Selim G. Akl & Henk Meijer

A Fast Pseudo Random Permutation Generator With Applications to Cryptology

Department of Computing and Information Science
Queen's University
Kingston - Ontario - Canada

1. Introduction

Pseudo random sequences of integers are most commonly generated by linear congruence methods [5] or by linear shift registers [1]. These sequences can be used in cryptology if they are cryptographically secure [9]:

> A pseudo-random sequence is cryptographically secure if given any segment of the sequence, it is computationally infeasible to compute other segments of the sequence.

It has been shown that sequences generated by linear congruence methods or by linear shift registers are not cryptographically secure [1,6,7]. By using one-way functions [2,3,9,10,11] or cryptographically strong encryption algorithms like the DES secure pseudo random number sequences can be constructed.

In this paper we propose a new class of generators. Members of the new class compare favourably with existing cryptographically strong generators since

- they are fast, even in software implementation, and
- they can quickly and easily be programmed in any computer language.

This work was supported by the Natural Sciences and Engineering Research Council of Canada under Strategic Grant G0381.

G.R. Blakley and D. Chaum (Eds.): Advances in Cryptology - CRYPTO '84, LNCS 196, pp. 269-275, 1985.

We first explain how the generators can be constructed in general. In sections 3 and 4 the generation of pseudo random permutations is described in detail. In section 5 it is shown how these permutations may be used in cryptology.

2. General scheme

Let S be a set with cardinality #S and let * and o be two binary operations on S. We construct a sequence $\{s^i\} = s^0, s^1, s^2, \ldots$ as follows:

Step 0 : Let x^0 be an arbitrary element of S.
Step 1 : $x^i = x^{i-1} * q$, for $i > 0$ and some $q \in S$.
Step 2 : $s^i = x^{ki} \circ x^{ki+1} \circ \ldots \circ x^{ki+k-1}$,
for all $i \geq 0$ and some positive integer k. []

The element q should be chosen such that the sequence $\{x^i\}$ has a large period. In the best case $\{x^i\}$ will have a period equal to #S, which implies that the period of the sequence $\{s^i\}$ is

$$\frac{\#S}{\gcd(k,\#S)}.$$

Obviously the set S, element q, integer k and operations * and o have to be chosen such that $\{s^i\}$ becomes cryptographically secure. We will use the above algorithm to construct a sequence of pseudo-random permutations. Although we have not been able to prove our random permutation generator secure, we conjecture that the sequence is secure since in our generator

i- q is chosen such that the period of $\{x^i\}$ is maximal

ii- the operations o and * do not have the distributive property

iii- the set S with operation o forms a group.

Fact -i- ensures that the sequence does not cycle in practice if S is chosen such that $\#S > 2^{200}$,

say. Fact -ii- makes it hard to simplify and solve equations that express elements of the sequenc $\{s^i\}$ in terms of the unknown quantities $\{x^i\}$ and q. Finally fact -iii- implies that

$$a = b \circ x$$

has a solution $x \in S$ for all $a, b \in S$. So if an element s^j of the sequence $\{s^i\}$ is known, then little is known about the values of

$$x^{kj}, x^{kj+1}, \ldots, x^{kj+k-1}.$$

3. Random permutations

Let P denote the set of all permutations of (0 1 2 ... n-1). Various algorithms that map the set $Z_{n!}$ onto the set P exist [4,5]. In this paper we use the mapping given in [5] and we denote it by f. So

$$f : Z_{n!} \rightarrow P$$

is a bijection. We compute the pseudo random sequence of permutations $\{p^i\}$ by the following algorithm:

Step 0: Let x^0 be an abitrary element of $Z_{n!}$ and $q \in Z_{n!}$ with $\gcd(n!,q) = 1$.

Step 1: $x^i = (x^{i-1} + q) \bmod n!$, for $i > 0$.

Step 2: $p^i = f(x^{ki}) \circ f(x^{ki+1}) \circ \ldots \circ f(x^{ki+k-1})$ for all k, $i \geq 0$ and some positive integer k, where $\circ$ is composition of permutations. []

Notice that the sequence generated in step 1 is a special case of the linear congruential sequence

$$x^i = a\, x^{i-1} + q \pmod m$$

with a=1 and m=n!. By choosing q relatively prime to n!, we are guaranteed a sequence $\{x^i\}$ of maximum period n! [5].

We will see in the next section how the above algorithm can be implemented efficiently.

4. Implementation

Elements in $Z_{n!}$ can be represented in a mixed-radix notation [5] as follows: if $x \in Z_{n!}$ it can be written as

$(x_0\ x_1\ \ldots\ x_{n-1})$ with

$$x = \sum_{i=0}^{n-1} x_i \cdot i!$$

and $0 \leq x_i \leq i$.

