ndonmRseas

nsenRdamso

esdnRsaonm

nsnasemdRo

emnosandsR

Randomness

dasnnmseoR

anmdRonsse

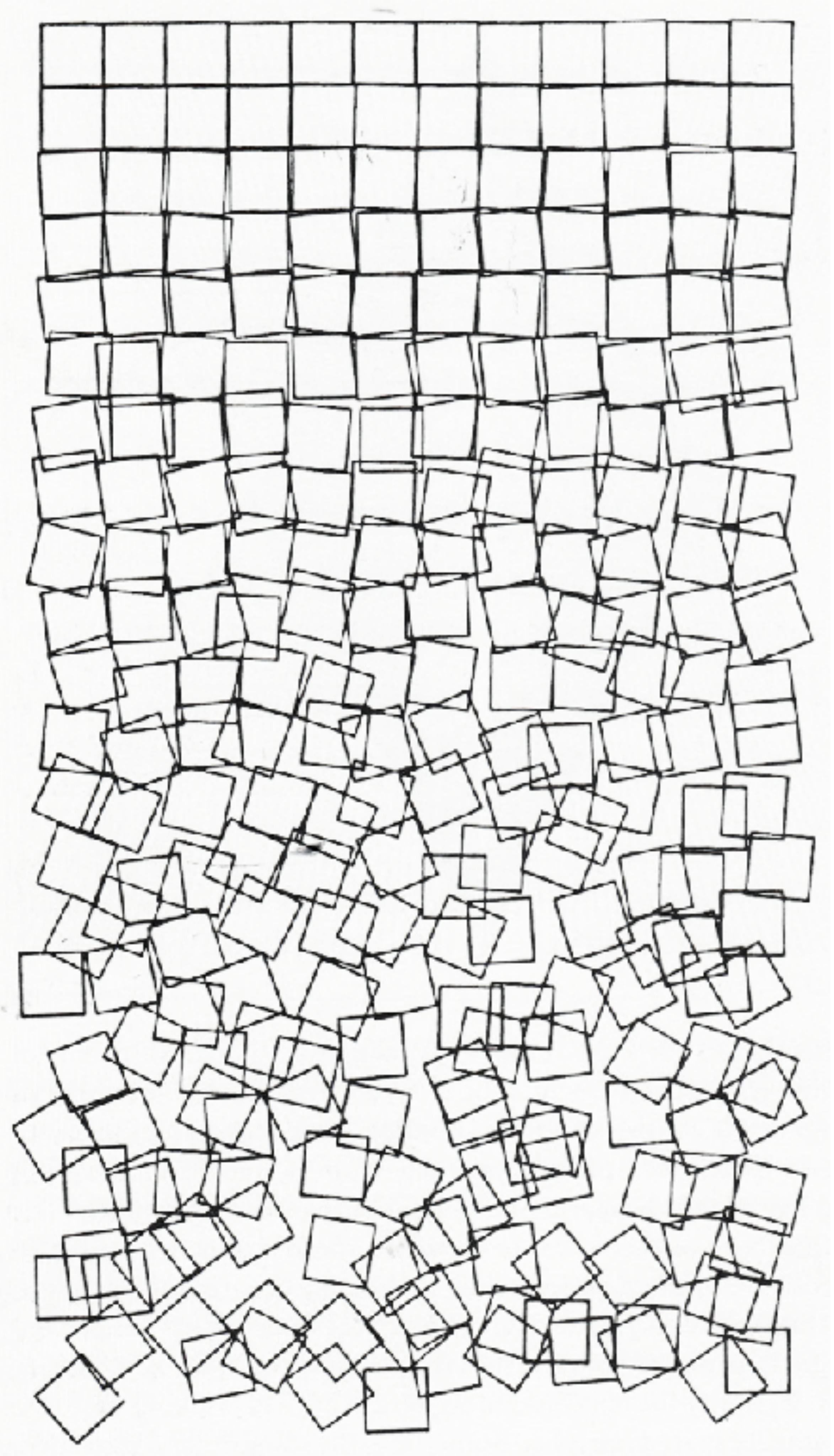
sRedmnaons

nndaoRssme

Module 08
CS 106 Winter 2018

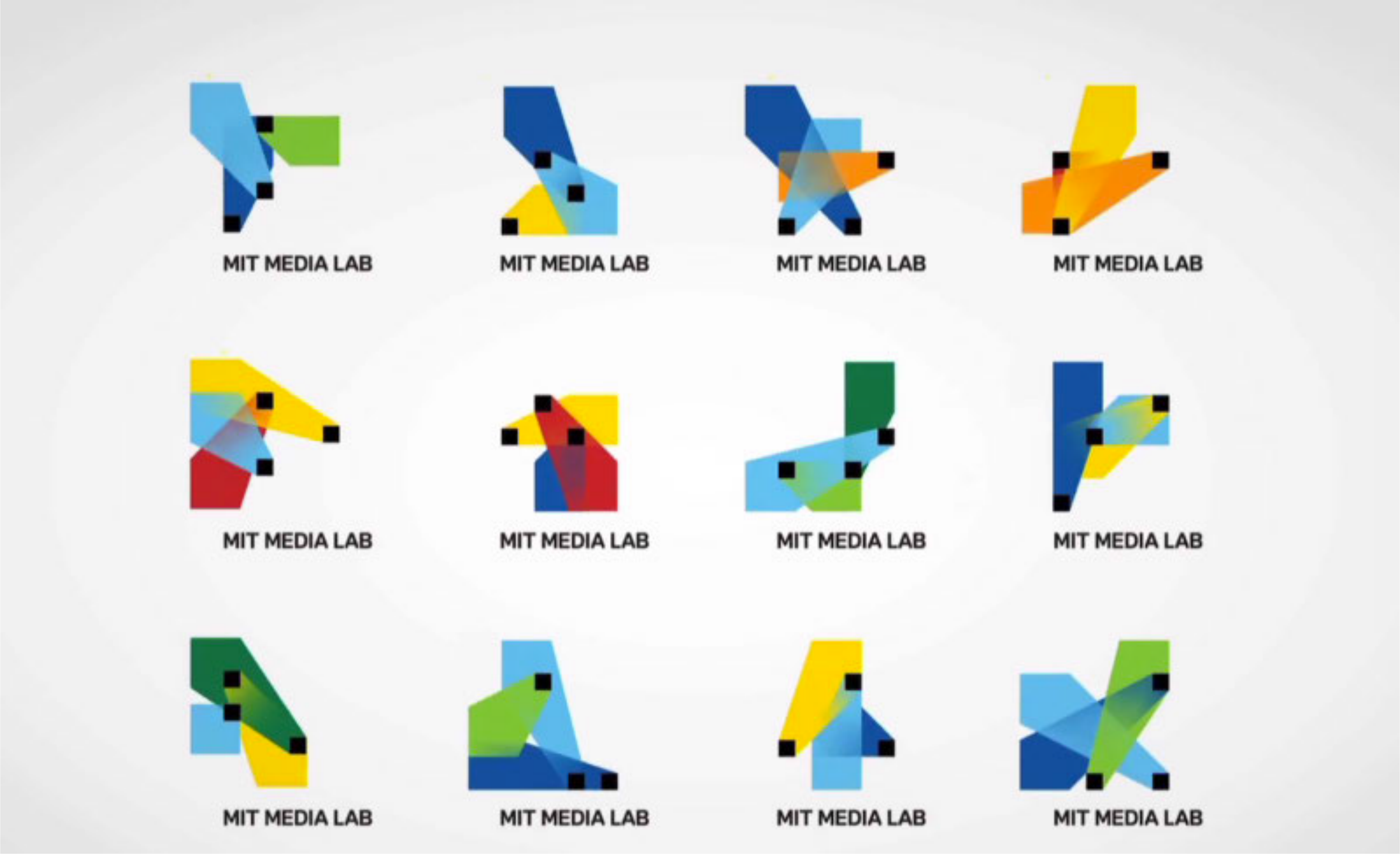
“Maybe the greatest novelty here is the ability of the computer not only to follow any complex rule of organization but also to introduce an exactly calculated dose of randomness.”

