Cryptographic Pseudo-random Numbers in Simulation

Nick Maclaren

University of Cambridge Computer Laboratory
Pembroke Street, Cambridge CB2 3QG.

A fruitful source of confusion on the Internet is that both cryptologists and statisticians use pseudo-random numbers, but their objectives and constraints are subtly different. This paper will describe some of the requirements for a good generator for statistical simulations, and attempt to put them into cryptological terms.

It is important to note that there is no consensus on when a pseudo-random number generator can be regarded as adequate, both because the theory is very incomplete and because so many different fields are involved. Every journal that includes work on either cryptology or statistical methods is likely to include important papers, and no worker in the field is familiar with the whole literature. Broad agreement on criteria is the best that can be expected.

1 Overview

The basic random numbers for simulation are almost invariably generated as K-bit fixed-point fractions between zero and one, where K is usually between 32 and 64. As people would expect, this is based on the operations that computers provide, and we can expect to see increased use of 96- and 128-bit fractions over the next five years. A few generators produce bit-streams, but this is rarely convenient for statistical simulation.

The next stage is to transform these fixed-point fractions into the distributions that are actually needed. While this is a complex and interesting area, it is irrelevant to cryptology and will be ignored here.

If we view the problem as a game between the random number generator and the program that uses it, cryptology implies an intelligent opponent whereas simulation does not. This means that the critical question is not how easy the sequence is to predict, but how likely it is be partially predicted by accident. For example, simple multiplicative congruential transformations (with a known multiplier and modulus) are still one of the most reliable techniques for simulation [3], but useful mainly for student exercises in cryptanalysis.

Another key difference from cryptology is that many simulations are sensitive to extremely subtle variations from true randomness. This is partly because of their nature, and partly because they may use a great many numbers (10^9 is usual, and 10^{12} or more is becoming so). A generator with a theoretical information leak of only 0.0001% may be excellent or useless, depending on the nature of the leak.

Many big simulations are trying to explain extremely subtle variations in otherwise explained phenomena, and it is these ones that are so sensitive to non-randomness. This problem ties up with the previous one, in that any particular simulation will be sensitive to only a few forms of non-randomness. Most simulation work produces reliable results, even though much of it uses generators of quite appalling quality.

Efficiency is important, but rarely all-important, because of the other calculations that are needed to make use of each random number. A reasonable rule is that a random number generator needs to be as fast as a complex transcendental special function (say the error or power functions). The better DES software implementations achieve this [10], but the cryptologically strong methods do not, as is mentioned later.

2 Uniformity

The original criterion used for random number generators was whether they were adequately uniform, and this is still critical. Some Monte-Carlo work will give wrong answers with non-uniformities of 0.0001% or less. Note that the required uniformity is that expected of a true random number from a uniform distribution; excessive (i.e. deterministic) uniformity can be as harmful as non-uniformity, because it reduces the variation properties. Generators with excess uniformity fall into the category called quasi-random, and have their uses [2], but need great care.

Random uniformity cannot be measured directly, and so certain, more measurable properties are used. At least following aspects of uniformity are known to be important [9]:

- Gross uniformity (i.e. deviation of the cumulative distribution function from a straight line), usually tested by Kolmogorov-Smirnov or chi-squared.
- Uniformity of neighbours (i.e. variations of the probability density function from a constant value), tested by the birthday or variance of spacings tests.
- Multi-dimensional uniformity (of either form), which is particularly important for Monte-Carlo integration and allied simulations. Obviously, this is more measurable for a small number of dimensions.
- Local uniformity (i.e. the uniformity of small samples), as distinct from full-period uniformity. Real simulations use few numbers compared with the full period, but most theory applies to the latter.

These criteria can be used to give upper bounds on the sizes of simulation for which particular generators are adequate. Consider a K-bit generator with a period of N. Older generators used to have a single word of state, giving $N = 2^K$, but modern practice is to use multiple words of state [6], giving much longer periods N, typically between 2^{100} and 2^{10000}. This subtlety turns out to be rather important.

With $N = 2^K$, the well-known birthday test will start to show problems with samples of size $\sqrt{N}$, because the expected duplications will not occur.

This is rarely important for simulation, because the finite precision of floating point arithmetic means that few programs rely upon the equality of the numbers returned. Even so, it occasionally causes trouble.

