Science | DOI:10.1145/1924421.1924427

Gary Anthes

The Quest for Randomness

It's not easy to generate a string of numbers that lack any pattern or rule, or even to define exactly what randomness means.

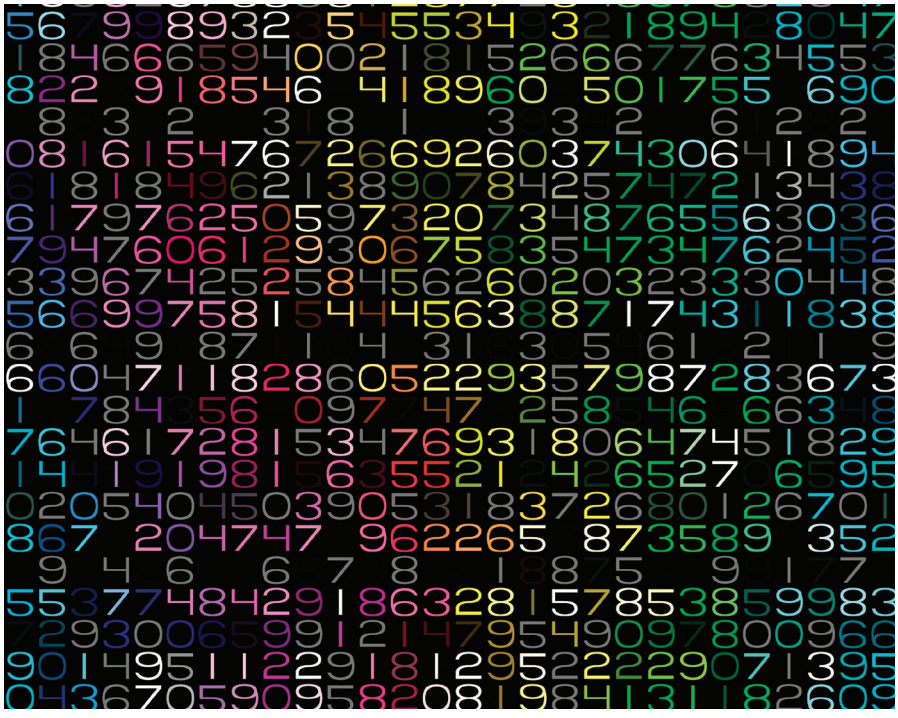

ILLUSTRATION BY WEKNOW/SHUTTERSTOCK

GENERATING RANDOM NUMBERS might seem pretty simple. After all, so many things in our everyday lives occur without pattern or repetition—coin tosses, for example, or the distribution of raindrops on your car windshield. In any case, the whole notion of striving for randomness might seem a bit alien to a computer scientist, who is trained to make processes predictable and repeatable.

Cryptographers use random number generators (RNGs) to produce unpredictable keys for coded messages, digital signatures, challenge-response systems, and other technologies at the heart of information security. And researchers in fields such as biology, economics, and physics use them to simulate the real world via Monte Carlo models. Human life and great sums of money may depend on the robustness of these systems, yet it is often difficult to prove that they work correctly—and will continue to do so.

Part of the problem is one of definition. One might say, for example, that a random bit generator (as RNGs are often called) must over time produce half 0s and half 1s, or that it must generate the sequence "00" one-quarter of the time. But, points out Silvio Micali, a computer scientist and cryptography expert at Massachusetts Institute of Technology (MIT), the sequence 0, 1, 2, ... 9 (written in binary) would pass those tests but would hardly be considered "random."

Definitions matter, Micali says. "If you define [randomness] correctly, then you can get it. But if you are vague or unclear, you can't possibly get it." In 1984, Micali and Manuel Blum, a professor of computer science at Carnegie Mellon University, published an influential paper, "How to Generate

Pseudorandomness, and How to Get It

"You can't start with a blank sheet of paper and generate 100 million random numbers," says Silvio Micali, a computer scientist at Massachusetts Institute of Technology. "That is very hard."

Instead, he says, one should begin with a relatively small list—say, 100 bits—that are truly random, most likely generated by some physical process such as quantum mechanics. He goes on to define pseudorandom number generators as software that "expands" the randomness in this initial small list, or seed.

