INSTITUTE FOR TESTING AND CERTIFICATION Inc.
Street T. Bati 299, 764 21 Zlín, Czech Republic

CERTIFICATION REPORT

Test Certificate for RNG Testing
No.: 11108133

Issued to Applicant:	Amatic Industries GmbH Traunsteinstrasse 12 A – 4845 Rutzenmoos Austria
Subject of Inspection:	Random Number Generator (RNG): RNG Windows
Scope of inspection:	RNG Evaluation in accordance with the Methodology applied by ITC
Test Engineers:	Mr. Ivan Dvořák, Mr. Martin Kořínek
Result of inspection:	Satisfactory
Issue Date:	26 October 2011

AUTORIZOVANÁ OSOBA 224
INSTITUT PRO TESTOVÁNÍ A CERTIFIKACI, a.s.
ÚNMZ
1

Michal Plešr
Head of gambling games certification

RNG description

The Random Number Generator (RNG) is an Implementation of the KISS Algorithm by George Marsaglia and Arif Zaman.

This remarkably short and fast generator combines 3 different simple generators and has a period of around 10^{37} i.e. it will start repeating the same sequence of numbers after that many calls. It is nessesary to set the seed values (at least x, y and z need to be changed) to random starting values else it will always generate the same sequence of numbers in the program. Avoid setting the seeds to zero or small numbers in general – it is adviced to choose large "complex" seed values

A nice consequence of combining different RNGs is that a statistical flaw in any one of the component generators is likely to be covered up by the other generators. Combining different RNGs is now considered sound practice in designing good RNGs by many experts in the field. KISS RNG represents the minimum acceptable standard in random number generation.

The Seed:

At Construction of the RNG the Seed is set with different time parameters (Performance Counter, Milliseconds).

Shuffling:

The RNG itself shows good test results, but nevertheless the parameters for the RNG are shuffled within everygame.

In every game there are periodic cycles where the system checks the buttons, additional credits, errors...with the global CheckError() method the RNG parameters are shuffled!

RNG source code

```
/*

 KISS

 the idea is to use simple, fast, individually promising
 generators to get a composite that will be fast, easy to code
 have a very long period and pass all the tests put to it.
 The three components of KISS are
        x(n)=a*x(n-1)+1 mod 2^32
        y(n)=y(n-1)(I+L^13)(I+R^17)(I+L^5),
        z(n)=2*z(n-1)+z(n-2) +carry mod 2^32
 The y's are a shift register sequence on 32bit binary vectors
 period 2^32-1;
 The z's are a simple multiply-with-carry sequence with period
 2^63+2^32-1.  The period of KISS is thus
      2^32*(2^32-1)*(2^63+2^32-1) > 2^127
*/

#define ulong unsigned long

static ulong kiss_x = 1;
static ulong kiss_y = 2;
static ulong kiss_z = 4;
static ulong kiss_w = 8;
static ulong kiss_carry = 0;
static ulong kiss_k;
static ulong kiss_m;

void seed_rand_kiss(ulong seed) {
    kiss_x = seed | 1;
    kiss_y = seed | 2;
    kiss_z = seed | 4;
    kiss_w = seed | 8;
    kiss_carry = 0;
}

ulong rand_kiss() {
    kiss_x = kiss_x * 69069 + 1;
    kiss_y ^= kiss_y << 13;
    kiss_y ^= kiss_y >> 17;
    kiss_y ^= kiss_y << 5;
    kiss_k = (kiss_z >> 2) + (kiss_w >> 3) + (kiss_carry >> 2);
    kiss_m = kiss_w + kiss_w + kiss_z + kiss_carry;
    kiss_z = kiss_w;
    kiss_w = kiss_m;
    kiss_carry = kiss_k >> 30;
    return kiss_x + kiss_y + kiss_w;
}
```

Testing conditions

For testing purposes numbers from RNG is used. Exactly 20 blocks size of 1.000.000 binary numbers, each block with new seed. That is total 20.000.000 binary numbers saved to a single file, numbers is represented by ASCII zeroes and ones or as a bit stream.

Numbers are tested by **STS** (*A Statistical Test Suite for the Validation of Random Number Generators and Pseudo Random Number Generators for Cryptographic Applications*) by **NIST** (National Institute of Standards and Technology). More about NIST's STS could be found here:
http://csrc.nist.gov/groups/ST/toolkit/rng/

STS tests are done on PC machine:

Operating System
Linux Ubuntu 11.10 desktop i386 LiveCD

CPU
Intel Core i7 920 @ 2.67GHz
Bloomfield 45nm Technology

RAM
6.0GB Triple-Channel DDR3 @ 539MHz (8-8-8-20)

Motherboard
Gigabyte Technology Co., Ltd. EX58-UD5 (Socket 1366)

Hard Drives
293GB Western Digital WDC WD3000GLFS-01F8U0 ATA Device (IDE)

STS is compiled using gcc under Linux Ubuntu:
Linux Ubuntu 11.10 desktop i386 LiveCD
Linux 3.0.0-12-generic
gcc (Ubuntu/Linaro 4.6.1-9ubuntu3) 4.6.1

STS version:
sts-2.1.1

STS tests parameters:

[1] Block Frequency Test - block length(M):	128
[2] NonOverlapping Template Test - block length(m):	9
[3] Overlapping Template Test - block length(m):	9
[4] Approximate Entropy Test - block length(m):	10
[5] Serial Test - block length(m):	16
[6] Linear Complexity Test - block length(M):	500

RNG Hardware Details

There are no hardware details, RNG is strictly software base.