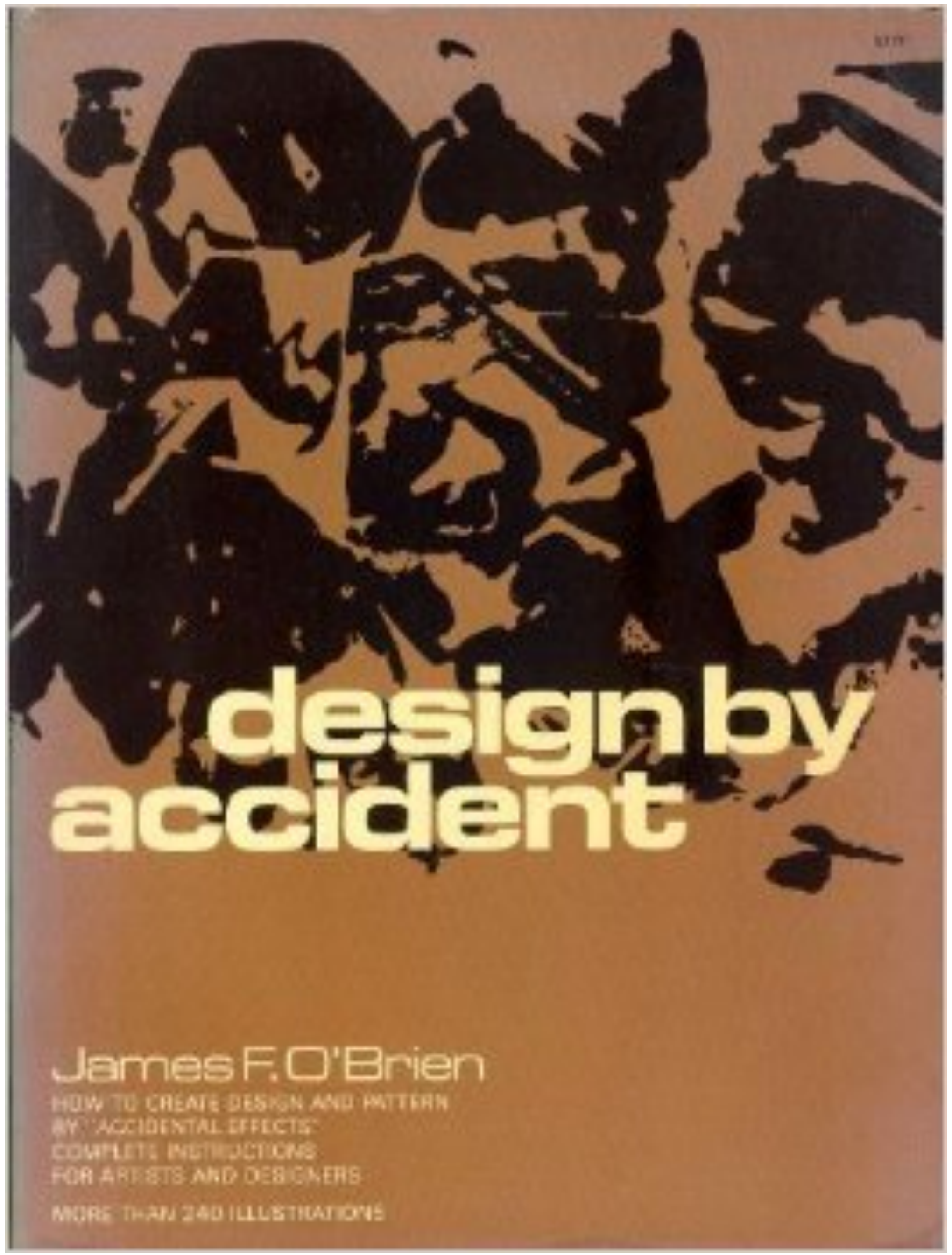
— E.H. Gombrich



Georg Nees, *Gravel Stones* (1971)

Generative Art



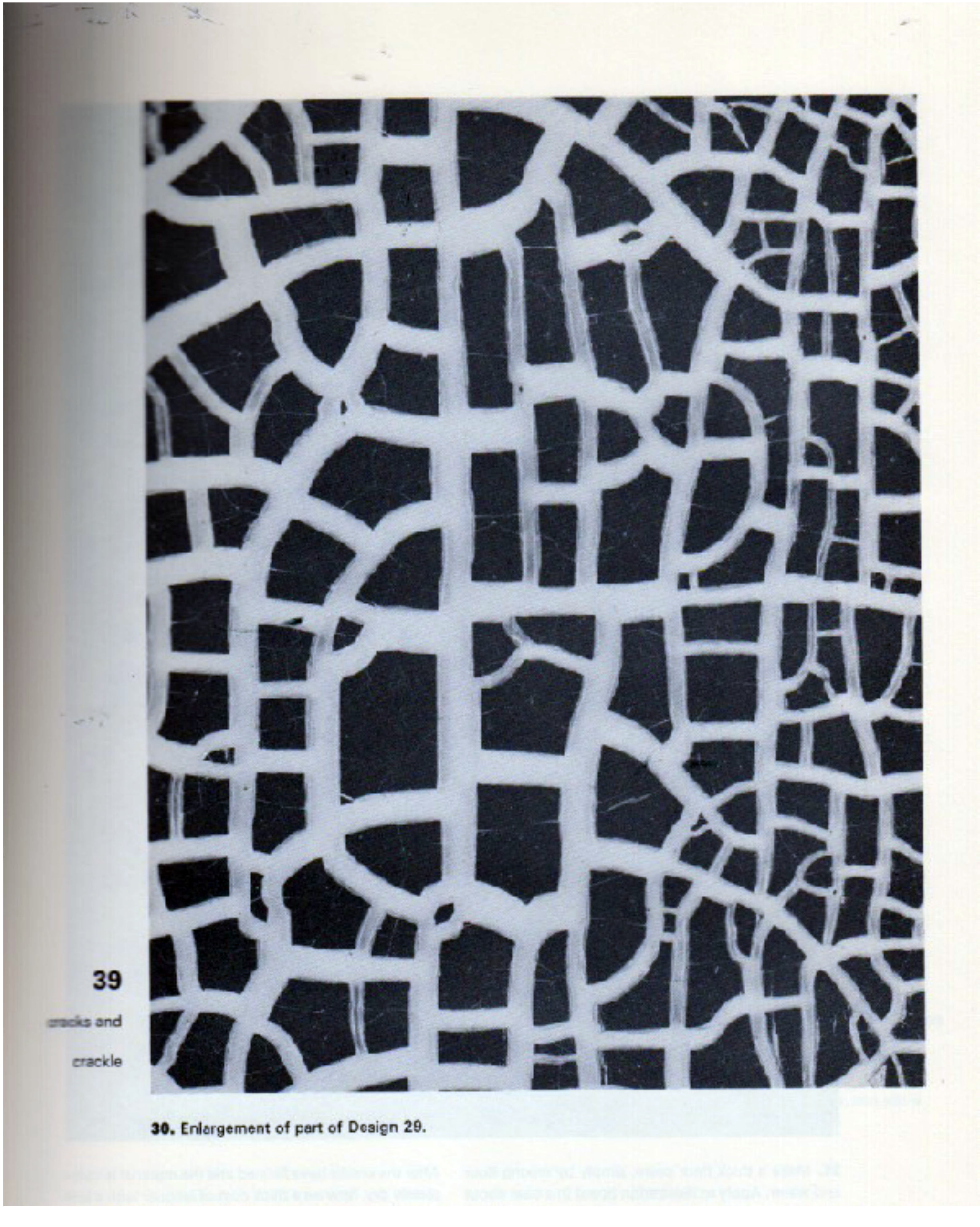


design by accident

James F. O'Brien

HOW TO CREATE DESIGN AND PATTERN
BY "ACCIDENTAL EFFECTS"
COMPLETE INSTRUCTIONS
FOR ARTISTS AND DESIGNERS

MORE THAN 240 ILLUSTRATIONS



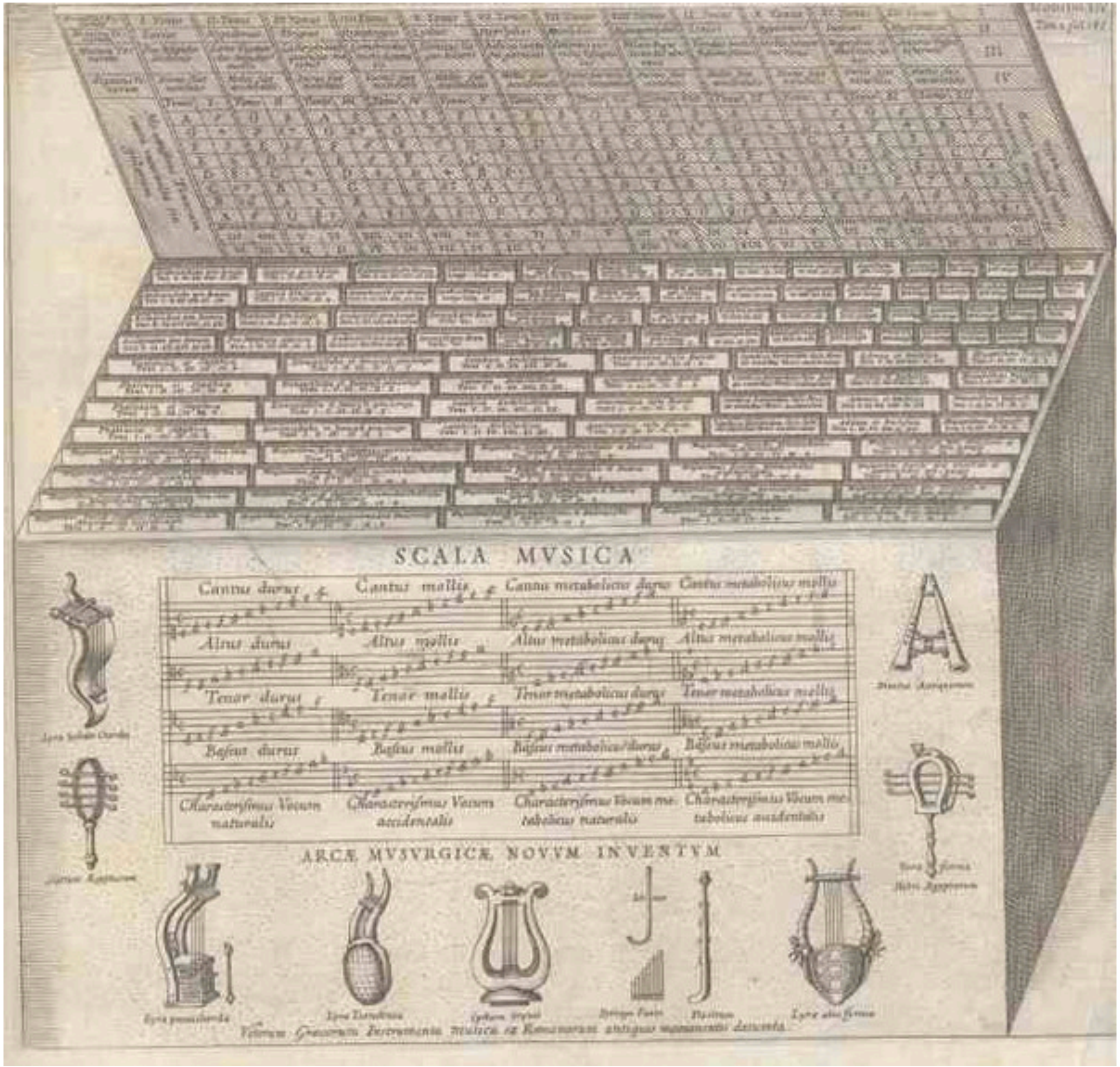
39

cracks and

crackle

30. Enlargement of part of Design 29.

Athanasius Kircher (1602–1680)


SCALA MVSICA
Cantus durus
Cantus mollis
Cantus metabolicus durus
Cantus metabolicus mollis
Altus durus
Altus mollis
Altus metabolicus durus
Altus metabolicus mollis
Tenor durus
Tenor mollis
Tenor metabolicus durus
Tenor metabolicus mollis
Baſſus durus
Baſſus mollis
Baſſus metabolicus durus
Baſſus metabolicus mollis
Characteriſmus Vocum naturalis
Characteriſmus Vocum accidentalis
Characteriſmus Vocum metabolicus naturalis
Characteriſmus Vocum metabolicus accidentalis
ARCÆ MVSVRGICÆ NOVVM INVENTVM
Veterum Græcorum Instrumenta musica ex Romanorum antiquis monumentis desumta

Arca Musarithmica

Sol Lewitt (1928–2007)

Plate 1. Within a twenty inch square area, using a black, hard crayon, draw ten thousand freehand lines, of any length, at random.

