USING THE DIVISION ALGORITHM TO GENERATE PSEUDO RANDOM DECIMAL SEQUENCES [+]

Ghusn Alban Ali Ahmed [*]

Abstract:

This paper presents a random number generator, for use in generating cryptographic keystreams, based on the division algorithm. This generator permits choice of seed values giving a long cycle length that is known a priori and costlessly decimal keystream periodicity. The keystream can be used in Stream Cipher System.

المستخلص:

في هذا البحث تم تقديم مولد رقمي عشوائي يمكن استخدامه لتوليد سلسلة من المفاتيح والتي يمكن استخدامها في مجال التشفير. هذا المولد يعتمد في بنائه على خوارزمية القسمة.

في هذا المولد يمكن استخدام قيم ابتدائية لغرض توليد مفاتيح ذات دورة معروفة ومحددة .عملية التوليـــد عادة ما تكون بسيطة وغير معقدة وبالتالي ليس هناك كلفة عالية بالوقت والحجز بالذاكرة.

Introduction:

Random number generator plays an important role in cryptology. For example, a long pseudorandom key sequence can be added to plaintext to obtain ciphertext, as the basic version of polyalphabetic substitution. Thus it is important to have available a wide variety of random number generators, to aid in the search for the "best" one, see [1] for an extensive survey of this topic. Since linear random number generators tend to provide easy entry for the cryptanalyst, nonlinear random number generators are important for cryptology, see [2] for a recent example.

A random number generator can generate an initial number sequence of unpredictable length called a transient, which does not recur, followed by a repeating sequence which occurs over and over. Since it is vital both to avoid reusing any sequences and to acquire key sequence which are as long as possible, the most important property of a random number generator is the length of the repeating sequence-i.e., the cycle length [1]. Two problems tend to arise in the context of nonlinear random number generators: first, the probability of choosing a parameterization and a seed such that the true cycle length is sufficiently large, may not be known, and checking each generated key sequence for cycling can be costly in computer time. And second, rounding error will lead to cycling even when it would not otherwise occur, and the probability that such rounding-induced cycles will be of sufficiently greater length is not known.

The purpose of this paper is to present a simple nonlinear random number generator $A_1/N_1 + A_2/N_2 + \ldots$ (using no carry addition) A_i and N_i are positive integers with $A_i<N_i$ and A_i coprime to N_i for all i. The key sequence is the repeating sequence

[+] **Received on 20/2/2006 , Accepted on 22/12/2008**

[*] **Asst. Lecturer / Institute of technical Medical**

in the decimal of the seed. The cycle length depends on the N_i's; the remainder of this paper discusses how to choose the N_i's so as to achieve a large cycle length.

Some Properties of the Decimal Expansions:

The first six properties are either obvious or appear in most number theory textbooks with a chapter on decimal expansions (such as [3]). We will use the notation $C(N)$ to refer to the length of the repeating sequence (cycle) in the decimal expansion of $1/N$.

Prop.1: $C(N) \leq N-1$. Obviously, then, to get a long cycle we need to pick a large denominator N.

Prop.2: $C(N)$ divides evenly into $N-1$ for prime N.

Prop.3: $C(N) < N-1$ for composite N.

Prop.4: The cycle length of the decimal expansion of A/N, for A coprime to N, equals $C(N)$. here "coprime to" means "having no common factors with (other than unity)". Thus for a fully reduced fraction A/N, the cycle length is independent of A.

Prop.5: The cycle length of $A_1/N_1 + A_2/N_2 + \ldots$(using no carry addition), where each A_i/N_i is fully reduced, equals the lowest common multiple of $C(N_1)$, $C(N_2)$,….

This property is useful for building up the cycle length: if N_1, N_2,… are picked so that $C(N_1)$, $C(N_2)$,… have many factors not in common, then adding together the decimal fractions greatly increases the cycle length, yielding a cycle length which is known and potentially large.

Prop.6: If $N = 2^a 5^b n$ where $n>1$ and n is not divisible by 2 or 5, then the length of the transient of $1/N$ is $\max(a,b)$, and $C(N)=r$, where r is the smallest integer such that $10^r \equiv 1 \pmod{n}$.