Much more seriously, the variance of spacings test will start to show excess uniformity for simulations of size 2^{K-4} or $N^{2/3}$, whichever is the less [5]. Because many simulations rely upon adjacency properties, these figures should be regarded as hard limits to the maximum size of simulation. For example, a DES-encrypted counter should not be used for more than 7×10^{12} numbers.

The above limits are based upon the difference between sampling with and without replacement, and will thus apply to any periodic generator, whatever its underlying principles. However, the precise limits are for generators that cycle through all (or almost all) possible values in their state arrays, and generators that do not do this may have higher or lower limits. Sometimes it is possible to calculate these limits precisely, but not always.

These limits are easy to avoid, provided that they are known about and allowed for, but the other aspects of uniformity are harder. Nevertheless, it is essential to demonstrate near-perfect random uniformity before a new class of generator can be considered for simulation work, and this is one reason that number-theoretic methods remain so popular.

Most good cryptographic generators are obviously uniform in one dimension (and often more) over their full period, but analysing their multi-dimensional and small sample properties is harder. On the other hand, any major flaws for simulation would immediately lead to a cryptographic weakness. The real danger is that they may have small multi-dimensional information leaks (say $< 0.1\%$), of a nature that some simulations are extremely sensitive to.

3 Other tests

Theoretically, testing for multi-dimensional uniformity is adequate on its own, but this does not work in practice. The reasons are that the empirical uniformity tests for more than a couple of dimensions are too weak and the theoretical requirements are too strong. Hence a battery of empirical tests has been developed to check on particular properties that experience shows to be important [3].

One good example is the runs test, which turns out to be unusually powerful, though it is not quite clear why this should be. There are about a dozen standard tests, with many variations, and it is a matter of judgement which to use. The only universal rule is that the size of sample used to check a generator should be at least as large as that used for real simulations, but this is not always possible.

Some classes of generator have known weaknesses and, in these cases, the most reliable tests are often those designed to measure the particular weakness of the class. The spectral test for multiplicative congruential generators is the clearest example of this; it is reliable enough to predict how well a particular generator will perform on the empirical tests. There are few other examples as definite.

Actual experiments with DES- and SHA-based generators indicate that these are of very high quality, and they have passed all of the empirical tests that were tried with samples of up to 10^9. It seems unlikely that they have no weaknesses for use in simulation, but it can be said that they are definitely adequate for most simulation work. Only time will tell exactly how important their flaws are.

There is also the point that the standard range of statistical tests has been developed for generators that use number-theoretic algorithms, rather than the bit shuffling ones of DES and SHA. It is probable that a new generation of tests, based on different principles, will be needed to explore the weaknesses of these new generators. For example, it is likely that a test based on differential cryptanalytic techniques would be extremely powerful.

There have been several recent papers on universal tests for pseudorandom number generators, mostly based in information theory [7]. It is important to note that universal tests have been known to statisticians for half a century, but are little used because they tend to be very weak. This seems to apply to the cryptological tests as well, though it is difficult to prove that this must necessarily be the case. Note that we need to test generators with 1000 bits of state and a corresponding period.

4 Number-theoretic methods

One of the reasons that number-theoretic algorithms (like multiplicative congruential) are often used in preference to empirical ones (like DES) is that it is easier to analyse some randomness properties theoretically. This may well lead to cryptographic weaknesses, but that does not necessarily matter for use in simulation. As mentioned above, the weaknesses of well-understood generators can be measured and avoided.

An example is that generators based on primitive polynomials have proven full-period uniformities in dimensions up to the order of the polynomial. Note that this does **not** apply to linear additive generators. On the other hand, there is good evidence that this full-period uniformity does not translate into small sample randomness as well as for simple multiplicative congruential ones (i.e. polynomials of order 1).

This problem is one of the most important and intractable in pseudorandom number theory, and has an important parallel in differential cryptanalysis. The question of how long a subsequence can be and still be acceptably random is of key importance to the usability of a pseudorandom number generator.

5 'Perfect' generators

There are some published 'perfect' (i.e. cryptologically strong) generators, but these are not much used for simulation. The main reason is that they are published in journals that are not easily found or understood by statisticians. There are more fundamental reasons, but these are a statistician's interpretation of cryptological papers and may not be true for all cryptologically strong generators.

One problem is that they are based on unproven polynomial versus exponential complexity results, rather than the proven number-theoretic results of many traditional generators. Such complexity results are even less definite than asymptotic formulae, and it is unclear how relevant they are to necessarily bounded computer programs. We need to remember that the uniformity requirements are extremely strong, and that they are not just asymptotic in nature. This being said, it is likely that the cryptologically strong generators would prove reliable in practice.