Addition modulo n! of two elements of $Z_{n!}$, written in this mixed-radix notation can be achieved with at most 2n additions, n subtractions and n comparisons.

If we denote a permutation $p \in P$ by the sequence (p(0) p(1) ... p(n-1)) then the mapping f from $Z_{n!}$ to P can be defined by [5]:

```
function f(x):
    let p(i) = i, for i=0, ..., n-1
    for i = n-1, n-2, ..., 1, 0 do
        exchange (p(i), p(x_i))
    end for
return p.  []
```

Since composition of permutations of size n can be achieved with n exchanges, it is obvious that each element of the sequence $\{p^i\}$ can be computed in O(kn) steps.

5. Generating bits

A pseudo random sequence of bits can be added bitwise modulo 2 to the bits of a message to construct a cryptogram. If the bit sequence is cryptographically secure, then so is

the resulting cryptogram. We show below how the random sequence of permutations can be used to produce a pseudo-random sequence of bits.

Let b^i be an arbitrary bit vector of length n and let p $[b^i]$ denote the bit vector that is the result of applying permutation p to vector b^i. If $\{b^i\}$ is an arbitrary pseudo-random sequence of bit vectors then we conjecture that $\{p^i [b^i]\}$ is a cryptographically secure bit stream.

In the algorithm below we use sequence $\{b^i\}$ defined by

b^0 = (01010 ... 0101)

$b^i = p^{i-1}[b^{i-1}]$, i > 0.

We propose to use k=2 and n=64. So the period of $\{x^i\}$ and therefore of $\{p^i [b^i]\}$ is approximately 2^{300}.

The algorithm applies the permutations $p^i = f(x^{2i}) \circ f(x^{2i+1})$ to the bit vectors b^i without in fact computing p^i, but rather by applying x^{2i} and x^{2i+1} directly to b^i. Implemented in the programming language C and running on a VAX 11/780, the algorithm generated more than 10 000 bits/second.

Algorithm to generate a bit stream:

```
n <- 64
k <- 2
x_0, x_1,..., x_n-1 <-initial value
q_0, q_1,..., q_n-1 <-integer relatively prime to n!
b_0, b_1,..., b_n-1 <- 0,1,0,1,0,1,...,0,1

repeat as many times as required
   do k times
      {compute x + q mod n!}
      carry <- 0
      for i = 1,2,..., n-1 do
          x_i <- x_i + q_i + carry
          if x_i > i then x_i <- x_i - (i+1)
                          carry <- 1
```

```
                    else carry <- 0
        end for
        {apply x to b }
        for i = n-1, n-2,..., 1 do
            exchange (b_i, b_x_i)
        end for
    end do
    output b_0, b_1,...,b_n-1
end repeat. []
```

Notice that the greatest common divisor of two numbers written in mixed radix-notation can easily be computed using the Euclidean algorithm.

6. Conclusion

We applied the statistical tests for randomness described in [1] to the bitstream generated by the algorithm described above. For these tests we chose n=64, k=2 and random values for q and we used an input bitstream of vectors $\{b^i\}$ given by

$$b^i = 111\ldots1000\ldots0 \quad \text{for } i=0,1,2,\ldots$$

where the number of 1's in each vector b^i has a binomial distribution with parameters 64 and 1/2. The generated bitstreams did not exhibit any statistical weaknesses.

References

[1] H. Beker and F. Piper, Cipher Systems, John Wiley, 1982.

[2] L. Blum, M. Blum and M. Shub, Comparison of two pseudo-random generators, Proceedings of Crypto 1982.

[3] M. Blum and S. Micali, How to generate cryptographically strong sequences of pseudo random bits, FOCS, 1982.

[4] G.D. Knott, A numbering system for permutations and combinations, Communications of the ACM Vol. 19, No. 6, June 1976.

[5] D.E. Knuth, The Art of Computer Programming, Volume 2: Seminumerical Algorithms, Addison-Wesley, 1981.

[6] D.E. Knuth, Deciphering a linear congruential encryption, Technical Report 024800, Stanford University, 1980.

[7] J.B. Plumstead, Inferring a sequence generated by a linear congruence, FOCS, 1982.

[8] J. Reeds, Cracking a random number generator, Cryptologia, Vol. 1, January 1977.

[9] A. Shamir, On the generation of cryptographically strong pseudorandom sequences, ACM Transactions on Computer Systems, Vol. 1, No. 1, February 1983.

[10] U.V. Vazirani and V.V. Vazirani, Trapdoor pseudo-random number generators with applications to protocol design, FOCS, 1983.

[11] A. Yao, Theory and applications of trapdoor functions, FOCS, 1982.