2.5 Linear Congruential Generators with Unknown Module

The attack on linear congruential generators surely becomes harder if the module m is kept secret and cannot be guessed in an obvious way. We assume that the attacker has a (short) subsequence $x_0, x_1, \ldots$ of the output sequence at her disposal.

Surprisingly it is easier to attack the multiplier first. The following proposition yields a "surrogate" value a' in a few steps. Note the Noetherian approach via the (implicit) formula $y_{t+1} \in \mathbb{Z}y_1 + \cdots + \mathbb{Z}y_t$, the principal ideal generated by the integer $\gcd(y_1, \ldots, y_t)$.

Proposition 8 (Plumstead-Boyar) *Let (y_i) be the sequence of differences of the linear congruential generator with generating function $s(x) = ax + b \bmod m$, $m \geq 2$, and initial value x_0. Let $y_1 \neq 0$ and t be the smallest index such that $e = \gcd(y_1, \ldots, y_t) \mid y_{t+1}$. Then:*

(i) $t < 1 + \log_2 m$.

(ii) *If $e = c_1 y_1 + \cdots + c_t y_t$ with $c_i \in \mathbb{Z}$ and $a' = (c_1 y_2 + \cdots + c_t y_{t+1})/e$, then $a' \in \mathbb{Z}$ and*

$$y_{i+1} \equiv a' y_i \pmod{m} \quad \text{for all } i.$$

(iii) *If $b' = x_1 - a' x_0$, then*

$$x_i = a' x_{i-1} + b' \bmod m \quad \text{for all } i.$$

Proof. (i) If $e_j = \gcd(y_1, \ldots, y_j)$ doesn't divide y_{j+1}, then $e_{j+1} \leq e_j/2$. Since $e_1 = |y_1| < m$ we conclude $e = e_t < m/2^{t-1}$, hence $t - 1 < \log_2 m$.

(ii) We have

$$ae = c_1 a y_1 + \cdots + c_t a y_t \equiv c_1 y_2 + \cdots + c_t y_{t+1} = a' e \pmod{m}.$$

The greatest common divisor d of m and y_1 divides e by Lemma 6, hence also $d = \gcd(m, e)$. We divide the congruence first by d:

$$a \frac{e}{d} \equiv a' \frac{e}{d} \pmod{\bar{m}}$$

with the reduced module $\bar{m} = m/d$. Since e/d and $\bar{m}$ are coprime we may divide by e/d:

$$a \equiv a' \pmod{\bar{m}}, \quad a = a' + k\bar{m}.$$

Hence $y_{i+1} \equiv a y_i = a' y_i + k y_i \bar{m} \pmod{m}$. From $d \mid y_i$ follows $y_i \bar{m} \equiv 0$, hence $y_{i+1} \equiv a' y_i \pmod{m}$.

(iii) is an immediate consequence of Lemma 6 (viii). ◇

Examples

1. Let $m = 8397$, $a = 4381$, $b = 7364$ [Reeds 1977]. Generate

$$\begin{array}{lll} x_0 = 2134 & & \\ x_1 = 2160 & y_1 = 26 & e_1 = 26 \\ x_2 = 6905 & y_2 = 4745 & e_2 = 13 \\ x_3 = 3778 & y_3 = -3127 & e_3 = 1 \\ x_4 = 8295 & y_4 = 4517 & \end{array}$$

   We get $c_1 = 87542$, $c_2 = -481$, $c_3 = -1$, and $a' = 416881843$.

2. Let $m = 2^q + 1$, $a = 2^{q-1}$, $b = 2^q$, and $x_0 = 0$. By the corollary of the following Lemma 7 we have $y_i = (-1)^{i-1} \cdot 2^{q-i+1}$ for $i = 1, \ldots, q+1$, and thus $e_i = 2^{q-i+1}$. Hence $t = q + 1$. Thus the upper bound for t in Proposition 8 is sharp, and indeed we need the $q + 3$ elements x_0 to x_{q+2} of the output sequence to determine the surrogate multiplier a'.

Lemma 7 *Let the sequence (c_i) in $\mathbb{Z}$ be defined by $c_0 = 0$, $c_i = 2^{i-1} - c_{i-1}$ for $i \geq 1$. Then*

(i) $c_i = \frac{1}{3} \cdot [2^i - (-1)^i]$ *for all i,*

(ii) $c_i - 2c_{i-1} = (-1)^{i-1}$ *for all $i \geq 1$.*

Proof. (i) follows by induction, (ii) by a direct calculation. ◇

Corollary 1 *Let (x_i) be the output sequence of the linear congruential generator with module $m = 2^q + 1$, multiplier $a = 2^{q-1}$, increment $b = 2^q$, and initial value $x_0 = 0$. Let (y_i) be the sequence of differences. Then*