Plate 2. Within a twenty inch square area, using a black, hard crayon, draw ten thousand straight lines, of any length, at random.

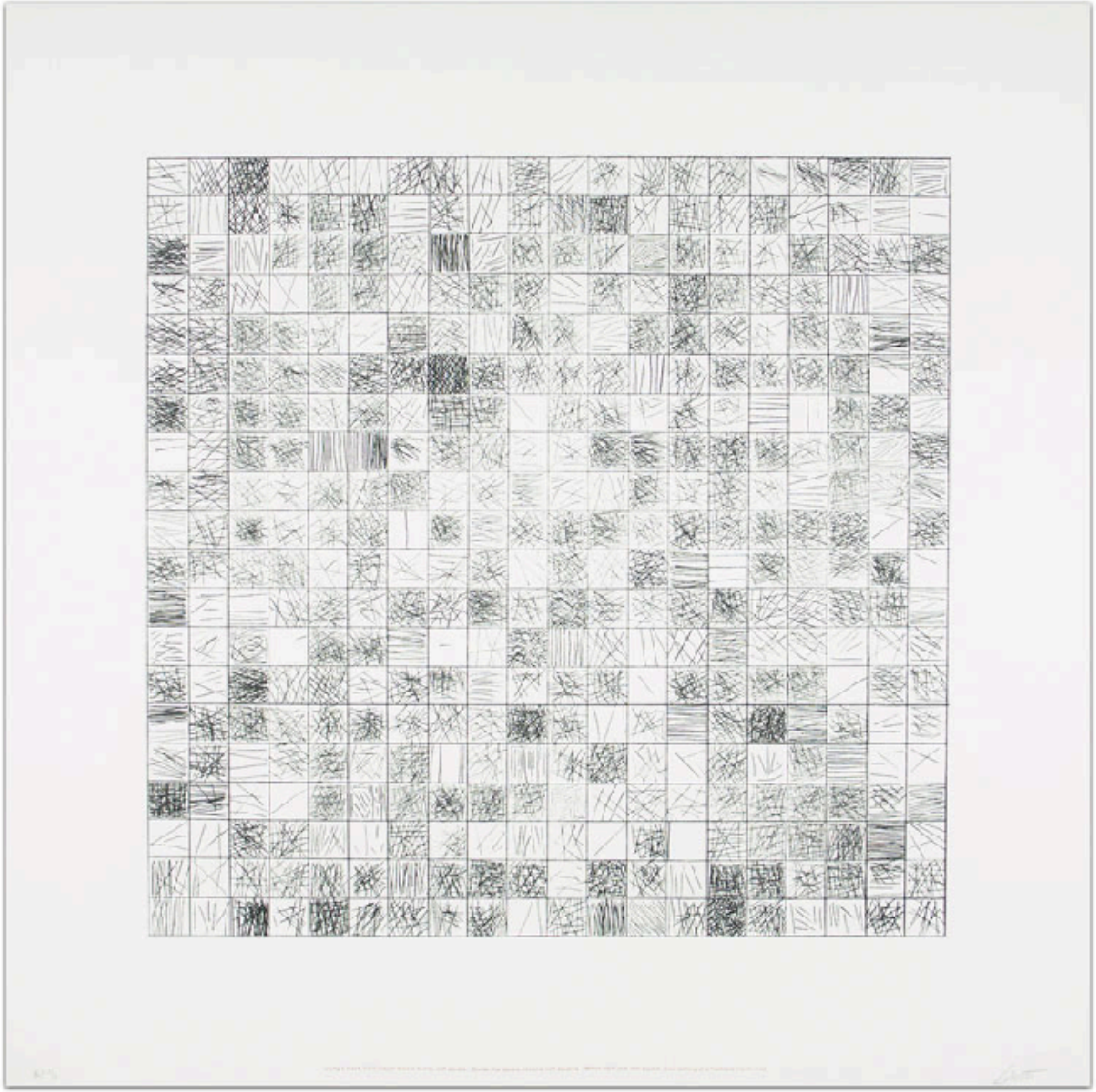
Plate 3. Using a black, hard crayon draw a straight line of any length. From any point on that line draw another line perpendicular to the first. From any point on the second line draw another line perpendicular to that line. Repeat this procedure.

Plate 4. Using a black, hard crayon, draw four contiguous ten inch squares, forming one twenty inch square, divided into quarters. Within the first square (upper left) draw one line, one inch long. Within the second square (upper right) draw ten lines, each one inch long. Within the third square (lower left) draw one hundred lines, each one inch long. Within the fourth square (lower right) draw one thousand lines, each one inch long. All lines should be drawn at random, and straight.

Plate 5. Using a black, hard crayon draw a twenty inch square. Divide this square into one inch squares. Within each one inch square, draw nothing or a freehand line or lines.

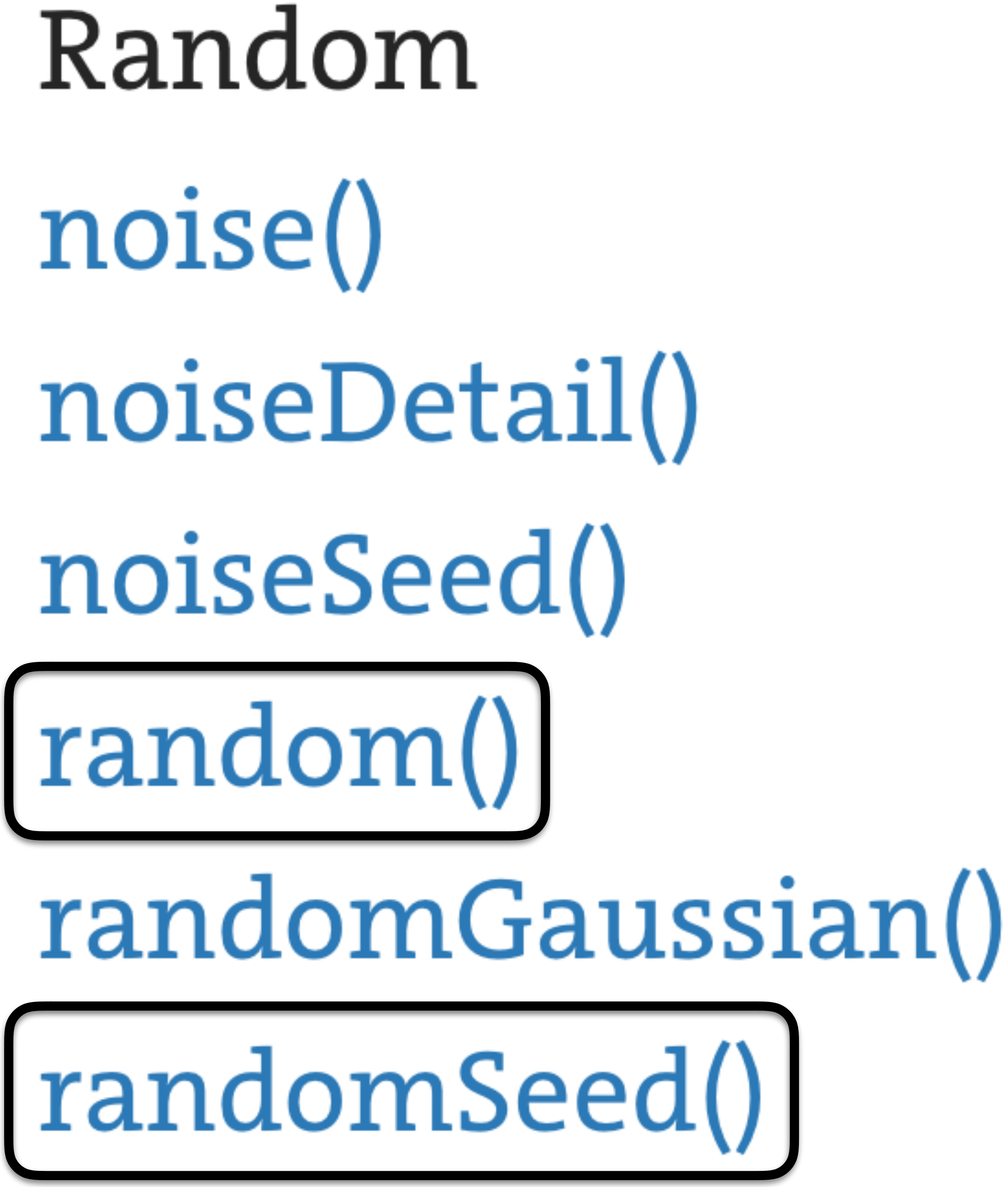
Plate 5. Using a black, hard crayon draw a twenty inch square. Divide this square into one inch squares. Within each one inch square, draw nothing or a freehand line or lines.

Plate 5. Using a black, hard crayon draw a twenty inch square. Divide this square into one inch squares. Within each one inch square, draw nothing or a freehand line or lines.



Random

- noise()
- noiseDetail()
- noiseSeed()
- random()
- randomGaussian()
- randomSeed()



Random

- noise()
- noiseDetail()
- noiseSeed()
- random()
- randomGaussian()
- randomSeed()

```
float random( float lo, float hi ) { ... }
```

Return a random number at least as big as `lo` but smaller than `hi`.

Get a different answer every time!

```
float random( float lo, float hi ) { ... }
```

Return a random number at least as big as `lo` but smaller than `hi`.

Get a different answer every time!

```
float random( float hi )
{
  return random( 0, hi );
}
```

Random integers

```
int( random( N ) )
```

Choose a random integer from the set
0, 1, ...

