TEST REPORT

Jurisdiction:	Online Gaming
Authority:	N/A
Customer:	The Optimizer Investments Limited 3rd Floor, J&C Building Road Town Tortola, British Virgin Islands VG1110
Manufacturer:	The Optimizer Investments Limited
Submission Reference:	Submission received 7 December 2020
Evaluated Product:	Poker RNG
Issued By:	BMM Testlabs
Location(s) of Evaluation:	BMM Testlabs Level 3, 810 Whitehorse Road Box Hill, Victoria 3128 Australia
Project Number:	BGL.1002
Report Number:	BGL.1002.01
Date of Issue:	25 January 2021
Date(s) of Evaluation:	11 January 2021 to 15 January 2021
Standards Tested:	GLI-19: Interactive Gaming Systems Version 2.0
Compliance Certification:	BMM hereby certifies that the Poker RNG complies with the standards listed above.
Signed:	
	Christopher Van Prooije, Senior Systems Consultant, Mathematics

1. PURPOSE:

The Optimizer Investments Limited has requested BMM to evaluate the random number generator (RNG) used in the Poker RNG for operation in Online Gaming.

2. EVALUATION METHOD:

The evaluation was conducted using the BMM testing procedures designed to ensure compliance to the applicable technical standards and ISO/IEC 17025.

The full details of the tests are stored in the Quality Management System.

3. DESCRIPTION OF RNG:

The Poker RNG makes use of the Random.js library for node.js. Specifically, it uses the "nodeCrypto" engine, which is a wrapper for the node.js crypto.randomBytes call. This in turn draws bytes from the OpenSSL default RNG, which is a NIST SP 800-90A Rev. 1 compliant cryptographically secure RNG, using AES-256 in CTR-DRBG mode.

3.1 SOURCE CODE REVIEW:

The following sections describe the implementation of the RNG in the source code.

3.1.1 Seeding

The OpenSSL RNG seeds itself with entropy drawn from the system, which in turn uses numerous low-level hardware and software sources of entropy.

3.1.2 Cycling

The Poker RNG is cycled continuously in the background. The timing is randomised each time between 1 and 100ms, and draws between 2 and 10 random numbers each time for an average of around 138 times per second. The randomness of the cycling is provided by a separate RNG.

Additionally, the OpenSSL RNG updates its state regularly with entropy drawn from the system.

3.1.3 Scaling

Methods are provided by the Random.js library for providing random values in usable ranges from a uniform distribution without introducing bias. These methods are used for the shuffling algorithm (see below) to randomly draws cards from the deck.

3.1.4 Shuffling

The method for shuffling the deck correctly randomises the order of the cards uniformly, with each possible ordering equally likely.

4. FILE SIGNATURES:

The following file(s) are used by the RNG. Each signature is produced using the HMAC-SHA1 algorithm with an empty (0) key.