"Now I want to generate 100 million random bits of very high quality," he says. "But how do I define 'high quality'?" He and Manuel Blum, a professor of computer science at Carnegie Mellon University, proposed a definition in which anyone shown the first *n* bits generated, but not the seed, could not guess what the next bit would be substantially more than 50% of the time. "Substantially more" might be defined as 51%, 50.1%, 50.01%, or some other threshold, depending on one's security requirements.

The Micali-Blum definition also includes the notion of computational resources. "If all the computers in the world must work for one million years to guess the next bit, then it is high quality," says Micali, who believes the ideal seed contains 300 bits. Guessing the 300-bit seed that generated any given sequence by trial-and-error would mean testing more possibilities than there are elementary particles in the universe.

Given the seed list of truly random bits, Micali and Blum prescribed the use of one-way functions to produce the desired sequence of random bits. In cryptography, a one-way function is easy to compute in one direction—for example, determining the product of two large prime numbers—but computationally very difficult to reverse—factoring that product back into the two primes. That is the method used in public-key cryptography, for example.

Similarly, the Blum-Micali generator of pseudorandom bits employs the asymmetry between computing discrete logarithms, which is difficult, and the inverse, discrete exponentiation, which is easy.

Micali refuses to comment on any of the existing ways that so-called "true" random numbers are generated. "I prefer to be agnostic about seed generation," he says, "because I find it conceptually difficult to convince myself of their randomness."

—Gary Anthes

Cryptographically Strong Sequences of Pseudorandom Bits," that laid out just such a definitional framework (see the "Pseudorandomness, and How to Get It" sidebar above).

Another difficulty lies in testing a RNG. Traditionally, a test checks to see if the system in question reliably produces output known in advance to be correct, but when is a stream of random bits correct? The U.S. National Institute of Standards and Technology (NIST) has developed "known-answer tests" for software-based, or deterministic, RNGs, because any given algorithm is expected to produce the same answer every time, given the same input. But convincing tests of the output of nondeterministic RNGs, such as a noise source in hardware, are not so easy to construct.

A third problem, related to testing, is that some RNGs initially work correctly but gradually and silently fail over time, introducing biases that corrupt randomness.

The deterministic nature of software, which produces pseudorandom numbers, has led researchers to look for ways to produce "true" randomness, and that always involves hardware or a physical process not governed by rules. Indeed, the Russian mathematician Andrei Kolmogorov defined randomness in a way that rules out software as a way to produce it. According to Kolmogorov, a string of bits is truly random if the shortest description of the string is as long as the string itself. Researchers have based nondeterministic RNGs on disk head seek times, thermal noise, atomic scale quantum effects, and other phenomena that are statistically shown to be fundamentally random.

A cryptography system in finance or national security might need to generate millions of random bits per second.

Theoretically perfect hardware approaches can suffer from efficiency problems, says MIT's Micali. For example, one can imagine looking at the Brownian movement of a particle suspended in a liquid and counting it as a 0 if it is on the left and a 1 if it is on the right. "To get the first bit is easy," he says, "but to get 1,000 is a little bit hard." A cryptography system in banking or national security might need to generate millions of random bits per second.

A Matter of Timing

A more subtle problem is one of timing. Suppose one samples the randomly fluctuating noise from a diode and records a 1 if it is below a certain threshold and a 0 above. If you sample too often, you are likely to get biased strings like 000011110000 because the noise level doesn't change fast enough. For this and a variety of other reasons, hardware sources are not immune to biases that affect the randomness of their output.

But now Intel Corp. claims to have found a way around the drawbacks of nondeterministic, hardware-based RNGs. It recently announced that it had created a working prototype of a digital circuit, which could be resident within the central processing unit, that harvests randomness, or "entropy," from thermal noise in the chip. "Ours is the first truly, fully digital RNG," claims Ram Krishnamurthy, a senior principal engineer at Intel's Circuits Research Lab.

He says conceptually similar devices, known as true random number generators, have been built from analog circuits that are subject to disruption from the process and noise variations in the central processing unit, so they have to be isolated off-chip and connected by a bus that is vulnerable to snooping. They also employ large capacitors that make manufacturing and scaling difficult, Krishnamurthy says.

Intel's circuit is implemented in a 45nm complementary metal oxide semiconductor and can generate 2.4 billion truly random bits per second—200 times faster than the best available analog equivalent—while drawing just 7 milliwatts of power, according to Krishnamurthy. And the technology is highly scalable, he says, so that multiple copies of the digital circuit could be coupled in parallel arrays. The technology could be scaled up in this way to directly provide the random numbers needed by large systems, or it could be scaled down to low voltages so as to just provide high-entropy seeds for a software-based RNG, Krishnamurthy says. In the latter mode, it would operate at 10 megabits per second and draw just 10 microwatts of power.

