Training Exercise about Linear Congruential Generators

Master 1 of Applied Mathematics

Maxime Bros

October 2019

1 Exercise 1

As seen in the first exercise of the *TP 2*, a Linear Congruential Generator (LCG) generates a sequence of integers using the following recurrence relation :

$$s_{i+1} = as_i + b \mod m, \quad i \geq 0 \tag{1}$$

where s_0 is called the **seed** and (a, b, m) are the parameters of the LCG.

Such a generator is not secure and should never be used for cryptographic purposes. Indeed, given only a few outputs of the LCG, it is possible to recover all the parameters and thus to predict the next output. A classic idea to improve its security is to use a truncated version of the LCG: for instance, one could take only the last bit (the least significant one) of each integer of the sequence. This idea is widely used in cryptography, for instance with LFSR or with the BBS generator. Despite this modification, the truncaded LCG is still not secure at all.
Nevertheless, it is a nice training exercise to generate a sequence of pseudo-random bytes in order to decipher a file.

On my website `www.maxime-bros.fr/teaching`, in the Section *TP 2*, there is a file `ciphertext`; it corresponds to a plaintext file for which each bit has been XOR-ed with a bit of the following truncated (only the last bit is considered) LCG:

$$(a = 211,\ b = 84629,\ m = 29886029)$$

The seed used was 23, it corresponds to the **secret key**.

To make it easier to understand, here are the first 8 numbers of the sequence, and then the corresponding first bytes in the ciphertext and in the plaintext.

$$\begin{aligned}
s_0 &= 23 \\
s_1 &= 211 \times 23 + 84629 \mod 29886029 = 89482 \\
s_2 &= 18965331 \\
s_3 &= 26927613 \\
s_4 &= 3465462 \\
s_5 &= 14032415 \\
s_6 &= 2207323 \\
s_7 &= 17539347 \\
s_8 &= 24905279
\end{aligned} \tag{2}$$

As we consider the truncated LCG, we just keep the least significant bit of each number (starting with s_1), this gives the following binary number

$$(0110\ 1111)_2 = (111)_{10}$$

The first byte of the ciphertext is $(90)_{16} = (144)_{10} = (1001\ 0000)_2$, so the first byte of the plaintext is indeed $144 \oplus 111 = 255$ (i.e FF in hexadecimal).

The purpose of this exercise is to generate as many bytes as there are in the ciphertext and to "XOR" them with the ciphertext in order to recover the original plaintext. *Hint: you can re-use the program you made in TP 1 to XOR two files.*

2 References

For more informations about the security and the cryptanalysis of LCG, reader should refer to "Secret Linear Congruential Generators are not Cryptographically Secure" by Jacques Stern.