File(s)	Signature
poker.js	9EFF34E4DD76AB2CDB74D5F8F969FD61A6B0CD27
random-js\CHANGELOG.md	91A6973E02A88BEB5042CA41F0D0B12CB5E67EF3
random-js\LICENSE	2ABFD5B0657C5E74286D820F312A1DDF4DF51494
random-js\package.json	B9BC8B9DF5B6C3BAB43565B4C2E4C7AAF1FF37D8
random-js\README.md	03A123C3B0A7C86AA8B798BE85511F95C3BCC2AF
random-js\dist\index.d.ts	604D296FAEFDE60F9E3400DE1E574663FB94C7F2
random-js\dist\random-js.esm.js	76058E3E742696DEFA1127C6F9D34E66B1338149
random-js\dist\random-js.esm.js.map	7255C722D155C82691DBD5FED275AC5AD2B0E862
random-js\dist\random-js.umd.js	1AB025CD7E82101847FF97901A2AE3165D3D9861
random-js\dist\random-js.umd.js.map	AC37D5100329A072F09D09B4163C32CA4D56E341
random-js\dist\random-js.umd.min.js	46D18F164E4FF597D65E74E50527AA086122BAB5
random-js\dist\random-js.umd.min.js.map	51132FA8664A9EF35C14300C09660A599DC27A1E
random-js\dist\Random.d.ts	28BD23681C795FE36448FB2094F01A1F9483F25B
random-js\dist\types.d.ts	0294C487D43CAD7D9B85444B1E93D74D303F552B
random-js\dist\distribution\bool.d.ts	005DA4AC04AFA774D66E0369C9EEAE114D4A9FAC
random-js\dist\distribution\date.d.ts	12208339C7D0B7102A5AD4F43EA4AC7FDAAB9ED7
random-js\dist\distribution\dice.d.ts	37C443E9C5B7A3B870D2BC9B350180158450D3B3
random-js\dist\distribution\die.d.ts	F2FCFF2F1F9A6B6FABA281D716EE5489BC5BA366
random-js\dist\distribution\hex.d.ts	DDE6CD934CFDB4746101C5EE8F8E54A0FE6A57FD
random-js\dist\distribution\int32.d.ts	35BD544617AC364206DCB7A741438B56C28875B8
random-js\dist\distribution\int53.d.ts	FBFD4012AD0DDCFBAC710E2C3D177ABDCFFA55EF
random-js\dist\distribution\int53Full.d.ts	C45CFFA67DE26A3385EFD80E948FC713E56E841A
random-js\dist\distribution\integer.d.ts	26E4C2EDE883E03E928C961ECB52349FD127700B
random-js\dist\distribution\pick.d.ts	0B55B896E0D9B32ACF1ACCE6E0A0A8EC66131CBA
random-js\dist\distribution\picker.d.ts	EE0DE914812FE2C0A30892EEBA935B0298090AAC
random-js\dist\distribution\real.d.ts	9E292D988B5D320836D10F9C0751070D4C855B9A
random-js\dist\distribution\realZeroToOneExclusive.d.ts	F89E89E6D85C8CC0907C63F78B9C45DD320DFEA4
random-js\dist\distribution\realZeroToOneInclusive.d.ts	F2170E694922847D29D8CB861AB4A527BA9CA619
random-js\dist\distribution\sample.d.ts	EE87A512CA58DE0207118D71AF79B809B4C86CD6
random-js\dist\distribution\shuffle.d.ts	D5ABBDC370A3236BF29AB1649665F89E1CC67CBE
random-js\dist\distribution\string.d.ts	2D1276E1FACF9C7E355CD6F29C2ECEE47F1FA605
random-js\dist\distribution\uint32.d.ts	B1F4F33DACD1F5EF740F1017FFD8645AE16356C6
random-js\dist\distribution\uint53.d.ts	6393A04F6277340B682BC5E88A2DEF866CC49972
random-js\dist\distribution\uint53Full.d.ts	FF7630BCD8224DA40CCB30D38C1655F13A9C3700
random-js\dist\distribution\uuid4.d.ts	9DAE02B7F51FECFD49C4C76AF8DA6A330ECA8D08
random-js\dist\engine\browserCrypto.d.ts	3D24733415DF10F826E14506B937498585608AAE
random-js\dist\engine\MersenneTwister19937.d.ts	B804002A365ABCBEB46EAF977FF8117F02CF6E88
random-js\dist\engine\nativeMath.d.ts	24934E3E2E3FE58D38514B3DBF676C3EAD1D776C
random-js\dist\engine\nodeCrypto.d.ts	04C233DF9DE768776A033D8A466CF1068CF71242
random-js\dist\utils\add.d.ts	0577A01385D0166A20C77B93E3FC678A337F7767
random-js\dist\utils\constants.d.ts	D7CE195D9C100EEB22F4B093B2747C5D373EFAE6
random-js\dist\utils\convertSliceArgument.d.ts	17A7C78487719342ADF558A2A88485AC05BEAFA2
random-js\dist\utils\createEntropy.d.ts	9E801640AC00DF46917CE38519D2C3B53BA937F7
random-js\dist\utils\imul.d.ts	0C23964045A3E51CD6B69F72E6DA46CC5A005B07
random-js\dist\utils\Int32Array.d.ts	3F01C0BE4218D77138FC12F2E716DA7CCF3AF2E1
random-js\dist\utils\multiply.d.ts	2D35F99955D6C0CF2EA80CA8F3437C05D134CC8D

