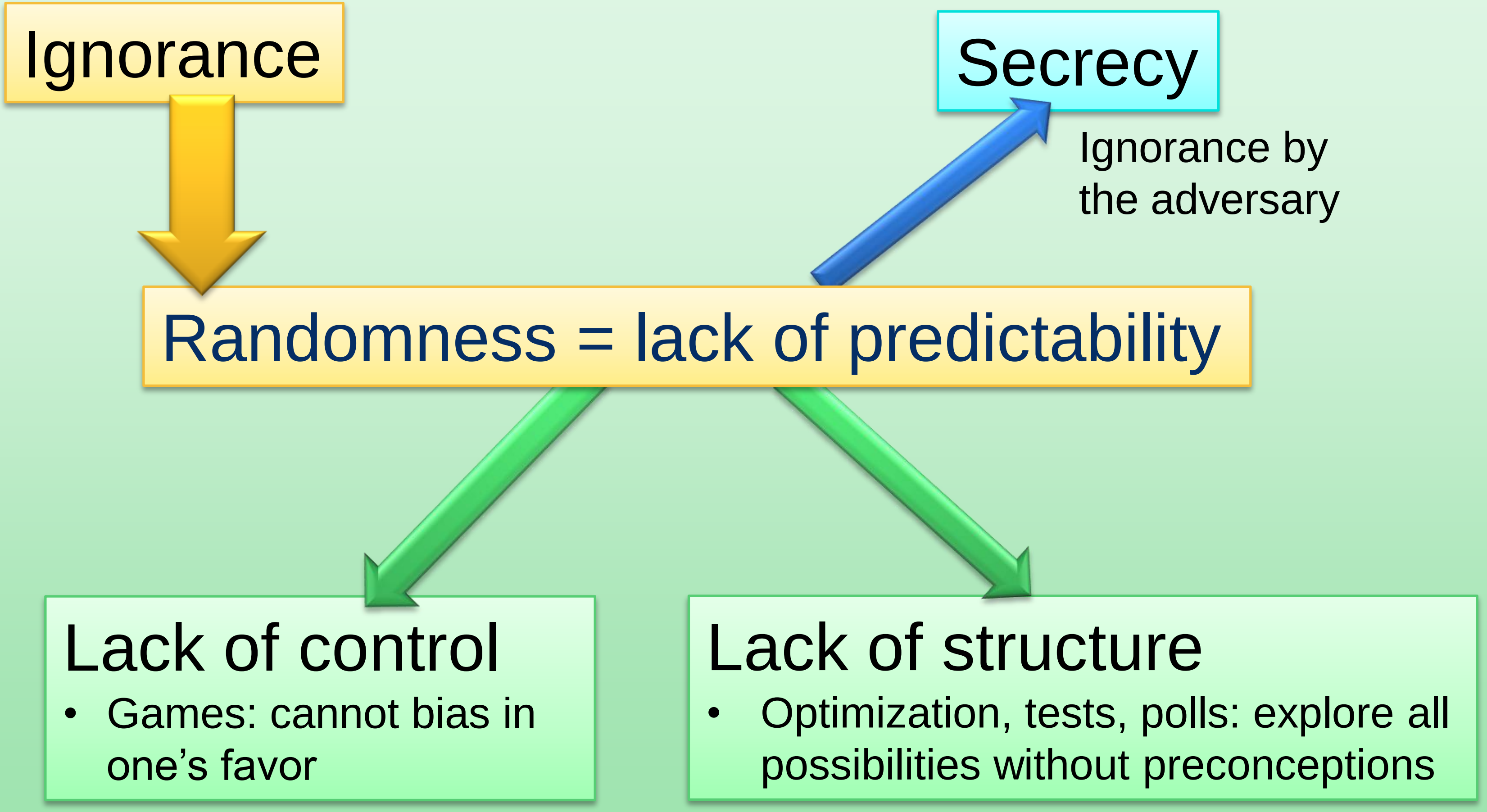
$$\boxed{\varepsilon = 10^{-9} \Rightarrow \log\frac{1}{\varepsilon} = 30}$$

Summary of Lecture 3

Non-ideal sources and what one can do with them

- Weaker sources of randomness
  - Biased coin
  - Correlations (Markov, humans and beyond)
  - Examples: RANDU, Linear Feedback Shift Registers
- Min-entropy as amount of randomness
  - Operational: extractable amount of ideal randomness
- Extraction of randomness

BALANCE OF LECTURES 1-3


Ignorance
Secrecy
Ignorance by the adversary
Randomness = lack of predictability
Lack of control
• Games: cannot bias in one’s favor
Lack of structure
• Optimization, tests, polls: explore all possibilities without preconceptions

Ignorance: By whom? Of what?

What is unpredictable for me may not be unpredictable for you

- Elements of trust
  - Non-adversarial: after these tests, I am confident enough that there is no structure (bad case: RANDU)
  - Adversarial: I trust that the key has no leaked out, that the adversary has limited computational power…
- Individuals vs. populations
  - A population may behave “at random” even if each individual is behaving deterministically

Sources of randomness

Pseudo-random sources: complicated algorithms with choice of seed

- $X_{j+1} = a\, X_j \bmod M$
- LFSR
- Scrambling of the digits of the computer clock when you touch a key
- Etc.

"True" randomness: unpredictable physical processes

- Coins, dice etc.
- "Noise", "fluctuations" (lecture 4)
- "Chaos" (lecture 5)
- Quantum: *intrinsically* unpredictable (lectures 6-7)