NIST/SP 800-22 and GM/T 0005-2012 Tests: Clearly Obsolete, Possibly Harmful

Markku-Juhani O. Saarinen
PQShield Ltd. Oxford, UK
mjos@pqshield.com

***Abstract*—When it comes to cryptographic random number generation, poor understanding of the security requirements and "mythical aura" of black-box statistical testing frequently leads it to be used as a substitute for cryptanalysis. To make things worse, a seemingly standard document, NIST SP 800-22, describes 15 statistical tests and suggests that they can be used to evaluate random and pseudorandom number generators in cryptographic applications. The Chinese standard GM/T 0005-2012 describes similar tests. These documents have not aged well. The weakest pseudorandom number generators will easily pass these tests, promoting false confidence in insecure systems. We strongly suggest that SP 800-22 be withdrawn by NIST; we consider it to be not just irrelevant but actively harmful. We illustrate this by discussing the "reference generators" contained in the SP 800-22 document itself. None of these generators are suitable for modern cryptography, yet they pass the tests. For future development, we suggest focusing on stochastic modeling of entropy sources instead of model-free statistical tests. Random bit generators should also be reviewed for potential asymmetric backdoors via trapdoor one-way functions, and for security against quantum computing attacks.**

***Index Terms*—TRNG, Entropy Sources, SP 800-22, GM/T 0005-2012, Statistical Randomness Tests, Stochastic Models**

1. Introduction

In traditional "black-box" statistical testing for randomness, no analysis of the *process* of random number generation is performed. These tests were mainly motivated by the requirements of numerical methods in applied sciences. The approach predates the invention of digital computers, information theory, and modern cryptography.

In the late 1930s, Kendall and Babington Smith [KBS38], [KBS39] proposed four tests of randomness; the *frequency test*, the *serial test*, the poker test, and the *gap test*.

The famous RAND tables [RAN55] were generated via electronic means in 1947, and their statistical testing was performed with IBM electromechanical computers [Bro49]. The four main tests used for assessment are very similar to those used by Kendall and Babington Smith.

Essentially the same four tests were adopted for FIPS 140-1 standard in 1994 (monobit test, poker test, runs test, and long runs test). They were removed only in FIPS 140-2 Change Notice 2 in 2001 [NIS01].

At least from 2001, the AIS 20/31 [KS01], [KS11] evaluation documents from German BSI have also addressed the security of physical random number generators, not just their statistical qualities.

1.1. Cryptographic RNG Evaluation

We recall some of the main security assurance aspects of random number generators used in cryptography. The current standards in the SP 800-90 series [BK15], [TBK+18], [BKR+21] and AIS-31 [KS11] already cover many of these topics.

Pseudorandom Generator Evaluation. The process of assessing the security of a pseudorandom number generator is entirely analogous to that of other cryptographic modes and constructions. Rather than reinventing and re-evaluating symmetric cryptography algorithms, one generally wishes to demonstrate (via mathematical proofs) that breaking a DRBG (Deterministic Random Bit Generator) implies a break of an underlying vetted cryptographic algorithm such as AES or SHA-2/3 [BK15].

Pseudorandom Implementation Validation. A statistical test is a poor way of verifying that a deterministic algorithm has been correctly implemented. In reality, one would test DRBGs with chosen inputs and utilize Known Answer Tests (KATs) with reference tests vectors. This is one of the things that is done in NIST's CAVP[1].

A standard practice with hardware modules is to use formal methods to validate the equivalence of the implementation against an abstract mathematical model. Formal verification is increasingly being applied to software implementations as well.

Entropy Source Evaluation. The security of physical entropy sources is mainly assessed via design review and testing of entropy production processes (stochastic models). Robustness of implementation can be increased with start-up tests and continuous monitoring of noise sources with health tests [TBK+18]. The health monitoring should be tailored to the generator. Access to the raw noise is necessary to validate the actual entropy content and the correctness of stochastic models (See Section 3).

Physical / Interface Security Evaluation. Since cryptographic random number generators are ultimately the source of all secret key material, their physical security and interface links must be robust and protected from monitoring and interference. In many practical applications, the generator must be in the same physical enclosure as the other cryptographic components. The key lifecycle (generation, use, destruction) can happen entirely within the security boundary of that module.