File(s)	Signature
random-js\dist\utils\sliceArray.d.ts random-js\dist\utils\stringRepeat.d.ts random-js\dist\utils\toInteger.d.ts	283ED98B192835CC337927784FF08E9F4D5D6AE1 4044829439E482A9CCF08F5A0B5D4100A92E99D6 2A062F4CF7302DB006A97D61177B72B39F0A25B9

5. TEST RESULTS:

Each test tests the hypothesis that the RNG is a random source of numbers. A "p-value" is produced for each test run, which is the probability that a truly random process would produce the same or a more extreme result. P-values are expected to be uniformly distributed between 0 and 1. Each test is performed at least 100 times, and the p-values for each test are evaluated using an Anderson-Darling test. This produces a single p-value, which is the probability that the individual p-values have been produced from a uniform distribution.

Finally, the p-values from each test in the same test suite are combined using the Holm-Bonferroni method to provide an overall p-value. This process adjusts each p-value to ensure that the overall probability of accepting the RNG as random matches the confidence interval used. The overall p-value, equal to the minimum of the adjusted p-values, is compared to a specific alpha value to determine if the RNG is accepted or rejected as being random for a specific confidence interval. For a 99% confidence interval, the alpha value used is 0.01.

The following tables summarise the test results. See Appendix A for a description of the statistical tests used.

5.1 EMPIRICAL TESTS:

Test	P-values	99% Confidence
Frequency Test	1.000000	PASS
Serial Correlation Test	1.000000	PASS
Runs Test	1.000000	PASS
Gap Test	1.000000	PASS
Coupon Collector Test	1.000000	PASS
Subsequences Test	1.000000	PASS
Poker Test	1.000000	PASS
Overall	**1.000000**	**PASS**

Conclusion: The RNG is **ACCEPTED** as random at the 99% confidence interval.

5.2 DIEHARD TESTS:

Test	P-values	99% Confidence
Binary Rank 32x32 Test	1.000000	PASS
Binary Rank 6x8 Test	1.000000	PASS
Birthday Spacings Test	1.000000	PASS
Bitstream Test	1.000000	PASS
Count The 1's Stream Test	1.000000	PASS
Count The 1's Specific Test	0.280576	PASS
Runs Test	1.000000	PASS
Squeeze Test	1.000000	PASS
Overall	**0.280576**	**PASS**

Conclusion: The RNG is **ACCEPTED** as random at the 99% confidence interval.

5.3 NIST TESTS:

Test	P-values	99% Confidence
Approximate Entropy Test	1.000000	PASS
Block Frequency Test	1.000000	PASS
Cumulative Sums Test	1.000000	PASS
Discrete Fourier Transform Test	1.000000	PASS
Frequency Test	1.000000	PASS
Linear Complexity Test	1.000000	PASS
Longest Run of Ones Test	1.000000	PASS
Non-Overlapping Template Matchings Test	1.000000	PASS
Overlapping Template Matchings Test	1.000000	PASS
Random Excursions Test	1.000000	PASS
Random Excursions Variant Test	1.000000	PASS
Rank Test	1.000000	PASS
Runs Test	1.000000	PASS
Serial Test	1.000000	PASS
Universal Test	1.000000	PASS
Overall	**1.000000**	**PASS**

Conclusion: The RNG is **ACCEPTED** as random at the 99% confidence interval.

6. NON-COMPLIANCES:

None.

7. CONDITIONS:

None.

8. ADDITIONAL INFORMATION:

If the above Poker RNG requires any parameters in order to configure it for operation they must be those which are specified by the manufacturer and must comply with the jurisdictional/operational requirements.

9. CONCLUSION:

As a result of statistical testing and source code review, BMM believes that the Poker RNG provides uniformly random data suitable for its intended application. This RNG complies with the applicable requirements for operation in Online Gaming.