Random integers

```
int( random( N ) )
```

Choose a random integer from the set 0, 1, ... **N-1**

(The `int()` function always rounds down)

Flipping a coin

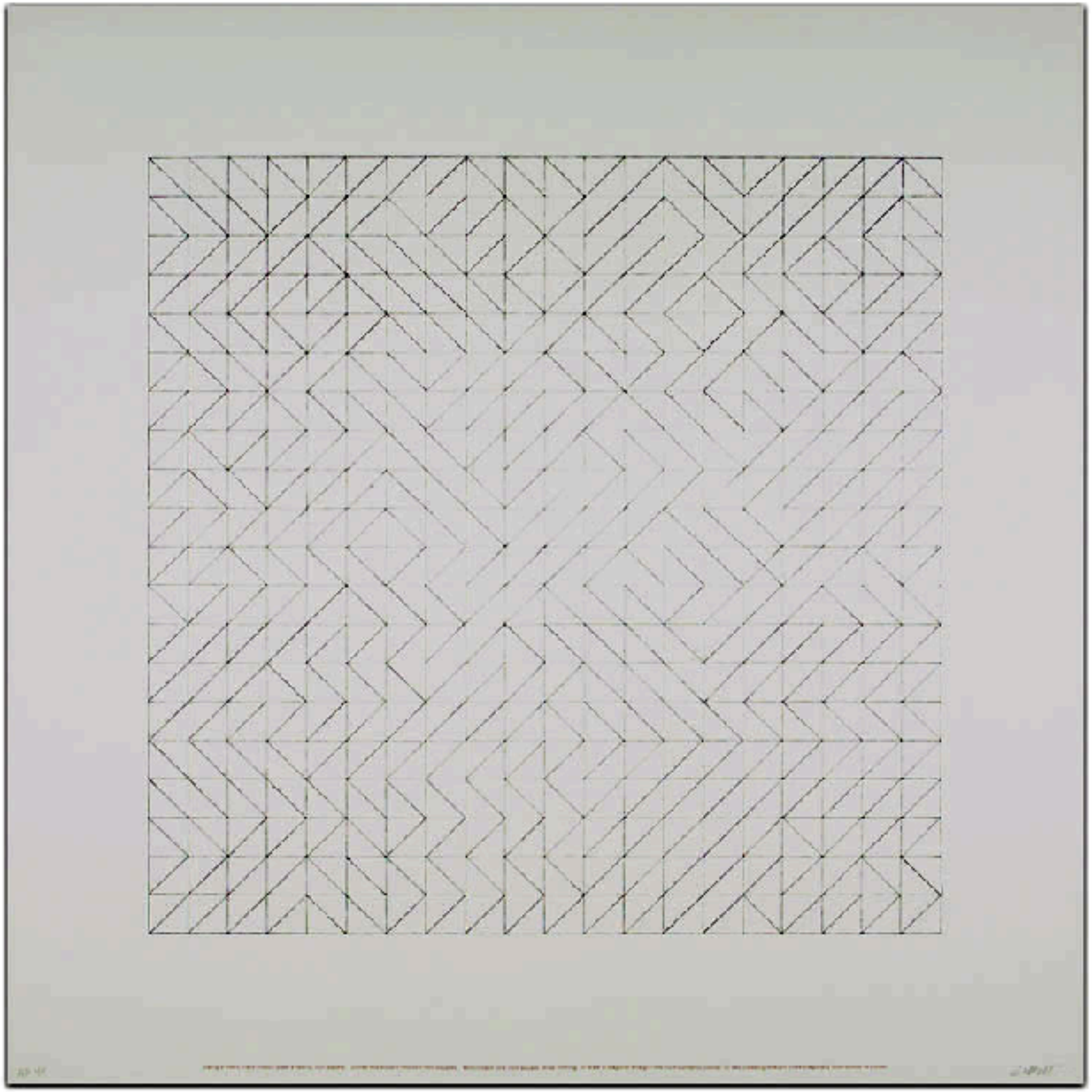
Write a function that simulates flipping a fair coin.

Flipping a biased coin

What if we wanted to get heads 75% of the time and tails 25% of the time?

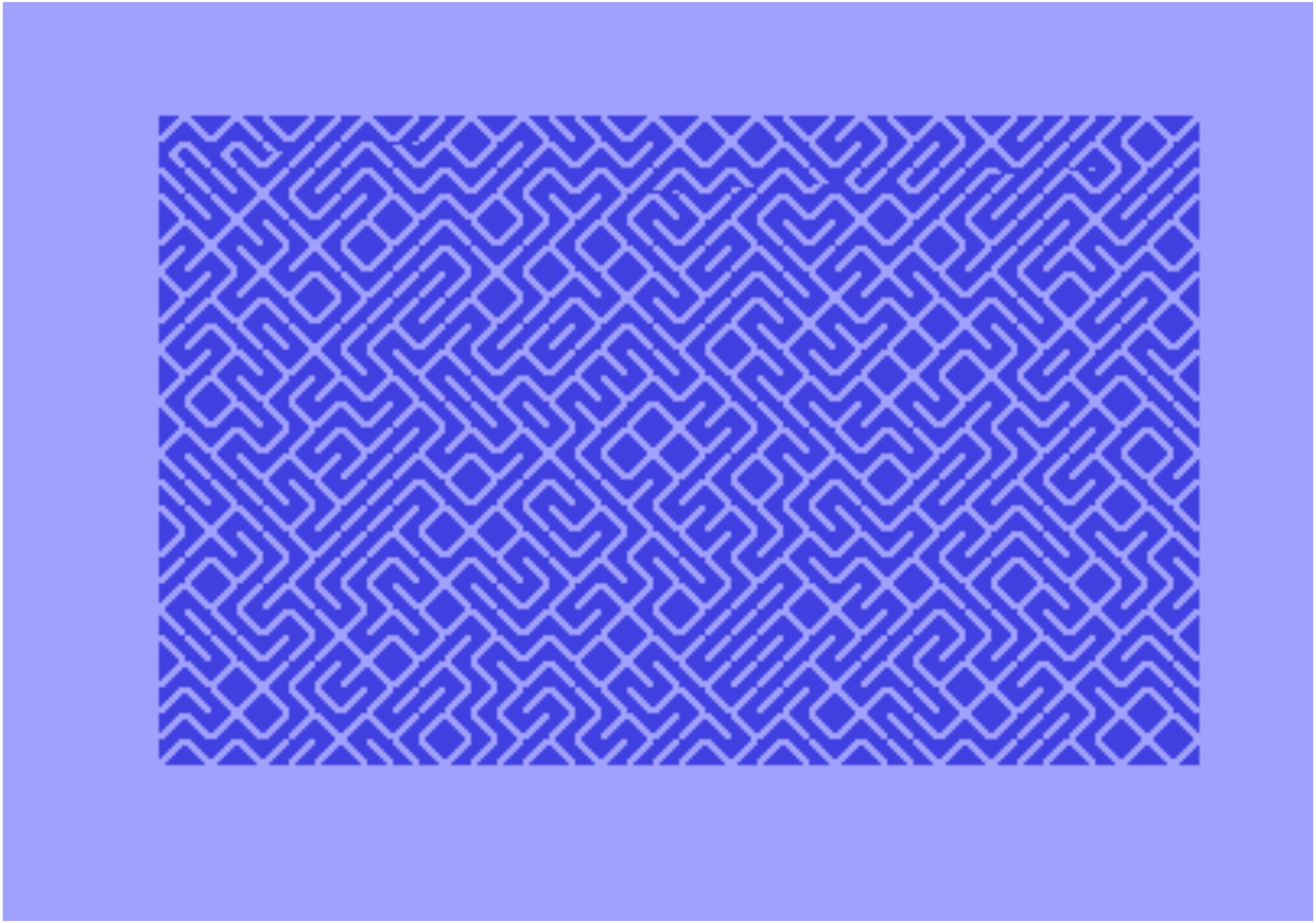
Plate 6. Using a black, hard crayon draw a twenty inch square. Divide this square into one inch squares. Within each one inch square, draw nothing, or draw a diagonal straight line from corner to corner, or two crossing straight lines diagonally from corner to corner.

Plate 6. Using a black, hard crayon draw a twenty inch square. Divide this square into one inch squares. Within each one inch square, draw nothing, or draw a diagonal straight line from corner to corner, or two crossing straight lines diagonally from corner to corner.



```
10 PRINT CHR$(205.5+RND(1)); : GOTO 10
```

```
10 PRINT CHR$(205.5+RND(1)); : GOTO 10
```



```
10 PRINT CHR$(205.5+RND(1)); : GOTO 10
```

NICK MONTFORT, PATSY BAUDOIN,
JOHN BELL, IAN BOGOST, JEREMY DOUGLASS,
MARK C. MARINO, MICHAEL MATEAS,
CASEY REAS, MARK SAMPLE, NOAH VAWTER

10print.org

Cloudflare lobby [photo by @mahtin]

Is this sequence of digits random?

94370277053921717629317675238467481846766940513200056812

3.141592653589793238462643383279502884197169399375105820
974944592307816406286208998628034825342117067982148086
5132823066470938446095505822317253594081284811174502841
0270193852110555964462294895493038196442881097566593344
6128475648233786783165271201909145648566923460348610454
32664821339360726024914127372458700660631558817488152092
096282925409171536436789259036001133053054882046652138
4146951941511609433057270365759591953092186117381932611793
105118548074462379962749567351885752724891227938183011949
129833673362440656643086021394946395224737190702179860
94370277053921717629317675238467481846766940513200056812
71452635608277857713427577896091736371787214684409012249
53430146549585371050792279689258923542019956112129021960
86403441815981362977477130996051870721134999999837297804
995105973173281609631859502445945534690830264252230825
33446850352619311881710100031378387528865875332083814206 1
71776691473035982534904287554687311595628638823537875937
519577818577805321712268066130019278766111959092164201989
3809525720106548586327886593615338182796823030195203530
185296899577362259941389124972177528347913151557485724245
4150695950829533116861727855889075098381754637464939319

Most random number generators are like the digits of π: completely deterministic, but *hard to predict.*

These are called **Pseudorandom Number Generators** (PRNGs).

```
void randomSeed( int seed ) { ... }
```

