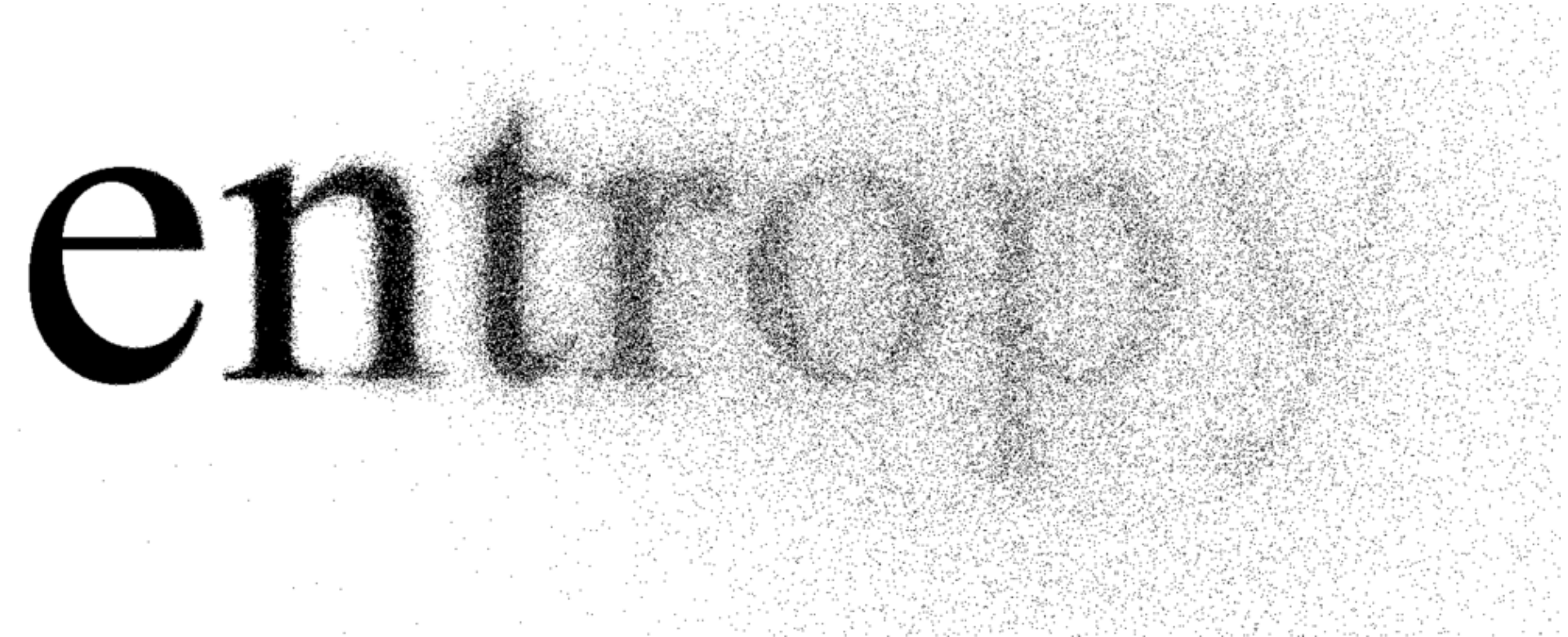
Random Number Generation (Entropy)

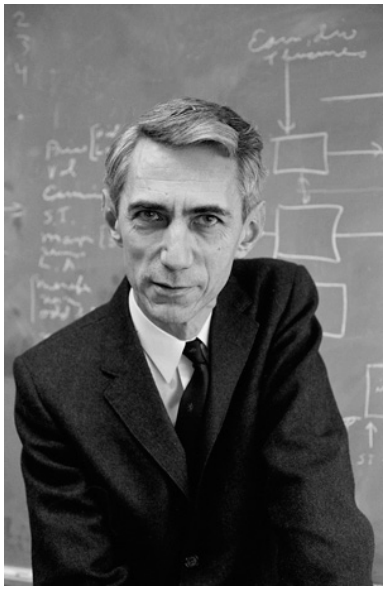
Entropy (Thermodynamics)

Measure of *disorder* of any system

Entropy will only increase over time! (2nd law of thermodynamics)



Entropy (Information Science)



- **Shannon Entropy**
- Measure of *unpredictability* of a state
  - Average information content
- Low entropy
  - Very predictable
  - Not much information contained
- High entropy
  - Unpredictable
  - Significant information contained

Desired for cryptography!

