Random Numbers Simplified

What is a random number?
Random Numbers are numbers you can't guess or predict by any algorithm. A measurement of something naturally random is required to produce a random number. Random numbers are also described as true, truly, hardware, non-deterministic or quantum.

What are some things that are naturally random?
Vibrations of electrons in a resistor (thermal noise), cascades of electrons passing through a semiconductor junction (breakdown or avalanche noise), bunching of electrons passing through a diode junction (shot noise), bunching of photons in a light beam (photon shot noise), charges moving across a thin insulator (quantum tunneling), photon polarization (polarization analyzer or polarization beam splitter), photon passing through a beam splitter and timing of nuclear decay.

Some people consider certain sources, such as the vibration of electrons in a resistor (thermal, Johnson or Johnson-Nyquist noise), to be merely chaotic rather than inherently random. In this context chaotic means the process is extremely complex, but given sufficient information on initial conditions and computational power, future bits could be predicted. This assertion is not correct because such measurements and the necessary knowledge of position and momentum of the electrons is not just astronomically complex, it is theoretically precluded by Heisenberg's uncertainty principle.

What is a pseudo-random number?
Pseudo-random numbers, meaning artificial or random in appearance only, are produced by a computer program to simulate unpredictability. Pseudo-random numbers are often used in place of true random numbers because they are easier to produce and they can be made very hard to predict. Every pseudo-random generator has a finite period, meaning its output sequence will begin to repeat after all its potential states have been produced.

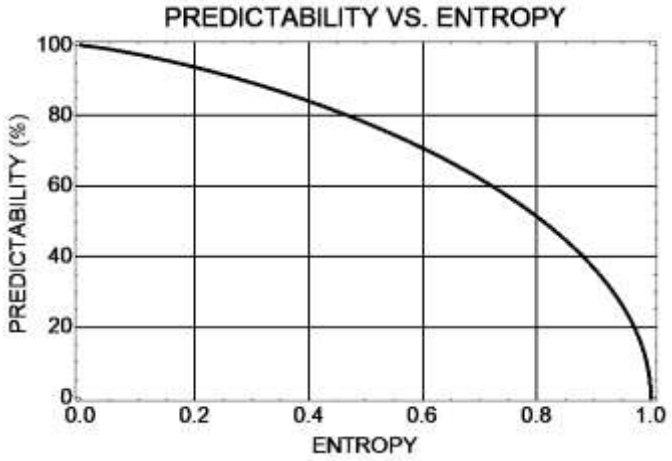
High-quality pseudo-random number generators have been notoriously difficult to design and there were a number of monumental flops in the use of what turned out to be very bad generators. One of the best modern generators is the Mersenne Twister, although it must be seeded very carefully including a runoff of numbers at the beginning before its output is reliable. Even then it will begin to fail very long-term statistical tests unless its output is further appropriately whitened.

What is Entropy?
Entropy exists in a physically unpredictable property or process measured to produce random numbers. On a scale from 0 to 1, it indicates how unpredictable the generated random number is. A True random number has an entropy of 1.0, while a pseudo-random number has real entropy of 0.0. Entropy has different definitions in other technical fields, but we use the definition for information entropy derived by Claude Shannon. For binary bits, the equation is $H = –(p(1)Log_2p(1) + p(0)Log_2p(0))$ where $p(1)$ and $p(0)$ are the probabilities of a one and a zero occurring. This equation may be modified by replacing the probabilities $p(1)$ and $p(0)$ with a single variable, predictability or P, and the entropy equation becomes, $H_p = –(PLog_2P + (1–$

$P)\text{Log}_2(1-P))$, where $0.5 \leq P \leq 1.0$. Predictability can also be calculated numerically from the entropy: $P = h^{-1}$, where h^{-1} is the mathematical inverse of the H_p equation.

Why is Entropy important?