10. TERMS AND CONDITIONS:

BMM Testlabs ("BMM") has conducted a level of testing of the gaming product which has historically been adequate for a submission of this type. However, inherent in testing in a laboratory environment are the unavoidable limitations of not being able to verify the effects of all possible configurations and environments that occur in actual gaming venues.

This test report is for use by the client for the jurisdiction ("Jurisdiction") referenced in the report (the "Report") and only verifies, as of the date stated, the gaming product described in the Report subject to any conditions or limitations set forth therein.

The manufacturer named in the Report is solely responsible for possession of the appropriate license to sell, lease, service, or provide gaming supplies or gaming-related services in the Jurisdiction and for compliance with the ongoing requirements of the Jurisdiction. It is the responsibility of the manufacturer and operators to ensure that the gaming product detailed in this Report is installed, maintained and operated correctly without defects and safely in accordance with requirements of the Jurisdiction.

The Report and testing performed by BMM is proprietary to BMM. This Report is issued solely for the benefit of the client and shall not be reproduced, reprinted, or transmitted in whole or in part to any party not named in the Report without the written approval of BMM, other than by a regulator of the Jurisdiction. No third party may use, rely, or refer to the Report, its contents, or any related documents, without written permission of BMM. If BMM grants consent, BMM will send this Report via email as directed. BMM takes precautionary measures to secure the "PDF" document, but BMM does not send the email via any encrypted methodology.

The undersigned certifies under penalty of perjury that the compliance testing of the gaming product detailed in this Report and any accompanying documents was conducted in accordance with the requirements of the Jurisdiction and that the gaming product meets the requirements of its laws and the regulations adopted thereunder, and all published technical standards, control standards, control procedures, policies, industry notices and similar requirements implemented or issued by the Jurisdiction to the best of BMM's knowledge and belief.

Notwithstanding the above, any regulator may reprint, reproduce and transmit any document or information to any party that the regulator, in their sole discretion, deems appropriate.

BMM DOES NOT MAKE, AND EXPRESSLY DISCLAIMS, ALL OTHER WARRANTIES OF ANY KIND, EXPRESSED OR IMPLIED, INCLUDING WITHOUT LIMITATION ANY WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, SUITABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. THE LIABILITY AND OBLIGATIONS OF BMM HEREUNDER, AND THE REMEDY OF THE RECIPIENT, UNDER OR IN CONNECTION WITH THIS AGREEMENT SHALL BE LIMITED TO, AT BMM'S OPTION, REPLACEMENT OF THE SERVICES PROVIDED OR THE REFUND BY BMM OF ANY MONIES RECEIVED BY IT FOR THE SERVICES PROVIDED. IN NO EVENT SHALL BMM BE RESPONSIBLE TO THE CLIENT OR ANY THIRD PARTY FOR ANY CONSEQUENTIAL, INCIDENTAL, DIRECT, INDIRECT, OR SPECIAL DAMAGES, INCLUDING WITHOUT LIMITATION DAMAGES FOR LOST PROFITS OR REVENUE, BUSINESS INTERRUPTION, OR PUNITIVE DAMAGES, EVEN IF BMM HAD BEEN ADVISED OF THE POTENTIAL FOR SUCH DAMAGES.

APPENDIX A. STATISTICAL TESTS

The following tests were used to test the statistical properties of the RNG.

A1. EMPIRICAL TESTS

The Empirical Tests are based on the tests described by Donald Knuth in The Art of Computer Programming Volume 2: Seminumerical Algorithms (1968, revised in 1997). They test sequences of numbers scaled to specific ranges.

Frequency Test	Counts of each number occurring across the sample set.
Serial Correlation Test	Counts of non-overlapping groups of numbers occurring together. Group sizes of two, three, and four are tested separately.
Runs Test	Counts of ascending and descending sequences of numbers. Note that this is a different test to the Runs Test in the Diehard and NIST Tests.
Gap Test	Counts of the size of gaps between successive occurrences of a given number. Each number in the range is tested separately.
Coupon Collector Test	Counts of sequence lengths required to complete a full set of each number in the range.
Subsequences Test	Similar to the Serial Correlation Test for pairs of numbers, except looking at numbers separated by a specific gap. Step sizes of 5, 10, 15, and 20 are tested separately.
Poker Test	The sequence is split into groups of five. The number of unique values in each group is counted.