1. NIST Cryptographic Algorithm Validation Program https://csrc.nist.gov/projects/cryptographic-algorithm-validation-program

1.2. Current Standards

This work focuses on the NIST SP 800-22 "black-box" tests, initially published in October 2000 and revised as recently as 2010 [RSN+10].

FIPS 140-3. Fortunately, SP 800-22 is no longer used in NIST's own Deterministic Random Bit Generator (DRBG) and Entropy Source validation processes [NC21, Annex D.J]. FIPS 140-3 [NIS19] requires compliance with significantly more robust SP 800-90A [BK15] and SP 800-90B [TBK+18] standards instead.

National Security Systems (NSS). SP 800-90B also forms the basis for NIAP Entropy Assessment Reports (EAR) [NIA13a], [NIA13b] used in U.S. Government NSS. SP 800-90B is explicitly mentioned in current NIAP Common Criteria Protection Profiles such as [NIA22].

Secure Electronics. Another widely used Common Criteria evaluation methodology is AIS-20/31 [KS11] from German BSI. This evaluation process includes stochastic modeling as well as a design review in addition to an array of statistical tests. It is referenced in protection profiles of random number generators for Trusted Execution Environments (TEE) [Glo20], and Secure Elements (SE) [Glo21] such as smartcards.

Elsewhere. Especially before the ratification of SP 800-90B in 2018, the AIS-31 was considered a global "golden standard" for cryptographic random numbers. AIS-31 was even translated to Russian and published by the official Technical Committee for Cryptography Standardization (TC26) for the Russian Federation [TC218].

2. Obsoleted but Still Harmful: SP 800-22

Unfortunately, the results of black-box randomness test suites such as Dieharder [BEB03], Fourmilab ENT [Wal98], and SP 800-22 [RSN+10] are still being presented as sole evidence of the suitability of Random Number Generators for cryptographic applications.

The situation is made worse by the active status of NIST Special Publication 800-22 Rev. 1a, *"A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications,"* [RSN+10] even though it has no actual role in security evaluation.

In January 2022, it was announced – against the expectations of many cryptographers in the CMUF Entropy Working Group and elsewhere – that the document is not withdrawn but will be revised instead[2].

There are many recent examples of commercial and academic usage of SP 800-22 in a cryptographic context. For instance, in [MKAG20], the authors use SP 800-22 as evidence of the suitability of DNA synthesis for random number generation. No entropy assessment or physical (interface) security analysis is performed. The actual evaluation standards are not even cited; mainly just other generators with no proper security analysis (See Fig. 1.)

2.1. The Tests: Cryptanalysis-in-a-Box?

SP 800-22 Rev. 1a [RSN+10] contains descriptions of 15 statistical tests, which have also been implemented in a software package available from the NIST website[3]. We note that the current Chinese standard GM/T 0005-2012 [SCA12] contains a very similar set of 15 tests. The Chinese standard is also coming into a review, with a revised standard GM/T 0005-2021 coming into effect in May 2022 (the changes in the revised Chinese standard are unknown to the author at the time of writing.)

The 15 tests take in a sequence of output bits, typically 1,000,000 bits, and produce P values based on that information, which leads to a PASS/FAIL metric.

1. Frequency, monobit.
2. Block Frequency.
3. Runs Test.
4. Longest Run.
5. Binary Matrix Rank.
6. Fourier (Spectral).
7. Non-overlap match.
8. Overlapping Match.
9. Maurer's Universal.
10. Linear Complexity.
11. The Serial Test.
12. Approx. Entropy.
13. Cumulative Sums.
14. Random Excursions.
15. Rand. Excursions 2.

2.2. A Systemic Problem

The SP 800-22 tests are based on a purely statistical interpretation of uniform and independent randomness [RSN+10, Sect. 1.1.1]. The definitions and stated goals of the tests do not relate to computational indistinguishability [KL14, Sect. 7] or other relevant cryptographic security notions. Randomness is not viewed from a cryptanalytic security perspective. In other words, the question of how hard it is to "break" the random and pseudorandom generators is not considered.

The word "cryptanalysis" can be found in the abstract of SP 800-22 three times, but not once in the body of the publication. Some of the included tests have a cryptanalytic flavor, however.