This is the most advanced relevant theorem given in almost any number theory textbook. It appears, however, not to be useful for our purpose of finding a value N with a known, large cycle length. Note the implication of the first conclusion of the theorem for the question of whether we should choose an N which generates a transient as well as a cycle: suppose the computer can only handle numbers less than 2^D; then the longest feasible transient is generated by the seed $N=2^D-1$, and this transient has length $D-1$. Since D might be, say, 48 for typical computer, this transient is trivial in comparison to the cycle lengths we wish to generate. Therefore, it is best to choose N s.t. $a=0=b$ i.e., choose an N which is not divisible by 2 or 5 in order to maximize the possible size of n.

Prop.7: [4] If N, N', N'',... are distinct primes, then $C(N, N', N'',...)$ equals the lowest common multiple of $C(N)$, $C(N')$, $C(N'')$,….

This property allows cycle lengths to be built up as rapidly as does Prop.5, but requiring only a single division $1/ NN'N''....$ For example, use the assertion [1] that $C(q)=q-1$ for each of the primes q=97, 59, 47 and 29. Then $C(97*59*47*29)=\text{lcm}(96,58,46,28)=448,224$. Thus a cycle length of almost half a million is obtained using the decimal expansion of the reciprocal of 97*59*47*29 (i.e.

the reciprocal of 7,800,449), and of course a longer cycle could be achieved by continuing the building-up process.

Prop.8: [4] If N and N' have no common prime factors other than 2 and/or 5, then $C(NN')$ equals the LCM of $C(N)$ and $C(N')$.

Prop.9: [4] For prime p, if $C(p)=C(p^2)=\ldots=C(p^m)$ but $C(p^{m+1})\neq C(p)$, then for $c \geq m$ we have $C(p^c)=p^{c-m}C(p)$.

Fortunately, we don't have to rely absolutely on the validity of Prop.8. One acceptable procedure is to use Prop.7 and 9 to choose one or more divisors N_i, and then to use the algorithm in the next section to generate the expansions of A_i/N_i, knowing in advance the cycle length. Alternatively, Prop.8 could be used to choose one or more divisors with probable cycle length; the division algorithm in the next section has the great virtue of computing the cycle length at no additional computation cost. In either case, if more than one N_i was used, the quotients can be summed as in Prop.5 to generate a keystream with a longer known cycle length.

It is also asserted [2] that m=1 for all primes in the open interval (5,1000) with exception of N=487. Thus, for example, using the earlier assertion that $C(97)=96$, we have $C(97^c)=97^{c-1}*96$ for $c \geq 1$; so we can generate a key sequence with an arbitrarily large cycle length by using the decimal expansion of $1/97^c$ (subject of course to the constraint that 97^c must not be large enough to generate computer overfolw). Thus the prime p and the exponent c are the user-chosen parameters. This approach may build the cycle length faster than does the approach implied by Prop.7, but the resulting keystream may be less secure because it is based on fewer parameters.

Prop.10: [4] if q is a proper prime ending in a "1" i.e. if $C(q)=q-1$ and $q=10h+1$ for some h, then each digit 0,1,…,9 appears in the repeating sequence exactly $h=(q-1)/10$ times.

This follows directly from the fact that each possible remainder 1,…,q-1 occurs exactly once during the division algorithm for a proper prime. Thus for such seed values q, the digits 0 through 9 are guaranteed to have an exactly uniform frequency distribution in the complete key sequence. This is certainly a worthwhile from the point of view of cryptologic security. The smallest such q is 61, for which the reader can confirm that each digit appears exactly six times in the decimal expansion of 1/61.

The Division Algorithm:

The standard grade school division algorithm for A/N ($A<N$) can be expressed as follows, denoting the m^{th} key value (digit of the quotient) as s_m and using $[x]$ to denote the largest integer $\leq x$, the *Division Algorithm* is:

Division Algorithm
INPUT: 1. N, A, L ;
PROCESS: 2. $Y_0 = 10*A$;
3. $m := 1$;
4. Repeat
4.1 $s_m = [Y_{m-1} / N]$
4.2 $Y_m = 10*(Y_{m-1} - (s_m*N))$
4.3 $m := m + 1$;
Until $m=L$;
OUTPUT: 5. the keystream S ;
END.