Donald Knuth

- Author, *The Art of Computer Programming*
  - Algorithms!
- Creator of TeX typesetting system
- Winner, ACM *Turing Award*, 1974

"Random numbers should not be generated with a method chosen at random." – Donald Knuth

The Market

- High demand for random numbers in cryptography
  - Keys
  - Nonces
  - One-time pads
  - Salts

Great Disasters in Random Numbers

- Netscape Navigator (1995)
  - Random numbers used for SSL encryption
  - Export laws (at time) limited international version to 40 bit key lengths
- Algorithm `RNG_CreateContext()`
  - Generates seed for RNG
  - Get time of day, process ID, parent process ID
  - Hash with MD5
- All predictable!
  - Brute force in much fewer than 2^{40} attempts, which was already weak (< 30 hours)

https://people.eecs.berkeley.edu/~daw/papers/ddj-netscape.html

Great Disasters in Random Numbers

- Debian Linux (2006-2008) - **CVE-2008-0166**
  - OpenSSL package (specific Debian/Ubuntu variant)
  - Developer commented out two lines of code because they caused uninitialized memory warnings in Valgrind/Purify code review tools
    - `MD_Update(&m,buf,j);`
  - Result: Only "random" value used in seeding OpenSSL PRNG was the current process ID! (integer from 0-32768)
    - Lets attackers guess your **private** SSH and SSL keys

https://www.schneier.com/blog/archives/2008/05/random_number_b.html

Caprica, 2 years ago

http://blog.dieweltistgarnichtso.net/Caprica,-2-years-ago

Random Number Generators

- Produce a sequence of numbers with following properties:
  - New value must be *statistically independent* of previous value
    - Particular values should not be more or less likely
  - Distribution of numbers is *uniformly distributed*
    - Cannot have some values more or less likely
  - Sequence is *unpredictable*
    - Cannot guess next value based on current or past values
    - Cannot guess previous values based on current value
- Other desirable features:
  - Fast! (can produce many random numbers quickly)
  - Secure against attackers (cannot observe/modify underlying state)

Random Number Generators

- Options
  1. True random number generators
  2. Cryptographically secure pseudorandom number generators (PRNG)
  3. Pseudorandom number generators (PRNG)

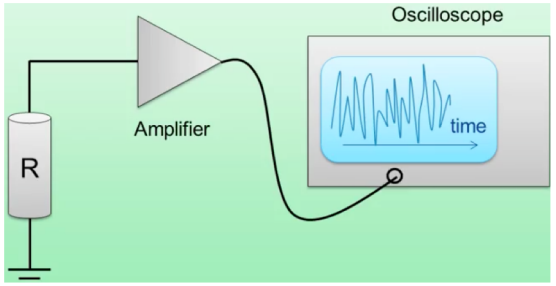
https://www.zerodayclothing.com/

True Random Number Generators

Real-world has true randomness. Algorithms do not
(unless the algorithm data is based on real world events)

Dice Rolls

Radioactive Decay
(Measure interval between
successive counts of a Geiger counter)
https://www.fourmilab.ch/hotbits/



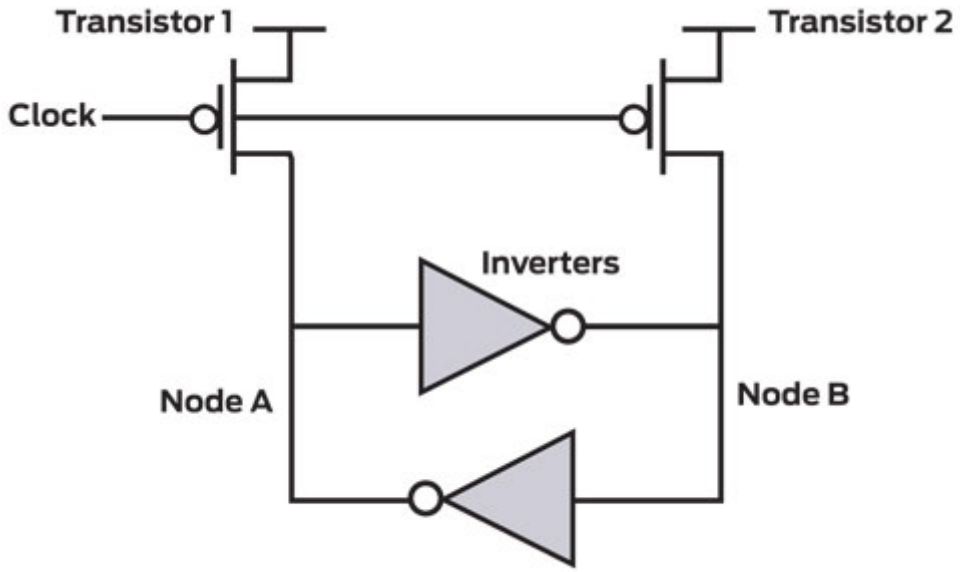
Thermal Noise (Nyquist noise)
(Measure voltage change due to
electrons moving through medium)
https://www.fourmilab.ch/hotbits/

