Stream Cipher
Pseudorandom Number Generator

Sareh Jalalizad

The PRNG was designed based on the Linear Congruential Generator according to the lecture slides Symmetric Key Encryption I.

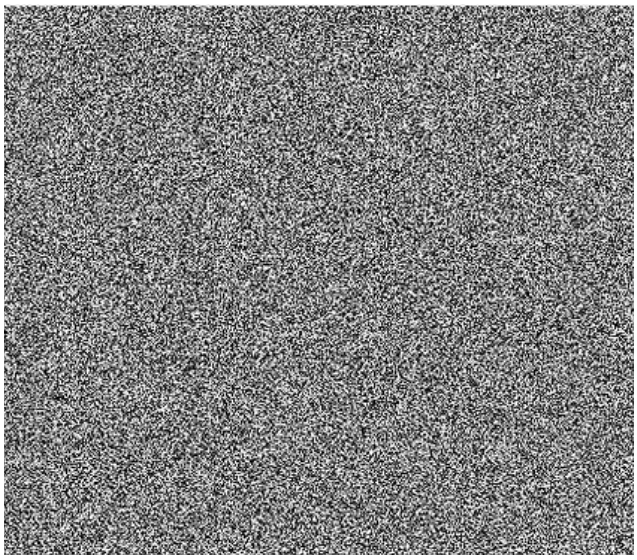
It generates uniform random numbers by using modulo arithmetic and a recursive calculation:

$$xi+1 = axi + b \bmod m$$

The following conditions were considered in design of this generator:
- Very long period
- multiplier a ε [1, m – 1] , constant b ε [0, m – 1]

When choosing a number for m in a linear congruential generator (LCG), there are different strategies, however, in order to safely encrypt as many bytes as feasible, we needed to choose a high value modulus and a primitive root that would result in a high period.
The most common is for m to be a prime (according to Euler's function, prime numbers give long period). As a result, I chose the largest prime that can be represented by int.[1,2] The variable modulus that I've chosen is
$2^{31} - 1 = 2,147,483,647$.
The suggested primitive root modulo M31 which is tried and true is
a = 48, 271, now known as MINSTD. Languages like C++11 make use of these values.[3] The offset b does not change the statistical properties, so I have just set the constant to 0.

Java's java.util.Random use 2^{48} for m and 25214903917 ($5DEECE66D_{16}$) for multiplier a.[4] So, $\Phi(m) = 2^{47} = 140737488355328$.



Bitmap generated by Java Random[5]