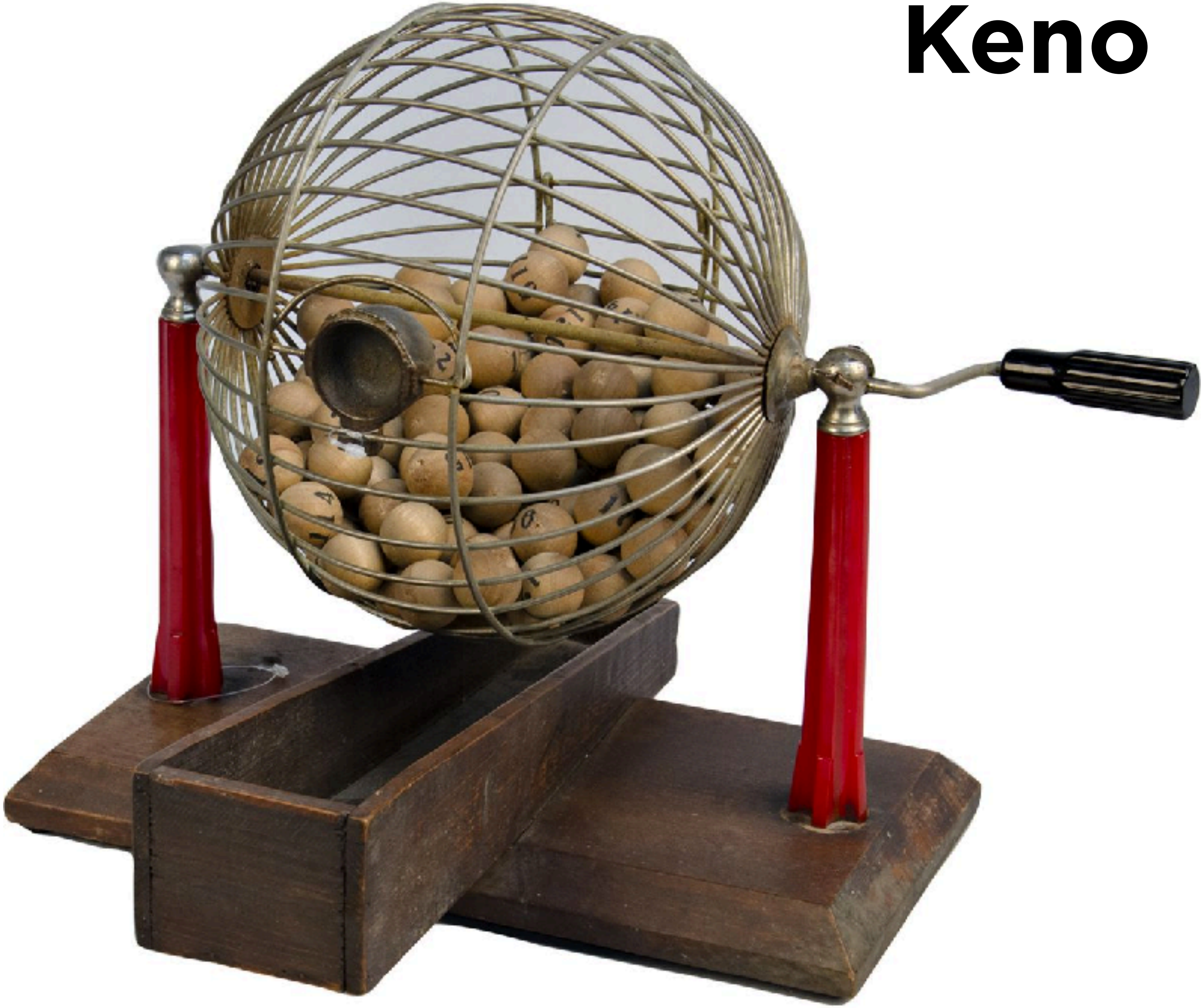
Reset the internal state of Processing’s PRNG based on the passed-in `seed`. A given seed will always produce the same sequence of answers to a given sequence of calls to `random()`.

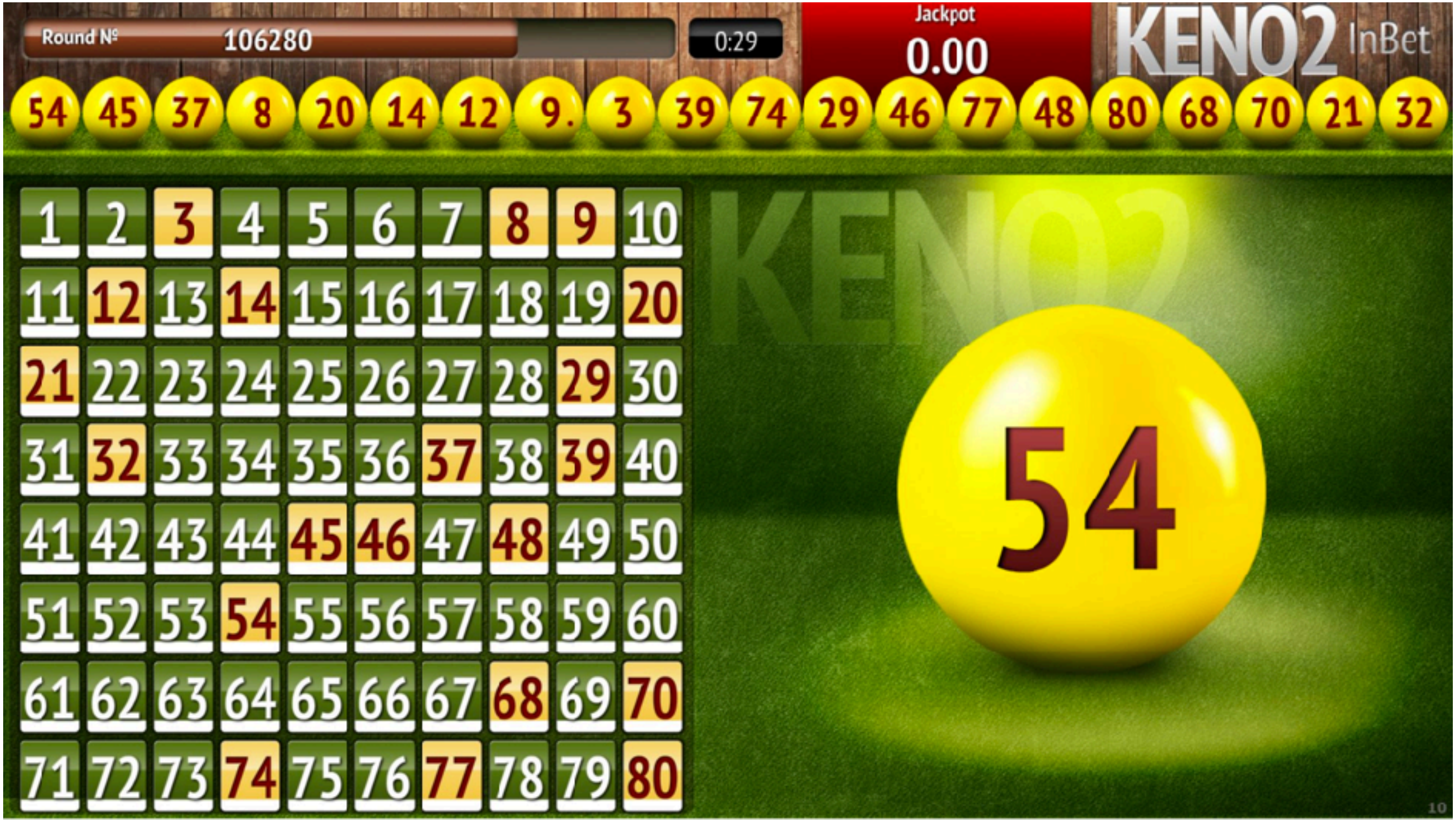
Pseudorandom number generators are a double-edged sword.

The good: we can always “replay” a sequence of pseudorandom numbers.

The bad: pseudorandom numbers ***are not actually random.***

Keno





Round № 106280

0:29

Jackpot
0.00

KENO2 InBet

54 45 37 8 20 14 12 9. 3 39 74 29 46 77 48 80 68 70 21 32

1	2	**3**	4	5	6	7	**8**	**9**	10
11	**12**	13	**14**	15	16	17	18	19	**20**
21	22	23	24	25	26	27	28	**29**	30
31	**32**	33	34	35	36	**37**	38	**39**	40
41	42	43	44	**45**	**46**	47	**48**	49	50
51	52	53	**54**	55	56	57	58	59	60
61	62	63	64	65	66	67	**68**	69	**70**
71	72	73	**74**	75	76	**77**	78	79	**80**