Where L is the length of generated sequence. Here, "/" means "quotient", or in programming field can be denoted by "div", and the sequence of Y_m values, m=1,2,…, are the successive dividends. Y_m always has between 2 and d+1 digits inclusive, where d is the number of digits in N; so computer overfolw is never a problem if N and thus d is picked such that (d+1)-digit numbers can be handled by the computer. If we substitute the s_m equation into the Y_m equation we can see that this generator is a first order nonlinear dynamic equation in the state variable Y_m, with the key values s_m produced as a by-product. Lastly, the keystream S consists of the elements (s_1, s_2, …).

Example:

Let's pick N=7, A=1, and L=6 then

Step0: Y_0=10*1=10
Step1: s_1=10/7=1; Y_1=10(10-(1*7))=30;
Step2: s_2=30/7=4; Y_2=10(30-(4*7))=20;
Step3: s_3=20/7=2; Y_3=10(20-(2*7))=60;
Step4: s_4=60/7=8; Y_4=10(60-(8*7))=40;
Step5: s_5=40/7=5; Y_5=10(40-(5*7))=50;
Step6: s_6=50/7=7; Y_6=10(50-(7*7))=10;
OUTPUT: S=1,4,2,8,5,7.

If the user picks A/N such that A is coprime to N and N is coprime to 10, then Prop.6 can be shown to imply that the decimal expansion of A/N has no transient. The absence of a transient is convenient since it permits us to know exactly where the first cycle starts, and thus it allows us to build into the algorithm a costless stopping rule which identifies the end of the repeating sequence as well as the length of the cycle: stop at the first iteration m=M for which Y_M=Y_0; then the cycle length equals M. Note that computer rounding error, which can lead to premature cycling for some random number generators, never distorts the cycle length in the division algorithm.

The existence of this costless stopping rule is useful for two reasons. First, it serves as a check on a priori calculations of cycle length. And second, it permits the following alternative approach to choosing seed values A/N, without prior calculation of cycle length: always pick A=1; arbitrarily select large odd N, not ending in a "5", and compute the repeating sequence (i.e. one cycle) until the algorithm terminates; then arbitrarily select another large odd N not ending in a "5"; etc. In this way key sequences of arbitrary length can be pieced together from successive choices of N, without a priori calculation of the cycle length.

Further Thoughts and Conclusions:

Several additional observations are in the order here. First, we can get a feel for the nearness to uniformity of the digit frequency distributions we are likely to encounter in practice by checking the distribution for a few random examples. Recall from Prop.10 that seed values like 1/61 are guaranteed to give a perfectly uniform distribution. For 1/17, which has 16 digits in the repeating sequence, each digit 0 though 9 appears either once or twice, which is as close to uniformity as possible for a sequence of 16 digits. For 1/49, each digit appears either 4 or 5 times in the 42 digit

sequence, which again is as close to a uniform distribution as possible. For $1/243=1/3^5$, with a 27-digit sequence, each digit appears either 2 or 3 times, again as close as possible to uniformity. On the other hand, for 1/119=1/(7*17), with a 48-digit sequence, the digit frequencies range from one appearance of "9" to nine appearance of "zero" (including the two leading zeros at the beginning of the repeating sequence). Seven digits appear 4, 5, or 6 times each. Thus, while there appears to be a tendency toward uniformity of the frequency distribution of digits, this tendency is by no means absolute, it is plausible to speculate, however, that the frequency distribution of a combined sequence of the form $A_1/N_1 + A_2/N_2 + \ldots$ would strongly tend toward uniformity, with departures from uniformity by individual quotients tending to be offset by different departures by other quotients.

A second observation is that, given N, many numerators A will yield a string of zeros at the beginning of the repeating sequence. Since this might provide a point of entry for cryptanalysis, it would be a good practice to always drop all lead zeros from the key sequence.