RNG Software Details

The RNG is built into the system. The RNG is compiled into the DLL called "SysUtil.dll" which exists inside the so called "Execute Container" (file: mg_Exec_000.acf). The Execute Container is the file that is installed onto the machine.

File name:	mg_Exec_000.acf
Media Type:	Application / Execute Container
Function:	Event result generator
Size:	71 735 kB
MD5:	a2cd0acd7374e46d1b2afb32ec29a682

Applied Tests Descriptions

The NIST Test Suite is a statistical package consisting of 15 tests that were developed to test the randomness of (arbitrarily long) binary sequences produced by either hardware or software based cryptographic random or pseudorandom number generators. These tests focus on a variety of different types of non-randomness that could exist in a sequence. Some tests are decomposable into a variety of subtests. The 15 tests are:

1. The Frequency (Monobit) Test,
2. Frequency Test within a Block,
3. The Runs Test,
4. Tests for the Longest-Run-of-Ones in a Block,
5. The Binary Matrix Rank Test,
6. The Discrete Fourier Transform (Spectral) Test,
7. The Non-overlapping Template Matching Test,
8. The Overlapping Template Matching Test,
9. Maurer's "Universal Statistical" Test,
   Lempel-Ziv Complexity Test
10. The Linear Complexity Test,
11. The Serial Test,
12. The Approximate Entropy Test,
13. The Cumulative Sums (Cusums) Test,
14. The Random Excursions Test, and
15. The Random Excursions Variant Test.

The order of the application of the tests in the test suite is arbitrary. However, it is recommended that the Frequency test be run first, since this supplies the most basic evidence for the existence of non-randomness in a sequence, specifically, non-uniformity. If this test fails, the likelihood of other tests failing is high.

The statistical package includes source code and sample data sets. The test code was developed in ANSI C. Some inputs are assumed to be global values rather than calling parameters.
A number of tests in the test suite have the standard normal and the chi-square (χ^2) as reference distributions. If the sequence under test is in fact non-random, the calculated test statistic will fall in extreme regions of the reference distribution. The standard normal distribution (i.e., the bell-shaped curve) is used to compare the value of the test statistic obtained from the RNG with the expected value of the statistic under the assumption of randomness. The test statistic for the standard normal distribution is of the form $z = (x - \mu)/\sigma$, where x is the sample test statistic value, and μ and σ^2 are the expected value and the variance of the test statistic. The χ^2 distribution (i.e., a left skewed curve) is used to compare the goodness-of-fit of the observed frequencies of a sample measure to the corresponding expected frequencies of the hypothesized distribution. The test statistic is of the form

$$\chi^2 = \sum \left((o_i - e_i)^2 / e_i \right)$$

where o_i and e_i are the observed and expected frequencies of occurrence of the measure, respectively.

For many of the tests in this test suite, the assumption has been made that the size of the sequence length, n, is large (of the order 10^3 to 10^7). For such large sample sizes of n, asymptotic reference distributions have been derived and applied to carry out the tests. Most of the tests are applicable for smaller values of n. However, if used for smaller values of n, the asymptotic reference distributions would be inappropriate and would need to be replaced by exact distributions that would commonly be difficult to compute.

1. Frequency (Monobits) Test

Description: The focus of the test is the proportion of zeroes and ones for the entire sequence. The purpose of this test is to determine whether that number of ones and zeros in a sequence are approximately the same as would be expected for a truly random sequence. The test assesses the closeness of the fraction of ones to ½, that is, the number of ones and zeroes in a sequence should be about the same.

2. Test for Frequency within a Block

Description: The focus of the test is the proportion of zeroes and ones within M-bit blocks. The purpose of this test is to determine whether the frequency of ones is an M-bit block is approximately M/2.

3. Runs Test

Description: The focus of this test is the total number of zero and one runs in the entire sequence, where a run is an uninterrupted sequence of identical bits. A run of length k means that a run consists of exactly k identical bits and is bounded before and after with a bit of the opposite value. The purpose of the runs test is to determine whether the number of runs of ones and zeros of various lengths is as expected for a random sequence. In particular, this test determines whether the oscillation between such substrings is too fast or too slow.

4. Test For The Longest Run Of Ones In A Block

Description: The focus of the test is the longest run of ones within M-bit blocks. The purpose of this test is to determine whether the length of the longest run of ones within the tested sequence is consistent with the length of the longest run of ones that would be expected in a random sequence. Note that an irregularity in the expected length of the longest run of ones implies that there is also an irregularity in the expected length of the longest run of zeroes. Long runs of zeroes were not evaluated separately due to a concern about statistical independence among the tests.