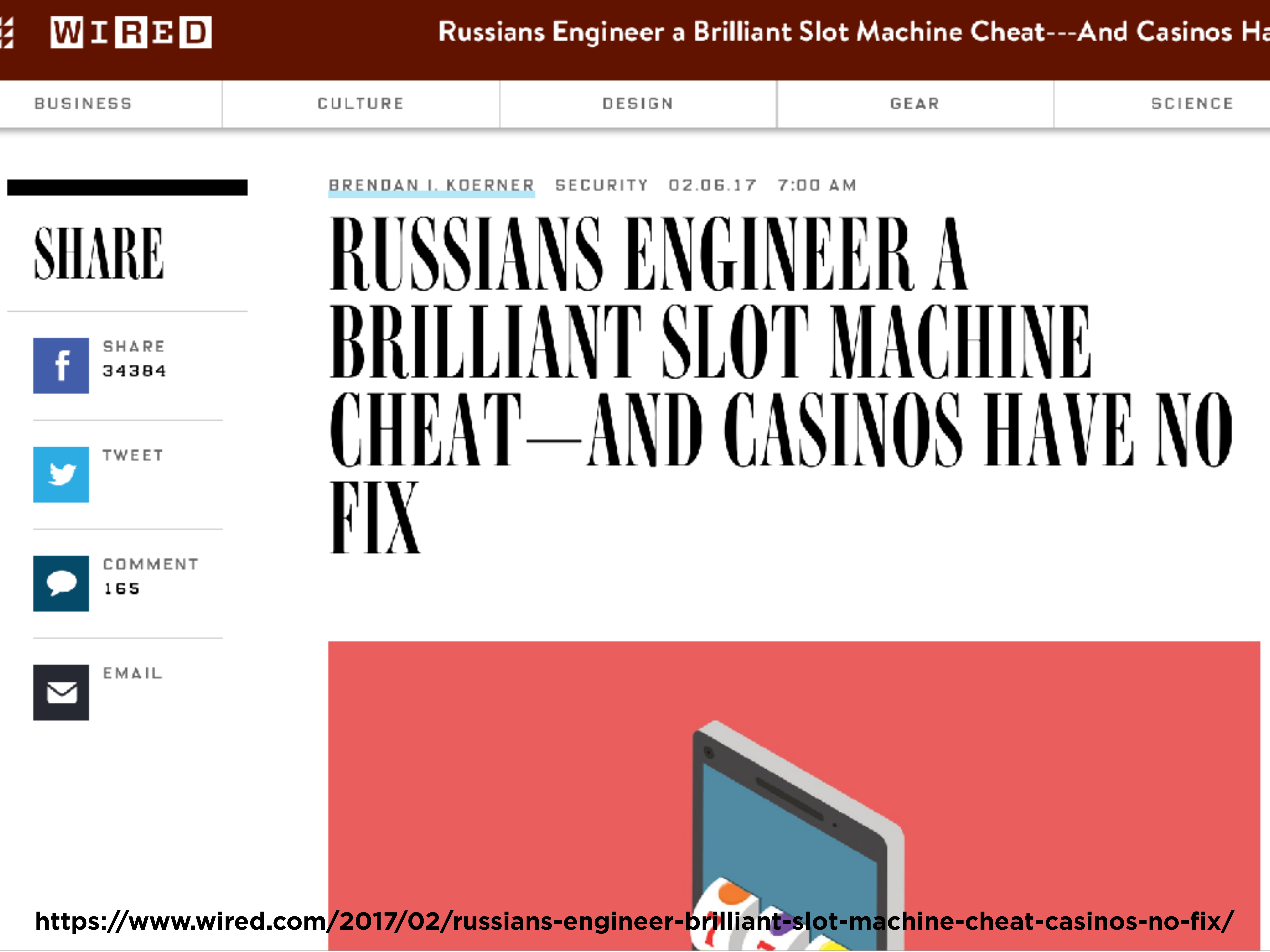
In April 1994, Daniel Corriveau won $620,000 CAD playing keno. He picked 19 of the 20 winning numbers three times in a row. Corriveau claims he used a computer to discern a pattern in the sequence of numbers, based on chaos theory. However, it was later found that the sequence was easy to predict because the casino was using an inadequate electronic pseudorandom number generator. In fact, the keno machine was reset every morning with the same seed number, resulting in the same sequence of numbers being generated. Corriveau received his winnings after investigators cleared him of any wrongdoing.

en.wikipedia.org/wiki/Montreal_Casino

Partial solutions:

1. Set an initial seed based on the current time.

3. Generate random numbers continuously, not just when needed.



Russians Engineer a Brilliant Slot Machine Cheat---And Casinos Ha

BRENDAN I. KOERNER SECURITY 02.06.17 7:00 AM

RUSSIANS ENGINEER A BRILLIANT SLOT MACHINE CHEAT—AND CASINOS HAVE NO FIX

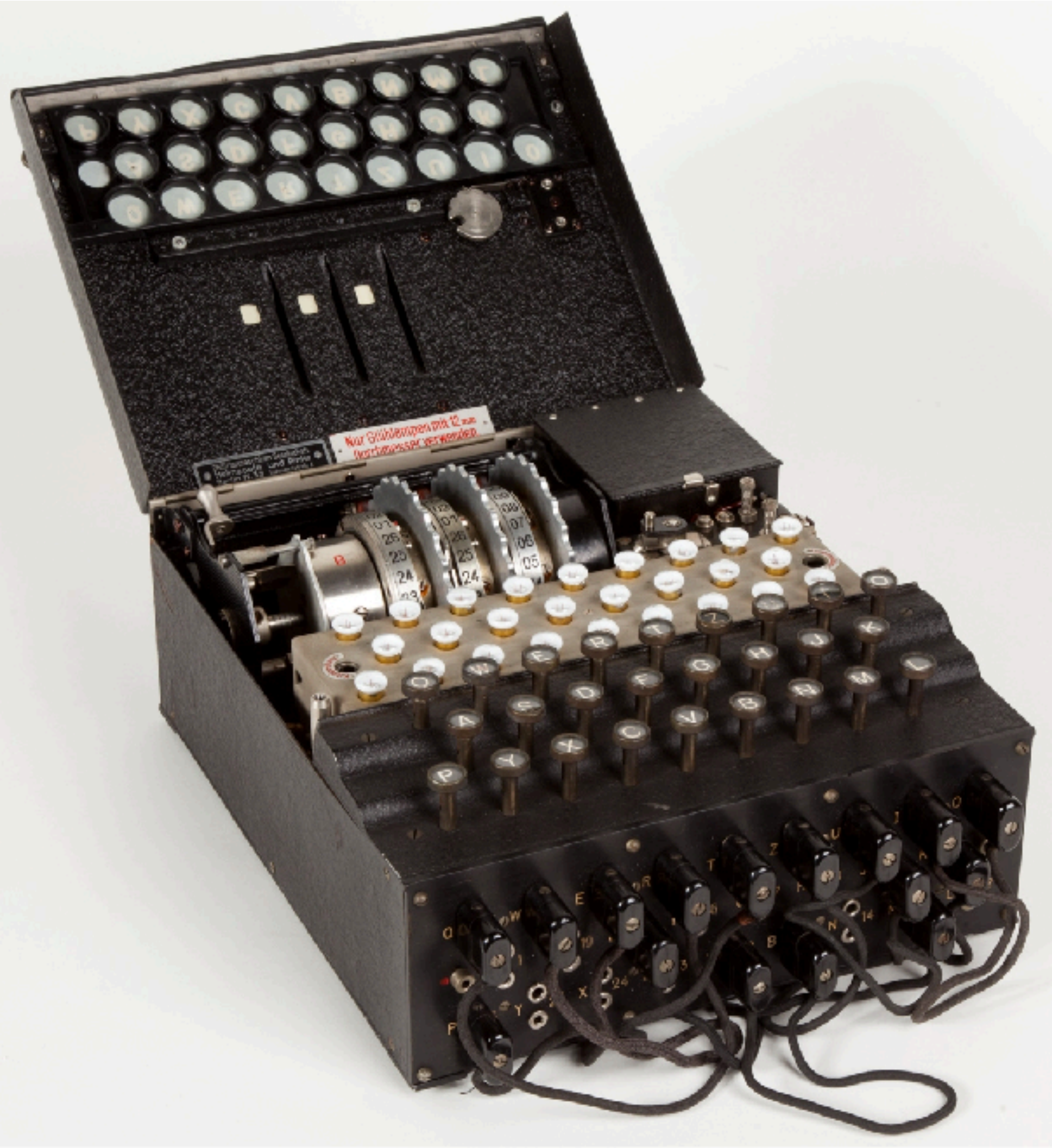
https://www.wired.com/2017/02/russians-engineer-brilliant-slot-machine-cheat-casinos-no-fix/

Ronald Dale Harris is a computer programmer who worked for the Nevada Gaming Control Board in the early 1990s and was responsible for finding flaws and gaffes in software that runs computerized casino games. Harris took advantage of his expertise, reputation and access to source code to illegally modify certain slot machines to pay out large sums of money when a specific sequence and number of coins were inserted.

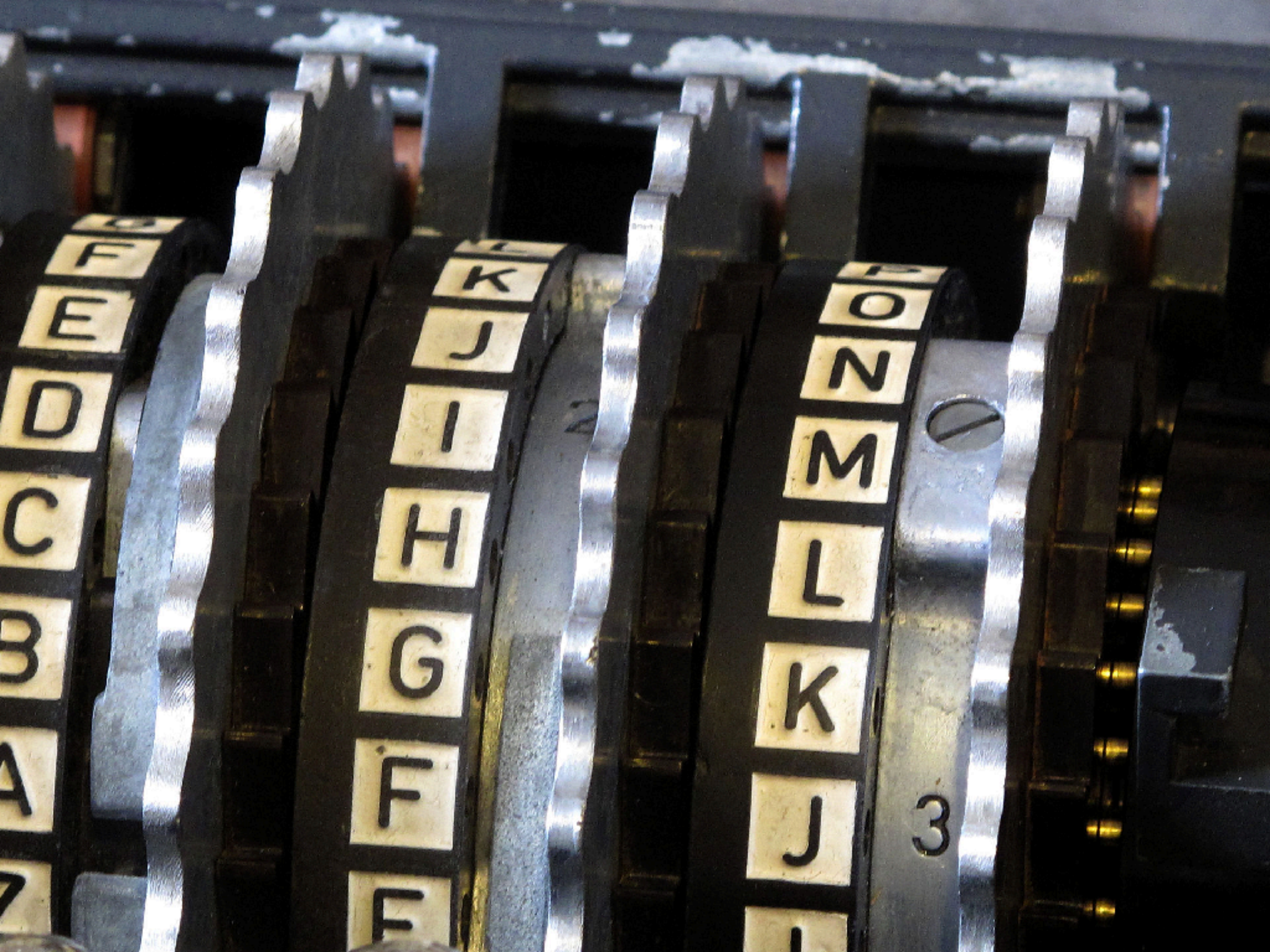
https://en.wikipedia.org/wiki/Ronald_Dale_Harris

Modern cryptographic protocols often require frequent generation of random quantities. Cryptographic attacks that subvert, or exploit weaknesses in, this process are known as **random number generator attacks**.