Random numbers with entropy less than 1.0 can be predictable to some degree. Information entropy is the type most often used when discussing random numbers. It is usually expressed in units of bits – the same ones or zeros that are the basic alphabet of binary computers. Several fundamentals of information theory were described by Claude Shannon in 1948. His equations can be modified to calculate the predictability of binary bits as a function of the average entropy content of each bit. The following plot shows the dramatic effect of having "random" bits with less than 1.0 bit of entropy each.



The plot shows that pseudo-random bits, containing zero bits of entropy, are theoretically 100% predictable, while true random bits, with 1.0 bit of entropy in each bit, are completely unpredictable. When otherwise random numbers are biased or defective in other ways, their entropy is less than 1.0 and their predictability rapidly increases. For example, if a sequence of random bits is biased so the number of ones is 55% of all bits, it has a predictability of 10% while the entropy is 0.99277 – still very close to 1.0. This decrease in entropy of only 0.72% allows an average of one in every 10 bits to be correctly predicted. Some messages of moderate length encoded using the biased numbers might be decoded by an attacker.

Typically, designers of true random number generators cover up the statistical defects – or lack of entropy – in their raw "random" output by some type of post-processing. This process is also called whitening in reference to the result of producing a "white" or uniformly flat frequency spectrum. A common example of this is to combine each bit in the raw sequence with a bit from the output of a pseudo-random generator. This combination is usually done by using the XOr (Exclusive-Or) function. The result is a sequence of numbers that appears to be statistically random, that is, it will pass statistical tests in the same way that a good pseudo-random generator will.

The fallacy in this approach is that the amount of entropy in each number is not changed by the combination with the pseudo-random bits, which contain no entropy at all. The theoretical predictability remains exactly the same and only the complexity of finding the patterns is

increased. The solution to this problem for an attacker is to use a more powerful computer and more sophisticated search algorithms.

Ultimately, statistically defective random numbers can be combined or "stirred" using a cryptographic hash function, like SHA-1. Again, this process produces sequences of numbers that appear statistically random. While the complexity of this transformation is increased tremendously, the SHA-1 algorithm is deterministic, meaning it cannot add any entropy to the numbers fed into it. While the entropy of the output will be increased in a predictable way if fewer bits are taken out of the hash function than are fed in, the ability to find patterns and potentially break codes using these cryptographically whitened random numbers is a matter of having enough computational power and adequately sophisticated algorithms.

Sufficiently powerful computers may not yet exist, but the recent rapid development of quantum computers will likely close that gap so that random numbers that are not effectively perfect, that is, having entropy indistinguishable from 1.0 bits per bit, will be vulnerable to some degree of predictability and potential attack. It should be noted that security flaws were discovered in SHA-1 causing it to no longer be approved for most cryptographic functions after 2010 – an example of an algorithm that was considered "cryptographically strong" but was later found to be inadequate as technology advanced.

Why do we need random numbers?
Random numbers are used to make codes to transfer private messages or money without danger of eavesdropping or theft. They are also used to make "random" selections, such as lottery numbers and all sorts of drawings. In addition, random numbers are used for a wide variety of other applications including simulation (such as Monte Carlo simulation), random computation, statistical testing, neural networks and even in music and art.

How do we tell how good the random numbers are?
The quality of random numbers is determined both from understanding the theory of how they are made and by applying statistical testing algorithms. Statistical tests always look for patterns in a string of numbers. The more pronounced any patterns are found to be, the more predictable (less random) are the numbers. Statistical tests must be very carefully selected or designed to reveal more subtle patterns in longer sequences of numbers. It used to be enough to say a generator passed Marsaglia's Diehard test, for example, to be considered a good generator. Now tests must be run on longer sequences of numbers and collections of tests must be combined in meta tests to see if there are patterns in collections of results.

Tests in test suites should be substantially independent or they may give a biased picture of test results. As an example, the statistical test suite developed by NIST to test random sequences includes two tests (Approximate Entropy and Serial Test) that produce virtually identical results if the index of one is incremented relative to the other. These tests should not be used together in a test suite. In addition, some of the tests provided by the NIST test suite fail regardless of the statistical quality of the numbers tested. This occurs due to using inexact or incorrect statistical distributions that diverge when the sequence of numbers is even moderately long.