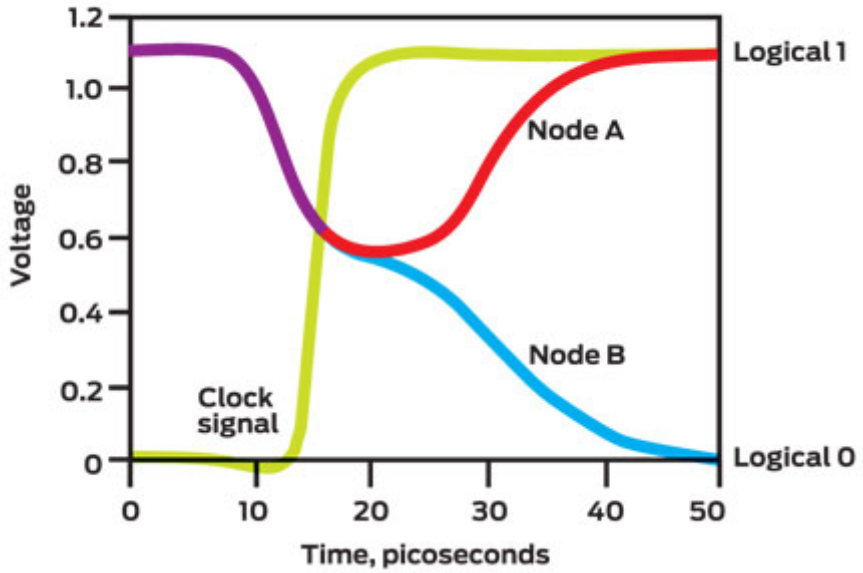
True Random Number Generators

- Atmospheric noise
  - Below 30MHz
  - Generated by lightning
  - Varies by day/weather/location
- https://www.random.org/

True Random Number Generators

- Idea: Break key digital design rule of "Circuit should always be in known state". *Metastability* is a feature!





https://spectrum.ieee.org/computing/hardware/behind-intels-new-randomnumber-generator

True Random Number Generators

- "Wall of Entropy" by Cloudflare
- Take picture every millisecond
- Convert image into random data
- 16,384 bits of entropy from each picture

https://www.fastcodesign.com/90137157/the-hardest-working-office-design-in-america-encrypts-your-data-with-lava-lamps

Entropy Pool (Computer Systems)

- Observed phenomena, not natural events
- Measure time with 100 nanosecond resolution using processor clock
  - Time of all threads spent in user mode?
  - Time of all threads spent in the kernel?
  - Time the scheduler spent idle?
- Least significant bits of RDTSC (CPU cycle counter)
- Instantaneous snapshot of global memory usage statistics
- X and Y position of the mouse
- Inter-interrupt timings
- Inter-arrival time of network packets

Cryptographically Secure PRNG

- All the goals of any random number generator (*statistically independent*, *uniform distribution*, …) PLUS being resistant to attack
  - Resistant to attackers calculating the past or the future from the present
  - Resistant to attackers acquiring internal state
  - Resistant to attackers interfering/modifying the entropy generation/collection

Cryptographically Secure PRNG

Don’t write your own!

Use the service provided by your OS

(more eyes on code, easy to update)

Cryptographically Secure PRNG

- Linux: **`/dev/urandom`** preferred or **`getrandom()`**
  - `/dev/random` will block if entropy exhausted
  - https://www.2uo.de/myths-about-urandom/
- FreeBSD,OSX,iOS: **`/dev/urandom`** or **`/dev/random`**
  - Identical – based on **Yarrow algorithm**
  - John Kelsey, Bruce Schneier, Niels Ferguson

- Windows: **`CryptGenRandom()`**
  - AES counter-mode based PRNG specified in NIST Special Publication 800-90
- Programming language should expose OS primitives
  - Example: Python `os.urandom` and `random.SystemRandom` provide identical values

PRNG

- Not cryptographically random
  - **Mersenne Twister** – **Don't Use!**
    - Great random numbers, just **not** for cryptography
      - Mersenne Twister MT19937 PRNG with 32-bit word length has a periodicity of 2^{19937}-1 (won't repeat for a long time)
    - Completely *deterministic* - possible to predict future or recover past values from current value
      - Only need to observe 624 values from MT19937
  - `srand()/rand()/random()` in C standard library
  - And many similar examples…

Testing

A Pseudorandom Number Sequence Test Program

This page describes a program, **ent**, which applies various tests to sequences of bytes stored in files and reports the results of those tests. The program is useful for evaluating pseudorandom number generators for encryption and statistical sampling applications, compression algorithms, and other applications where the information density of a file is of interest.

NAME

ent - pseudorandom number sequence test

SYNOPSIS

ent [**-b -c -f -t -u**] [*infile*]

- Tests: Chi-square test, arithmetic mean, Monte Carlo Value for Pi, Serial Correlation Coefficient

https://www.fourmilab.ch/random/