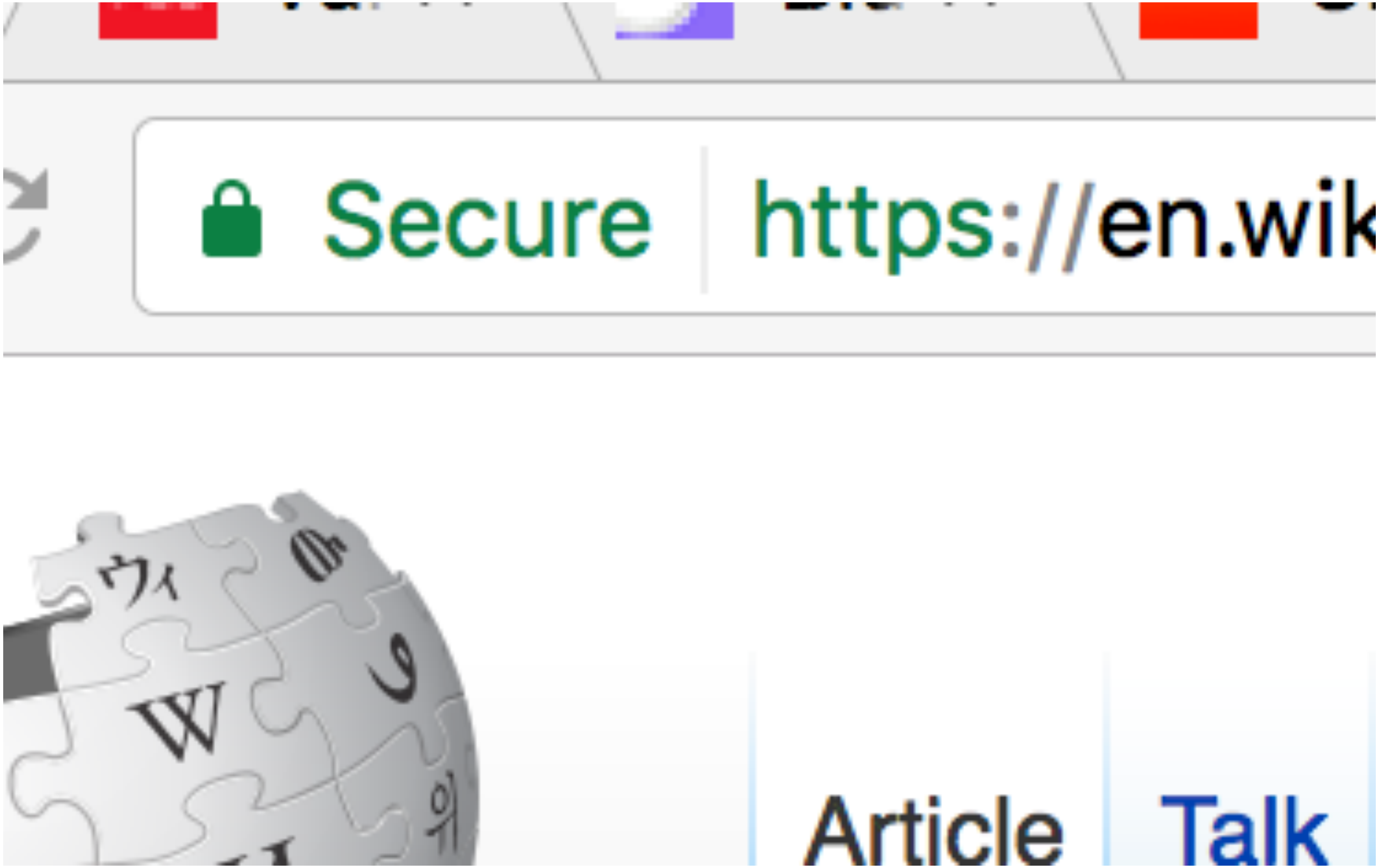
https://en.wikipedia.org/wiki/Random_number_generator_attack



Enigma Machine


F E D C B A
K J I H G F
O N M L K J
2
3

Transport Layer Security (TLS/SSL)



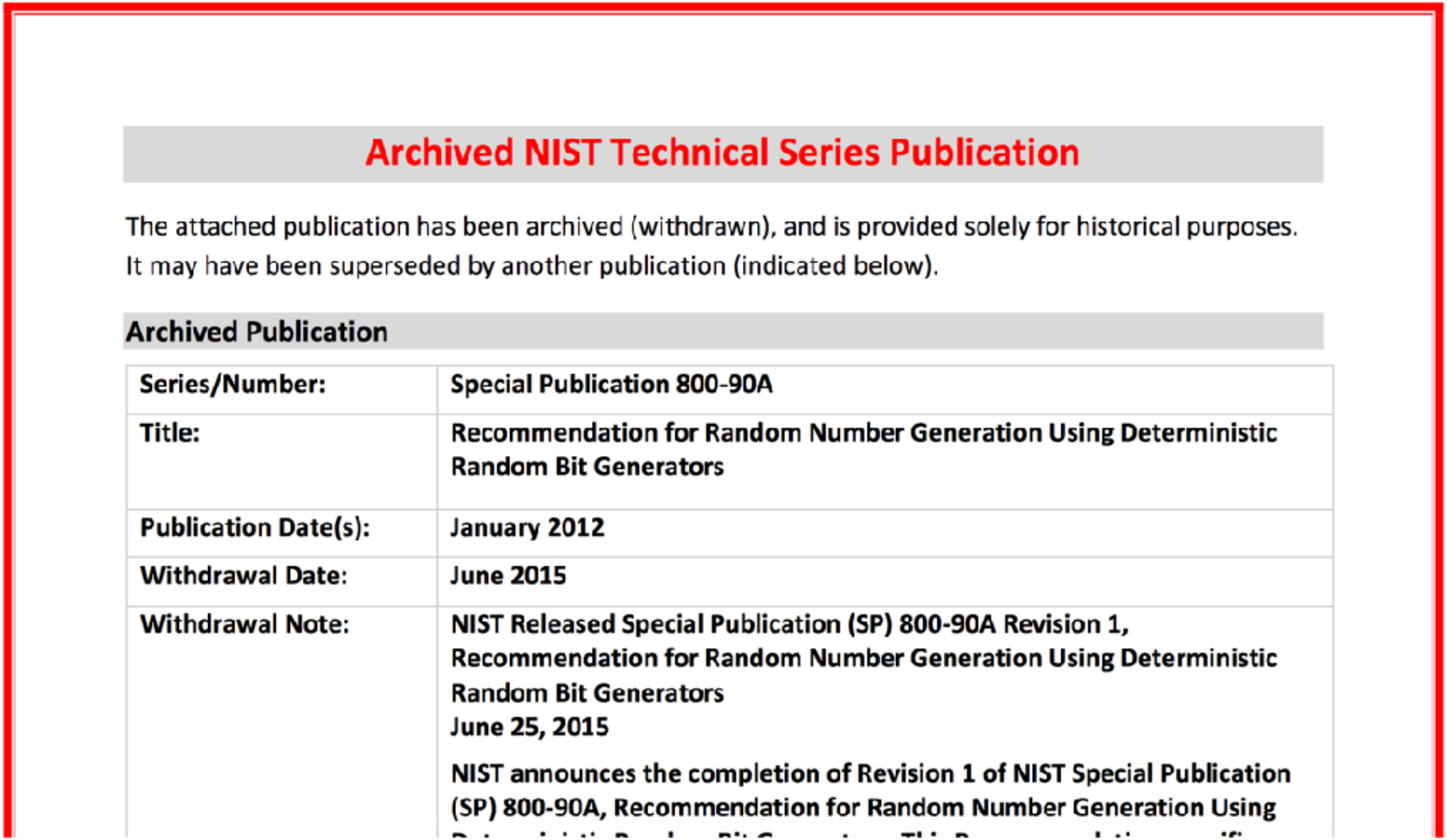
Early versions of Netscape's Secure Socket Layer (SSL) encryption protocol used pseudo-random quantities derived from a PRNG seeded with three variable values: the time of day, the process ID, and the parent process ID.

Early versions of Netscape's Secure Socket Layer (SSL) encryption protocol used pseudo-random quantities derived from a PRNG seeded with three variable values: the time of day, the process ID, and the parent process ID.

…The problem in the running code was discovered in 1995 by Ian Goldberg and David Wagner…

Early versions of Netscape's Secure Socket Layer (SSL) encryption protocol used pseudo-random quantities derived from a PRNG seeded with three variable values: the time of day, the process ID, and the parent process ID.

...The problem in the running code was discovered in 1995 by Ian Goldberg and David Wagner...