5. Random Binary Matrix Rank Test

Description: The focus of the test is the rank of disjoint sub-matrices of the entire sequence. The purpose of this test is to check for linear dependence among fixed length substrings of the original sequence.

6. Discrete Fourier Transform (Spectral) Test

Description: The focus of this test is the peak heights in the discrete Fast Fourier Transform. The purpose of this test is to detect periodic features (i.e., repetitive patterns that are near each other) in the tested sequence that would indicate a deviation from the assumption of randomness.

7. Non-Overlapping (Aperiodic) Template Matching Test

Description: The focus of this test is the number of occurrences of pre-defined target substrings. The purpose of this test is to reject sequences that exhibit too many occurrences of a given non-periodic (aperiodic) pattern. For this test and for the Overlapping Template Matching test, an m-bit window is used to search for a specific m-bit pattern. If the pattern is not found, the window slides one bit position. For this test, when the pattern is found, the window is reset to the bit after the found pattern, and the search resumes.

8. Overlapping (Periodic) Template Matching Test

Description: The focus of this test is the number of pre-defined target substrings. The purpose of this test is to reject sequences that show deviations from the expected number of runs of ones of a given length. Note that when there is a deviation from the expected number of ones of a given length, there is also a deviation in the runs of zeroes. Runs of zeroes were not evaluated separately due to a concern about statistical independence among the tests. For this test and for the Non-overlapping Template Matching test, an m-bit window is used to search for a specific m-bit pattern. If the pattern is not found, the window slides one bit position. For this test, when the pattern is found, the window again slides one bit, and the search is resumed.

9. Maurer's Universal Statistical Test

Description: The focus of this test is the number of bits between matching patterns. The purpose of the test is to detect whether or not the sequence can be significantly compressed without loss of information. An overly compressible sequence is considered to be non-random.

Lempel-Ziv Complexity Test

Description: The focus of this test is the number of cumulatively distinct patterns (words) in the sequence. The purpose of the test is to determine how far the tested sequence can be compressed. The sequence is considered to be non-random if it can be significantly compressed. A random sequence will have a characteristic number of distinct patterns.

10. Linear Complexity Test

Description: The focus of this test is the length of a generating feedback register. The purpose of this test is to determine whether or not the sequence is complex enough to be considered random. Random sequences are characterized by a longer feedback register. A short feedback register implies non-randomness.

11. Serial Test

Description: The focus of this test is the frequency of each and every overlapping m-bit pattern across the entire sequence. The purpose of this test is to determine whether the number of occurrences of the 2m m-bit overlapping patterns is approximately the same as would be expected for a random sequence. The pattern can overlap.

12. Approximate Entropy Test

Description: The focus of this test is the frequency of each and every overlapping m-bit pattern. The purpose of the test is to compare the frequency of overlapping blocks of two consecutive/adjacent lengths (m and m+1) against the expected result for a random sequence.

13. Cumulative Sum (Cusum) Test

Description: The focus of this test is the maximal excursion (from zero) of the random walk defined by the cumulative sum of adjusted (-1, +1) digits in the sequence. The purpose of the test is to determine whether the cumulative sum of the partial sequences occurring in the tested sequence is too large or too small relative to the expected behaviour of that cumulative sum for random sequences. This cumulative sum may be considered as a random walk. For a random sequence, the random walk should be near zero. For non-random sequences, the excursions of this random walk away from zero will be too large.

14. Random Excursions Test

Description: The focus of this test is the number of cycles having exactly K visits in a cumulative sum random walk. The cumulative sum random walk is found if partial sums of the (0,1) sequence are adjusted to (-1, +1). A random excursion of a random walk consists of a sequence of n steps of unit length taken at random that begin at and return to the origin. The purpose of this test is to determine if the number of visits to a state within a random walk exceeds what one would expect for a random sequence.

15. Random Excursions Variant Test

Description: The focus of this test is the number of times that a particular state occurs in a cumulative sum random walk. The purpose of this test is to detect deviations from the expected number of occurrences of various states in the random walk.

Applied Tests Results

```
-------------------------------------------------------------------------------
RESULTS FOR THE UNIFORMITY OF P-VALUES AND THE PROPORTION OF PASSING SEQUENCES
-------------------------------------------------------------------------------
   generator is <./data/rngfile.txt>
-------------------------------------------------------------------------------
 C1  C2  C3  C4  C5  C6  C7  C8  C9 C10  P-VALUE  PROPORTION  STATISTICAL TEST
-------------------------------------------------------------------------------
  3   2   0   4   2   2   1   1   4   1  0.534146     20/20       Frequency
  2   1   2   0   3   2   0   5   2   3  0.350485     20/20       BlockFrequency
  3   1   1   3   0   3   5   2   1   1  0.350485     20/20       CumulativeSums
  3   2   1   2   2   2   2   2   3   1  0.991468     20/20       CumulativeSums
  2   0   1   4   5   1   3   1   2   1  0.275709     20/20       Runs
  2   0   4   2   1   0   1   2   5   3  0.213309     19/20       LongestRun
  0   1   2   8   2   1   2   2   1   1  0.008879     20/20       Rank
  4   2   3   2   2   1   3   2   1   0  0.739918     20/20       FFT
```