Fortuna PRNG

Abstract

This document will give a short summary about the Fortuna pseudo random number generator in general and also some details about this specific implementation.

The algorithm

The very short summary is that it is a normal pseudo random generator using a cryptographic function but with several additions that adds a lot of unpredicability to it.

The algorithm consists of several parts, the generator, accumulator and a set of entropy sources.

The generator is the part that generates the random data. It does this by using a block cipher with a 256-bit key to encrypt a 128-bit counter. The counter is incremented for each block of random data generated and the key is replaced by generating two additional blocks after each request for blocks of random data.

Reseeding of the generator is done by using the SHA-256 hash function to hash the current cipher key together with the seed data supplied and replacing the key with the result.

The accumulator uses entropy sources to collect data which will be used to reseed the generator. Each entropy source tries to be an upredictable provider of bytes of data. The sources put their bytes of data into pools of collected entropy. There are 32 pools and each source separately keeps track of when it should add data and which pool it should put the data into, simply looping over them.

Entropy sources can be basically anything, as long as at least one of them is unpredictable for an attacker either in the data provided or the timing of the data provided an attacker will not be able to predict the key that eventually will be supplied to the generator.

The accumulated data is used to create seed data for the generator by picking a number of pools of entropy data and using SHA-256 to hash their data. Pools are reset when their data is used. Pools are picked if 2 to the power of the number of the pool is a divisor of the total number of reseeds done. This means that the first pool is used every reseed, the second is used every other, the third is used every fourth and so on.

Reseeding is done automatically when there is enough entropy data available. This is determined by looking at the first pool of data, since it is used for every reseed, and simply checking if it has more than a set amount of bytes. This is checked every time random data is fetched.

For more specifics and also calculations showing the robustness of the algorithm read chapter 9 of the book Cryptographic Engineering (ISBN: 9780470474242) where this algorithm is described by the creators of it.

Implementation specifics

This implementation is in pure java and considerations were made to make it as easy to use as possible and to avoid potential mistakes when adding it to a project.

For the generator a public domain implementation of the AES-256 cipher was chosen over Java:s own implementation since it requires additional system configuration.

As of the writing of this document the implementations has seven different entropy sources based on different things that are available to the JavaVM. For instance thread timing, memory usage and garbage collection statistics all of which are hard to affect in a predictable manner.