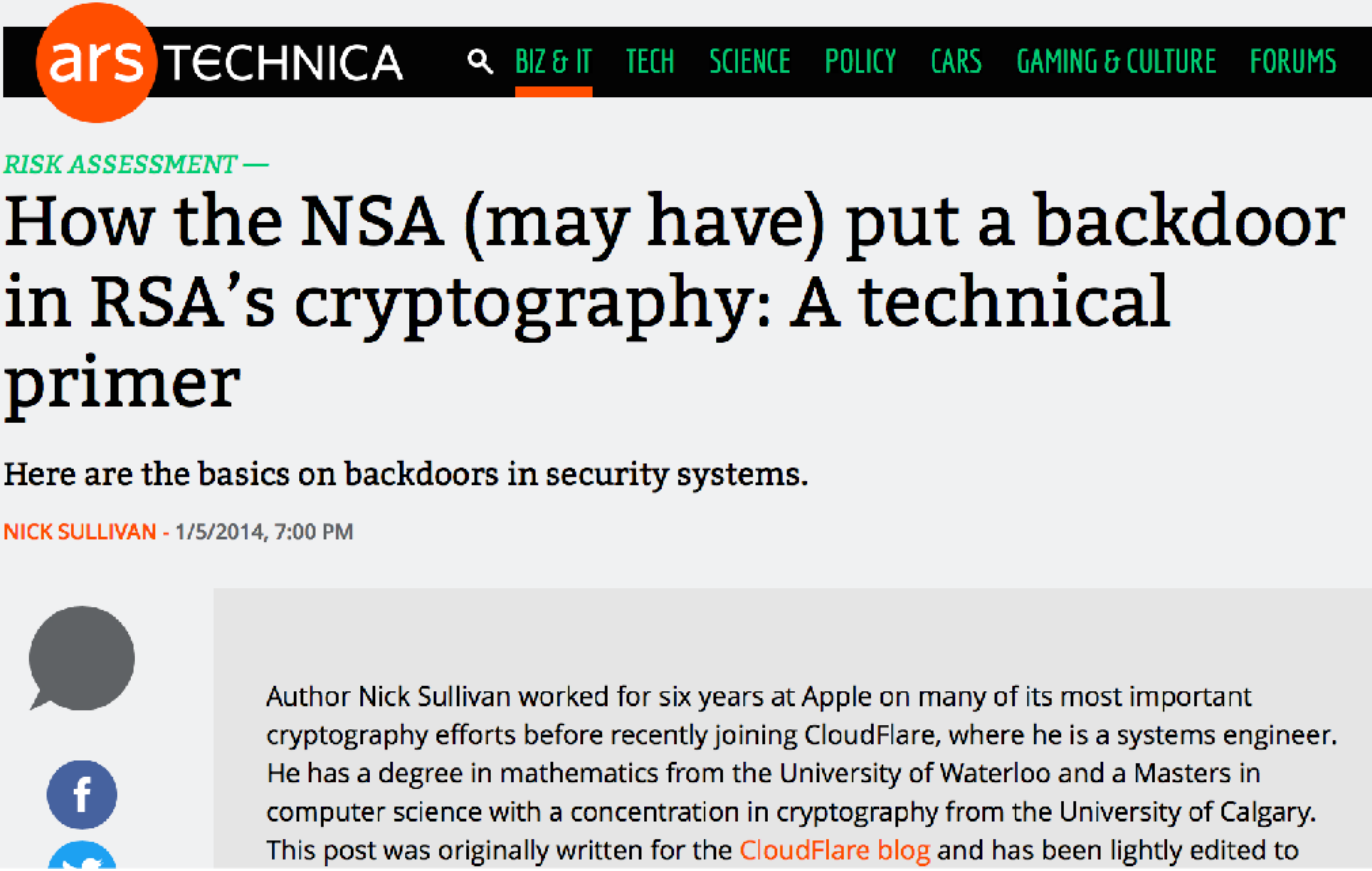


Archived NIST Technical Series Publication

The attached publication has been archived (withdrawn), and is provided solely for historical purposes. It may have been superseded by another publication (indicated below).

Archived Publication

Series/Number:	**Special Publication 800-90A**
Title:	**Recommendation for Random Number Generation Using Deterministic Random Bit Generators**
Publication Date(s):	**January 2012**
Withdrawal Date:	**June 2015**
Withdrawal Note:	**NIST Released Special Publication (SP) 800-90A Revision 1, Recommendation for Random Number Generation Using Deterministic Random Bit Generators** **June 25, 2015** **NIST announces the completion of Revision 1 of NIST Special Publication (SP) 800-90A, Recommendation for Random Number Generation Using Deterministic Random Bit Generators. This Recommendation specifi**



RISK ASSESSMENT —

How the NSA (may have) put a backdoor in RSA's cryptography: A technical primer

Here are the basics on backdoors in security systems.

NICK SULLIVAN - 1/5/2014, 7:00 PM

Author Nick Sullivan worked for six years at Apple on many of its most important cryptography efforts before recently joining CloudFlare, where he is a systems engineer. He has a degree in mathematics from the University of Waterloo and a Masters in computer science with a concentration in cryptography from the University of Calgary. This post was originally written for the CloudFlare blog and has been lightly edited to

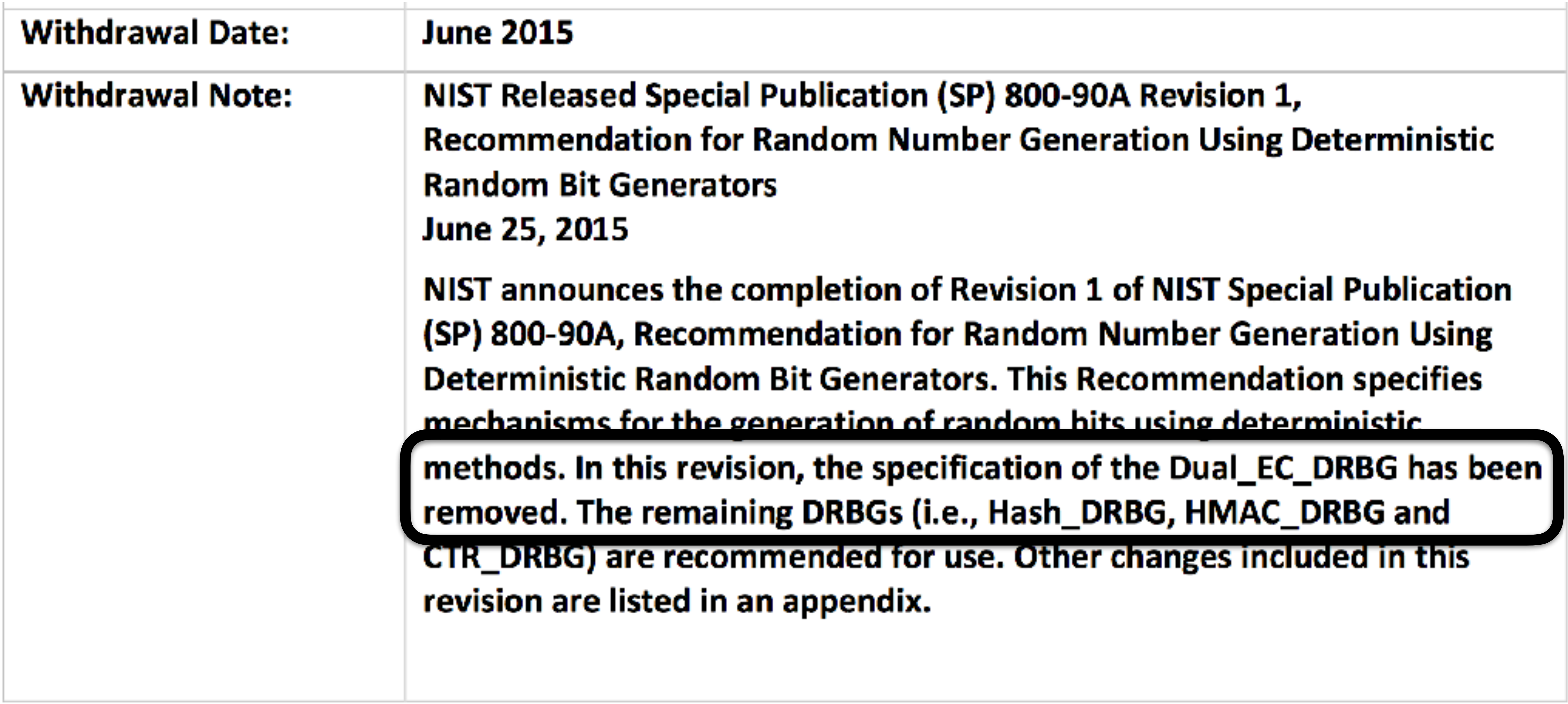
https://arstechnica.com/security/2014/01/how-the-nsa-may-have-put-a-backdoor-in-rsas-cryptography-a-technical-primer/

The U.S. National Institute of Standards and Technology has published a collection of “deterministic random bit generators” it recommends as NIST Special Publication 800-90. One of the generators, Dual_EC_DRBG, was favoured by the National Security Agency.

The U.S. National Institute of Standards and Technology has published a collection of “deterministic random bit generators” it recommends as NIST Special Publication 800-90. One of the generators, Dual_EC_DRBG, was favoured by the National Security Agency.

In 2013, Reuters reported that documents released by Edward Snowden indicated that the NSA had paid RSA Security $10 million to make Dual_EC_DRBG the default in their encryption software, and raised further concerns that the algorithm might contain a backdoor for the NSA.

Withdrawal Date:	**June 2015**
Withdrawal Note:	**NIST Released Special Publication (SP) 800-90A Revision 1, Recommendation for Random Number Generation Using Deterministic Random Bit Generators June 25, 2015 NIST announces the completion of Revision 1 of NIST Special Publication (SP) 800-90A, Recommendation for Random Number Generation Using Deterministic Random Bit Generators. This Recommendation specifies mechanisms for the generation of random bits using deterministic methods. In this revision, the specification of the Dual_EC_DRBG has been removed. The remaining DRBGs (i.e., Hash_DRBG, HMAC_DRBG and CTR_DRBG) are recommended for use. Other changes included in this revision are listed in an appendix.**



Withdrawal Date:	**June 2015**
Withdrawal Note:	**NIST Released Special Publication (SP) 800-90A Revision 1, Recommendation for Random Number Generation Using Deterministic Random Bit Generators** **June 25, 2015** **NIST announces the completion of Revision 1 of NIST Special Publication (SP) 800-90A, Recommendation for Random Number Generation Using Deterministic Random Bit Generators. This Recommendation specifies mechanisms for the generation of random bits using deterministic methods. In this revision, the specification of the Dual_EC_DRBG has been removed. The remaining DRBGs (i.e., Hash_DRBG, HMAC_DRBG and CTR_DRBG) are recommended for use. Other changes included in this revision are listed in an appendix.**

Goals

- Understand how to use random() to generate unpredictable behaviour in Processing sketches.
- Understand how to use randomSeed() to control the generation of pseudorandom numbers.
- Understand the difference between random numbers and pseudorandom numbers.