As an illustrative example, the "linear complexity test" is motivated by statement *"An LFSR that is too short implies non-randomness."* That may be true with some non-cryptographic notions of randomness, but every cryptanalyst knows that a plain LFSR is never a secure pseudorandom generator. Since LFSRs are linear, it is a simple numerical exercise to solve the internal state of a fixed LFSR from the output. The linear algebra required runs in polynomial time; hence plain LFSRs are cryptanalytically broken regardless of their length.

While the SP 800-22 tests can potentially be used in elementary cryptanalysis, just as any other analytic method, it is not usually considered relevant evidence in mainstream cryptographic literature. As discussed in Section 4, the "reference generators" described in the SP 800-22 document itself pass these tests while generally lacking security.

2.3. Perhaps It's For Implementation Bugs?

The SP 800-22 section "Testing Strategy and Result Interpretation" [RSN+10, Sect. 4] suggests that the 15 statistical tests are applied to random bit generators based on cryptographic hash functions and block ciphers. No

2. January 12, 2022: Announcement of Proposal to Revise Special Publication 800-22 Revision 1a https://csrc.nist.gov/News/2022/proposal-to-revise-sp-800-22-rev-1a

3. Visited February 10, 2022: "NIST SP 800-22: Download Documentation and Software." https://csrc.nist.gov/projects/random-bit-generation/documentation-and-software

Table 1 Selection of random number generators, the underlying generation methods and randomness production rate in MB/s.

Random number generator	Randomness production rate [MB/s]	Method
Meiser et al.	0.3	DNA synthesis
Gaviria Rojas et al.	Not available	Solution-processed carbon nanotubes
Lee et al.	0.025	Crystallization robot analyzing chemical processes
Reidler et al.	1560	Chaotic semiconductor laser
HotBits	0.0001	Timing successive pairs of radioactive decays
Random.org	0.0015	Entropy from atmospheric noise
Lavarand	0.02	Patterns photographed off floating material in lava lamps
Intel digital random number generator	800	Processor resident entropy source to seed hardware-implemented entropy from atmospheric noise
Mersenne Twister	15,000	Pseudo-random number generator: algorithm using polynomial algebra

Figure 1. Reproduced from Meiser et al. [MKAG20]. There exists a colorful "parallel universe" of ad hoc random number generators. These often ignore physical and interface security altogether and use black-box statistical test suites such as NIST SP 800-22 as evidence of security. The Intel DRNG forms an exception in the Meiser "references" as it is uses the current NIST SP 800-90 standards (its description is not accurate, however).

new or useful statistical features can be found in the output of standard cryptographic algorithms with these tests. Such algorithms pass these tests even without any secret seeding material (and hence, with no security).

The only sensible motivation for using the tests on the output of vetted cryptographic algorithms would be to discover flaws in their implementation. However, a buggy RNG should not be used, even if the bugs are so minor that the output still passes these trivial tests. It is much more useful to validate that a cryptographic hash function or block cipher is implemented according to the standard than to verify their statistical qualities.

We note good physical entropy sources are generally not expected to yield strictly uniform output and would fail these tests; that is why more appropriate stochastic models are used instead. FIPS 140-3 generally forbids the direct use of physical entropy sources without cryptographic post-processing (conditioning) [BKR+21].

3. Entropy Sources: Stochastic Models

There is a statistical aspect of random bit generation that a potential replacement to SP 800-22 could address: Entropy Source Stochastic Models. While stochastic models are mentioned and suggested (they're a "may" not a "should") in SP 800-90B [TBK+18, Sect 3.2.2], the creation and requirements of stochastic models are not sufficiently addressed in the current NIST (SP 800) publications.

Stochastic models are a common requirement in the AIS-20/31 [KS11] evaluation methodology from German BSI. It would benefit vendors and testing labs if the NIST and BSI requirements were further harmonized. The definition of a stochastic model for an entropy source (or a "TRNG" in AIS-20/31) is already very similar in the two documents.