Krishnamurthy acknowledges that the circuit's output is subject to process fluctuations—caused by transistor, power supply, and temperature variations—that could introduce bias in its output. But Intel has developed a self-calibrating feedback loop that compensates for those variations. The resulting design operates at a level of entropy of 99.9965% and has passed the NIST tests for randomness, Krishnamurthy says.

But more work on these tests is needed, says Elaine Barker, a mathematician in NIST's Computer Security Division. "The thing we have been really struggling with is how to test the entropy sources, the noise sources," she says. "We are kind of feeling our way along. How do you do general testing for technology when you don't know what will come along? We are not really sure how good these tests are."

Some random number generators initially work correctly but silently fail over time, introducing biases that corrupt randomness.

Entropy sources may not produce output that is 100% random, and different test samples from a single source may have different degrees of randomness. "We want to know how much entropy is in 100 bits—100, 50, or two?" Barker says. "And does it continue that way?"

Indeed, generating random numbers today clearly lags what is theoretically possible, Micali says. "We are still in the mode of using library functions and strange things that nobody can prove anything about," he says.

Further Reading

Barker, E. and Kelsey, J.
Recommendation for random number generation using deterministic random bit generators (revised), *NIST Special Publication 800-90*, U.S. National Institute of Standards and Technology, March 2007.

Blum, M. and Micali, S.
How to generate cryptographically strong sequences of pseudorandom bits, *SIAM Journal on Computing 13*, 4, November 1984.

Menezes, A., van Oorschot, P., and Vanstone, S.
Pseudorandom bits and sequences, *Handbook of Applied Cryptography*, CRC Press, Boca Raton, FL, 1996.

Rukhin, A, Soto, J., Nechvatal, J., Smid, M., Barker, E., Leigh, S., Levenson, M., Vangel, M., Banks, D., Heckert, A., Dray, J., and Vo, S.
A statistical test suite for random and pseudorandom number generators for cryptographic applications, *NIST Special Publication 800-22*, U.S. National Institute of Standards and Technology, April 2010.

Srinivasan, S., Mathew, S., Ramanarayanan, R., Sheikh, F., Anders, M., Kaul, H., Erraguntla, V., Krishnamurthy, R., and Taylor, G.
2.4GHz 7mW all-digital PVT-variation tolerant true random number generator in 45nm CMOS, 2010 IEEE Symposium on VLSI Circuits, Honolulu, HI, June 16–18, 2010.

Gary Anthes is a technology writer and editor based in Arlington, VA.

Society

Predictive Modeling as Preventive Medicine

If an ounce of prevention is worth a pound of cure, then how much prevention would it take to put a dent in the U.S.'s projected $2.8 trillion annual health-care tab?

How about $3 million worth? That's the amount the Heritage Provider Network is offering as prize money in a new contest for developers to create algorithms aimed at identifying those patients most likely to require hospitalization in the coming year. (Contest details are available at http://www.heritagehealthprize.com/.)

Previous studies have suggested that early treatment of at-risk patients can dramatically reduce health care expenditures. For example, a well-regarded study in Camden, NJ, demonstrated that hospitals could slash costs by more than 50% for their most frequently hospitalized patients by targeting them with proactive health care before they incurred expensive trips to the emergency room.

In hopes of replicating those savings on a larger scale, the contest organizers will furnish developers with a set of anonymized medical records for 100,000 patients from 2008. Using this data set, developers will try to develop a predictive algorithm to pinpoint those patients most likely to have been hospitalized the following year.

For example, an effective algorithm might identify patients who have already been diagnosed with diabetes, high cholesterol, hypertension, and premature menopause—and correctly predict the 90% likelihood that such a patient would wind up in the hospital within the year.

"The Heritage Health Prize is a high profile way to harness the power of predictive modeling and using it solves one of America's biggest challenges," says Jeremy Howard, chief data scientist for Kaggle, the company managing the contest.

If successful, this effort could yield a powerful computational tool to help rein in spiraling health-care costs, potentially saving billions of dollars—or at least a few pounds' worth of cure.

—Alex Wright