A more important reason is efficiency. One of the faster perfect generators with P bits of state [8] can be used to produce a sequence of total size $O(P)$ bits at an average cost per bit of $O(P/logP)$. This is fine when looking for a few hundred numbers, but simulations often need 10^{12} 64-bit numbers, which implies unreasonable storage requirements and quite impossible time ones. By comparison, a very high-quality simulation generator will produce a sequence of total size $O(2^{P/2})$ bits at an average cost of $O(1)$.

Because the size limitation is actually $O(P^t)$ for some t, rather than $O(P)$, it is tempting to set t to (say) 4, which would remove the problem. Similarly, the number of bits returnable per iteration is bounded by $O(logP)$ and we could choose the constant to be (say) 10^9. But would the complexity results hold for practical, bounded simulations if we did this? It is very doubtful.

6 Parallelism

Parallel computing is becoming increasingly important, and this is one area in which the simulation requirements are actually more stringent than the cryptographic ones. If we use M generators in parallel, the need is for all M to appear independent, irrespective of how the numbers are sampled from each sequence or whether any weaknesses are predictable. Even restricting ourselves to periodic sampling strategies, there are $O(N^M)$ possible combinations; despite this immense complexity, there are some useful results [1,4].

As mentioned above, the spectral test is a reliable measure of the quality of multiplicative congruential generators. Ones with coprime moduli are independent in the spectral test sense, irrespective of the sampling strategy. The same is almost certainly true for their multi-dimensional uniformity, because of the relationship of this to the distribution of prime numbers, but it has not been proven. This gives a good basis for parallel generators.

Things are less clear for other classes of generator and, with exponential complexities on all sides, it is very hard to provide even heuristic proofs. The one thing that is clear is that any major failure of independence of parallel generators would lead to severe cryptographic weakness. As an aside, it is worth noting that many of the papers purporting to describe independent parallel pseudo-random generators are quite simply rubbish; one extreme example defines independence as disjointness, and proceeds from there.

7 Conclusion

It seems unlikely that the two schools of random number work will converge in the near future, but there is considerable scope for both sides to learn from the other. It is unfortunate that relevant work is published in so many journals in so many fields, not least because it enables so many outdated methods (and just plain nonsense) to get into print.

There are three main areas where theoretical advances would be extremely useful for simulation:

- Better results on which full-period theoretical properties lead to effective small sample randomness. This has been an active topic in number theory for a century or more, and so cannot be easy.
- Complexity results oriented towards more practical requirements; for example, not just polynomial versus exponential complexity, but the proportion of admissible tests that will reject a generator. This would be extremely interesting to cryptologists as well!
- More results on the independence of parallel generators. This has had the least attention, and has more scope for lateral thinking than the other areas. When and if large-scale parallelism becomes the standard technology, this will become critical.

References

[1] De Matteis, A. and Pagnutti, A., "Parallelization of random number generators and long-range correlations", in *Numerische Mathematik*, **53** (1988) pp 595–608.

[2] Hammersley, J.M. and Handscomb, D.C., *'Monte-Carlo Methods'*, Methuen, 1967.

[3] Knuth, D.E., *'The Art of Computer Programming', Vol. 2 / Seminumerical Algorithms'* second edition, Addison-Wesley 1981.

[4] Maclaren, N.M., "The generation of multiple independent sequences of pseudorandom numbers", in *Applied Statistics*, **38** (1989) pp 351–359.

[5] Maclaren, N.M., "A limit on the usable length of a pseudorandom sequence", in *J. Statist. Comput. Simul.*, **42**, (1992) pp 47–54.

[6] Marsaglia, G., "A current view of random number generators", in *Computer Science and Statistics*, Elsevier (1985).

[7] Maurer, U.M., "A Universal Statistical Test for Random Bit Generators", in *Advances in Cryptology - CRYPTO '90*, Springer-Verlag Lecture Notes in Computer Science **537** pp 409–420

[8] Micali, S. and Schnorr, C.P., "Efficient, Perfect Polynomial Random Number Generators", in *Journal of Cryptology*, **3** (1991) pp 157–172.

[9] Stuart, A. and Ord, J,K., *'Kendall's Advanced Theory of Statistics', Vol. 1*, Griffin, 5th Edition 1987.

[10] Young, E.A., *DES code on Internet*, from University of Queensland, Australia.