> *A stochastic model is a mathematical description (of the relevant properties) of an entropy source using random variables. A stochastic model used for an entropy source analysis is used to support the estimation of the entropy of the digitized data and finally of the raw data. In particular, the model is intended to provide a family of distributions, which contains the true (but unknown) distribution of the noise source outputs. Moreover, the stochastic model should allow an understanding of the factors that may affect the entropy. The distribution of the entropy source needs to remain in the family of distributions, even if the quality of the digitized data goes down. [TBK+18, Appendix B, Page 65]*

In practice, one studies the noise source and identifies the stochastic processes that generate entropy. Understanding the entropy process allows a stochastic model, output distributions, and min-entropy estimates to be developed. It also helps in deriving failure threshold parameters for the start-up tests and continuous health tests. A stochastic model can also have environmental components (e.g., temperature, voltage, interference metrics).

The statistical hypothesis testing in entropy source validation tries to answer the question: Does the physical entropy source behave as predicted by its stochastic model (and produce the expected amount of entropy) ? Physical entropy sources rarely have a stochastic behavior of white noise. Note that entropy analysis can't be performed after post-processing as such steps generally introduce pseudo-randomness; this is why access to raw noise sources is generally required (and NIST has been wise to require this interface in NIST SP 800-90B). Entropy sources with non-uniform (but well-understood) stochastic models are equally useful if they provide real entropy and have implementation advantages (size, power, manufacturing cost, physical robustness).

4. All the bad "Reference Generators"

Appendix D of SP 800-22 contains descriptions of *"Reference Pseudorandom Number Generators."* It is unclear what the purpose of these generators is, apart from testing the proposed statistical suite itself. Many of these generators are insecure yet pass the tests, conveniently demonstrating the pitfalls of black-box statistical testing.

4.1. Basic Design Review

In addition to the standard high-level considerations of statistical inconspicuousness and (enhanced) forward and backward security (See AIS-20/33 [KS11, Sect 2.2.2].) we also use this to illustrate certain additional features of secure pseudorandom number generators (DRBGs) that

can be assessed in a design review: lack of backdoors and security against a quantum adversary.

4.1.1. Computational Indistinguishability. The RNG output should be computationally indistinguishable from random. This is a cryptanalytic part of the design review; random number generators are not exempt from Kerchoff's principle.

Security bounds can be set for both data and computational complexity, e.g., 2^{128} bits of output and computational requirements equivalent to breaking AES-256 in both classical and quantum computational settings.

This requirement implies that the determination of the internal state from DRBG output should be hard. Another implication is that the generator has a sufficiently large private internal state. Due to time-memory tradeoff attacks, the state is usually larger than the security level (e.g., twice as large).

Analysis: All generators in Appendix D fail this task; either trivial distinguishers can be built directly from output, or the exposure of the internal seed leads to distinguishers.

4.1.2. Sufficient start-up entropy. The internal state must be seeded with sufficient entropy before it can be used to produce output.

Analysis: Seeding is not addressed by SP 800-22.

4.1.3. Secure Reseeding. The mechanism for adding entropy (reseeding) must be secure against adaptive chosen input attacks. One can adapt IND-CPA and IND-CCA security models for this purpose.

Analysis: Reseeding is not addressed by SP 800-22.

4.1.4. Quantum Safety. The use of primitives that rely on the hardness of problems for which fast quantum attacks exist (e.g., factoring and the elliptic curve discrete logarithm problem [Sho94]) are not suitable for use with quantum-safe cryptography. However, there is no need to use "quantum" random number generators to achieve quantum safety in post-quantum cryptography; symmetric cryptography is generally safe.

Analysis: The BBSG and MSG generators explicitly fail in this aspect, possibly MODEXP too.

4.1.5. Absence of backdoors (and trapdoors). The existence of any "public key trapdoor" function in a place where secure symmetric cryptography could be used is often an indication that the generator can be used to instantiate a covert channel or a backdoor. Pseudorandom number generation and conditioning only require trapdoor-free one-way functions – such as *symmetric* cryptography primitives. While a failure to protect the private state effectively broadcasts secrets, "unique-access" backdoors generally require public-key trapdoor functions. With such a backdoor, a third party can choose the public instantiation parameters so that a "private key" allows the secret state to be covertly determined from DRBG output.

Analysis: The BBSG and MSG generators allow this, possibly MODEXP too.

4.2. Linear Congruential Generator (LCG)

The first "reference generator" in Appendix D is a multiplicative congruential generator, attributed to Fishman and Moore in the text, but really proposed by D. H. Lehmer in 1949 [Leh51] – Lehmer even proposed the specific Mersenne prime modulus $p = 2^{31} - 1$ used. Multiplicative generators can be seen as a subset of linear congruential generators with the addition constant set to zero, although their analysis is slightly different.