A2. DIEHARD TESTS

The Diehard Tests are based on the test suite published by George Marsaglia in 1995. They test sequences of raw binary output from the RNG.

Binary Rank 32x32 Test	Matrices are created using 32 32-bit words. The ranks of the resulting matrices are counted.
Binary Rank 6x8 Test	Same as the Binary Rank 32x32 Test, except each matrix is formed using 6 values, each taking 8 bits from successive 32-bit words with a specific offset. All possible offsets are tested separately.
Birthday Spacings Test	26-bit values are taken from successive 32-bit words with a specific offset. The values are sorted, and the spacings between them calculated. The number of spacings of the same size are counted. All possible offsets are tested separately.
Bitstream Test	Blocks of 2^18 values are treated as a stream of overlapping 20-bit values. The number of possible 20-bit values that are not found in each block is counted.
Count The 1's Stream Test	8-bit values are taken and assigned a “letter” based on the number of one’s appearing in the binary representation of each value. Overlapping groups of 5 “letters” are counted.
Count The 1's Specific Test	Similar to the Count The 1’s Stream Test, except 8-bit values are taken from successive 32-bit words with a specific offset. All possible offsets are tested separately.
Runs Test	Counts sequences of increasing and decreasing 32-bit words. Note that this is a different test to the Runs Test in the Empirical and NIST Tests.
Squeeze Test	A value of 2^31 is repeatedly multiplied by 32-bit words, dividing by 2^32 and taking the ceiling of the result each time. The number of successive words that are required to reduce the value down to 1 is counted. The value is reset to 2^31 and the process is repeated.

A3. NIST TESTS

The NIST Tests are based on the suite of tests released by the National Institute of Standards and Technology in Special Publication 800-22, Revision 1a (revised April 2010). They test sequences of raw binary output from the RNG.

Approximate Entropy Test	Similar to the Serial Test, count each possible m-bit value, except it does so for two adjacent m bit lengths and compares the two.
Block Frequency Test	Similar to the Frequency Test, except the data is split into equally sized blocks. The number of ones and zeroes in each block is counted.
Cumulative Sums Test	Random walks are created by converting the data to +1 / -1 for 1 / 0 respectively and summing consecutive values.
Discrete Fourier Transform Test	The data is transformed using a Discrete Fourier Transform. The number of peaks within the 95% threshold are counted.
Frequency Test	The number of ones and zeroes in the binary output is counted.
Linear Complexity Test	The length of the linear complexity of the random sequence is determined.
Longest Run of Ones Test	The data is split into equally sized blocks. The longest run of ones in each block is determined and counted.
Non-Overlapping Template Matchings Test	The data is split into equally sized blocks. Each block is searched for a specific pattern of bits and counted. A separate test is run for various bit patterns. Each bit pattern searched does not overlap with itself. That is, when the pattern is matched, the end of the pattern cannot be the start of another match.
Overlapping Template Matchings Test	Similar to the Non-Overlapping Template Matchings Test, except only one pattern is searched, which may overlap with itself.
Random Excursions Test	As with the Cumulative Sums Test, random walks are created by converting the data to +1 / -1 for 1 / 0 respectively and summing consecutive values. The number of times a given state is visited between returns to zero are counted. Separate tests are run for various states from -4 to +4, not including 0.
Random Excursions Variant Test	Similar to the Random Excursions Test, except the number of times the given state is visited is counted for the entire sequence. Separate tests are run for various states from -9 to +9, not including 0.
Rank Test	Matrices are created using 32 32-bit words. The ranks of the resulting matrices are counted. Note that this is fundamentally the same test as the Binary Rank 32x32 Test in the Diehard Tests, although the implementation may differ.
Runs Test	Runs of consecutive bits of the same value of various lengths are counted.
Serial Test	Counts of each possible m-bit values. Separate tests are run for various m bit lengths.
Universal Test	Distances between repeated patterns of bits are counted.

--- END OF REPORT ---