(i) $x_i = c_i \cdot 2^{q-i+1}$ *for $i = 0, \ldots, q+1$,*

(ii) $y_i = (-1)^{i-1} \cdot 2^{q-i+1}$ *for $i = 1, \ldots, q+1$.*

Proposition 8 provides a surrogate multiplier in an efficient way. Now we need a procedure for determining the module m. We close in on it by "successive correcting". In step j we determine a new surrogate module m_j and a new surrogate multiplier a_j as follows:

- In the first step set $m_1 = \infty$ and $a_1 = a'$. [Calculating mod ∞ simply means calculating with integers, and $\gcd(c, \infty) = c$ for $c \neq 0$, but $= \infty$ for $c = 0$.]
- In step j, $j \geq 2$, let $y'_j := a_{j-1} y_{j-1} \bmod m_{j-1}$. Then set $m_j = \gcd(m_{j-1}, y'_j - y_j)$ and $a_j = a_{j-1} \bmod m_j$.

Thus in iteration step j we use the current surrogate values m_{j-1} and a_{j-1} for m and a and predict a value y'_j for y_j that we compare with the real (known) value y_j. If these two numbers differ, then their difference is a multiple of m. In this case we correct the surrogate values. We always have $m \mid m_j$. The corrected values don't invalidate the former calculations since $y_i \equiv a_j y_{i-1} \pmod{m_j}$ for $i = 2, \ldots, j$, and also $y_i \equiv a_j y_{i-1} \pmod{m}$ for all $i \geq 2$. Also the true sequence (x_i) always fulfils $x_i \equiv a_j x_{i-1} + b_j \pmod{m_j}$ for $i = 1, \ldots, j$ with $b_j = x_1 - a_j x_0$ by Lemma 6 (viii).

In Example 1 above we have

$$\begin{array}{lll} & m_1 = \infty & a_1 = 416881843 \\ y'_2 = 10838927918 & m_2 = 10838923173 & a_2 = 416881843 \\ y'_3 = 5420327549 & m_3 = 8397 & a_3 = 4381 \end{array}$$

The calculation for m_3 is

$$\gcd(10838923173, 5420330676) = 8397.$$

Since $m_3 \leq 2x_2$ we conclude that necessarily $m = m_3$, $a = a_3$, and $b = x_1 - ax_0 \bmod m = 7364$. Thus we found the true values after two correction steps, and we didn't need any further elements of the output sequence than the five we used for determining a'. Note the large intermediate results that suggest that in general the procedure relies on multi-precision integer arithmetic.

Does the procedure always terminate? At the latest when we reach the period of the sequence, that is after at most m steps, the complete sequence is predictable. However this bound is practically useless. Unfortunately it is tight: For arbitrary m let $a = 1$, $b = 1$, and $x_0 = 0$. Then $x_i = i$ and $y_i = 1$ for $i = 0, \ldots, m-1$. The initial value for the surrogate multiplier is $a' = 1$. The first false prediction is $y'_m = 1$ instead of the correct value $y_m = 1 - m$. The end is reached only after evaluating x_m. Although this worst case is easily recognized and might be treated separately it nevertheless hints at the difficulty of finding good general results. And indeed we don't know of any.

From a slightly different point of view we count the number of necessary correction steps where the surrogate module changes. For if $m_j \neq m_{j-1}$, then $m_j \leq m_{j-1}/2$. Let $m^{(0)} = \infty > m^{(1)} > \ldots$ be the sequence of *distinct* surrogate modules. Then

$$m^{(1)} = m_{j_1} = |y'_{j_1} - y_{j_1}| < a'|y_{j_1-1}| + m < m(a'+1),$$

$$m \leq m^{(j)} < \frac{m(a'+1)}{2^{j-1}},$$

hence always $j < 1 + \log_2(a'+1)$. This gives an upper bound for the number of necessary corrections. Joan Plumstead-Boyar described a variant of the

algorithm that results in a potentially smaller value of a', and eventually in the upper bound $2 + \log_2 m$ for the number of correction steps. However in general the algorithm doesn't involve that many corrections making this bound obsolete as a terminating criterion.

It seems that the search for theoretical results is a worthwile task. Could we exclude a (maybe small) class of (maybe bad anyway) linear congruential generators such that the majority of the remaining (interesting) generators obey a practically useful terminating criterion? I would expect such a result. Is there a way to control the distribution of the number of steps? Or at least the mean value?

Anyway the known results suffice to disqualify linear congruential generators for direct cryptographic application.

For an implementation of this algorithm in C see `https://www.staff.uni-mainz.de/pommeren/Cryptology/Bitstream/2_Analysis/LCGcrack.html`.