It is useful to consider further the distribution of strings of zeros. Consider a proper prime p i.e., one for which $C(p)=p-1$. During the division algorithm p-1 distinct remainders occur, the integer from 1 through p-1. How many of them induce a series of L or more zeros in the keystream?

If $p=60\ldots01=6*10^{d-1}+1$ with d digits, for example, any remainder less than or equal to $6*10^{d-2}$ will cause Y_m (= ten times the remainder) to be $\leq 6*10^{d-1}$, inducing one or more zeros in the keystream. Likewise, any remainder $\leq 6*10^{d-L-1}$ will cause Y_m to be $\leq 6*10^{d-L}$, Y_{m+1} to be $\leq 6*10^{d-L+1}$, etc., inducing a string of at least L keystream zeros. Now the fraction of all remainder which are $\leq 6*10^{d-L-1}$ is $6*10^{d-L-1}/(6*10^{d-1})=1/10^L$ of the keystream digits is a zero followed by L-1 more zeros. But this is exactly what we would expect by chance in picking a "truly random" keystream: the probability that the next L digits drawn from a hat are all zeros is $(1/10)^L$. Thus the strings of zeros occur with the appropriate frequencies.

It is natural to speculate that other proper primes not of the form $c*10^{d-1}+1$ would also have appropriate frequencies for strings of zeros, as well as for strings of all other digits, and that some weaker version of this property would hold for N-values which are not proper primes. Note that the reason for the previous suggestion that the lead strings of zeroes be dropped from the keystream is clearly not that such strings should not be in a key sequence, but rather simply that the lead position would make them predictable.

Another recommendation is this: Avoid divisors of the form $N=99\ldots9$, since these give repeating sequences for $1/N$ of the form $00\ldots03$.

A more general rule of thumb is to avoid N-values which are heavily based on the number 3 and 9. In particular one should avoid $N=3^g$, which tends to give peculiar repeating sequences because of the fact that 3 divides evenly into 10^c-1 for any c. Consider, for example, the repeating sequence of $1/81=1/3^4$ which is 012345679: it contains all the digits except "8" in sequence. For 2/81 the sequence is 024691358, which roughly has the pattern of the even digits in order followed by the odd digits in order. For 4/81 the sequence is 049382716, which, ignoring the lead zero and grouped by digit pairs, shows each digit pair equal to eleven less than the previous pair. For

5/81 the sequence is 061728395, in which (aside from the final 5) each digit pair exceeds the previous pair by eleven. Consider also the 27-digit sequence generated by $1/243=1/3^5$: The first digit triplet is 004, and subsequent digit triplets exceed the previous triplet by 111, except that the final triplet exceeds the previous on by 112.
Such unfortunate patterns can and do occur in other sequences not based on 3, but they seem most pronounced when N is build-up from 3. Thus the dual recommendations are to avoid 3 when building up an N-values, and to obscure whatever more subtle patterns may exist in an expansion of A/N by using generators of the form $A_1/N_1 + A_2/N_2 +\ldots$. Note also that patterns tend to get obscured by the use of large A values (though using $A>1$ requires checking to make sure that A is coprime to N, to avoid decreasing the cycle length).

Future Works:

Several areas for future research suggest themselves:

1. As always, security can be enhanced by combining this encryption technique with a transposition scheme or with substitution based on another, different random number generator.
2. Are there other simple yet more security enhanced ways of combining the decimal quotients A_i/N_i rather than summing them?
3. The suggested random number generator can be treated as one unit in a system of n random number generators combined with each other by a combined boolean function F modulo 10 [5].

References:

1. Lagarias J. C. “Pseudorandom Number Generators”, Cryptology and Computational Number Theory, edited by C. Pomerance, Proceedings of Symposia in Applied Mathematics 42, American Mathematical Society, 1990.
2. Schneier B., “Applied Cryptography”, John Wiley & Sons, 1995.
3. Andrews, G. G, “Number Theory”, Dover Publications, October 1994.
4. Anderson J. A. and Bell J. M., “Number Theory with Applications”, Printice-Hall, 1997.
5. Whitesitt, J. E, “Boolean Algebra and its Application”, Dover Publications, April 1995.