Description and implementation of LCG. Section D.1 offers a definition for a multiplicative generator as $z_{i+1} = a * z_i \bmod (2^{31} - 1)$, without stating what the a value is. The text asserts that "a is a function of the state," which is clearly incorrect. Examination of the code and outputs reveals it to be $a = 950706376$, which leads to maximum $(2^{31} - 2)$ period.

The code (in `src/generators.c`) is a literal, line-by-line translation to C of the 1980s era Fortran code written by L. R. Moore of RAND corporation. The original can be found in [Fis96, Fig. 7.3, p. 604]. It is remarkable that the reference code implements the Lehmer generator with five double-precision multiplications, four full divisions, three calls to the `floor()` function, and some additional arithmetic. However, this is equivalent:

```
z = (z * 950706376) % 0x7fffffff;
```

Note that the $(\bmod\ p)$ reduction can be easily implemented with a shift and an add since $2^{31} \equiv 1 \pmod{p}$. No division is required, just a single 31×31-bit multiply. Perhaps their double-precision approach made sense with a specific 1980s computer without an integer multiplier, but we doubt it. We noticed the code does not work correctly under more aggressive floating-point optimization levels; it is dependent on specific IEEE 754 floating-point properties in a very "fragile" manner.

Weaknesses. Even if we ignore the cryptanalytically trivial 31-bit state size, a generator of this type can be rapidly attacked and distinguished from random [FHK+88].

4.3. Quadratic Congruential Generator I

The quadratic generator QCG-I is characterized by iteration $x_{i+1} = x_i^2 \pmod{p}$, a 512-bit p, and starting point x_0. The values x_i are used directly as 512-bit blocks.

Weaknesses. The generator outputs its entire state. From x_i it is trivial to compute x_{i+1} and also the two square roots $\pm x_{i-1}$. Hence the generator has neither forward nor backward security.

Statistically, the most significant bit of each 512-bit block will be biased by $2^{-512}p \approx 0.5956$ since $x_i < p$. The test suite does not seem to detect it, however. The bias leads to a trivial, low-complexity statistical distinguisher against the generator.

4.4. Quadratic Congruential Generator II

The QCG-II generator is based on iteration $x_{i+1} = 2x_i^2 + 3x_i + 1 \pmod{2^{512}}$.

Weaknesses. Entire 512-bit blocks are output, including the least significant bits. The generator allows both prediction and backtracking.

QCG-II does not have good statistical properties since there is no information flow from high-order bits towards the lower-order bits. The least significant bits $x_i \bmod 2^m$ will form a cycle with a period of at most x^m. For example, the least significant bit of each 512-bit block alternates in each block. The property leads to a trivial, low-complexity distinguisher against the generator.

4.5. Cubic Congruential Generator (QCG)

The iteration used in GCQ is $x_{i+1} = x_i^3 \pmod{2^{512}}$. We observe that this is equivalent to $x_i = x_0^{3^i} \pmod{2^{512}}$. The entire variable $\{x_i\}$ is output.

Weaknesses. The output blocks x_i can be reduced to a smaller modulus size 2^m for analysis. We can observe, for example, that the least significant byte satisfies $x_i \equiv x_{i+16} \pmod{2^8}$; the cycle is only 16. As a result, there will be an equivalent byte repeating with distance $16 * (512/8) = 1024$ bytes, which should be detectable purely statistically – but is not. These properties lead to trivial, low-complexity distinguishers against the generator.

4.6. Exclusive OR Generator (XORG)

The generated sequence starts with an initial value for $x_1, x_1, \ldots, x_{127}$ and applies the recurrence $x_i = x_{i-1} \oplus x_{i-127}$ for $i \geq 128$ to generate bits.

It is not explained that XORG implements an LFSR with an irreducible generator polynomial $x^{127} + x^{126} + 1$. Indeed its cycle $2^{127} - 1$ is prime too.

Weaknesses. There is no secret state with x_i, and hence the XORG / LFSR has no cryptanalytic security – despite attractive statistical properties and a long period. One can trivially both predict and backtrack outputs. A low-complexity attack models the LFSR as a system of linear equations over $\mathsf{GF}(2)$ and is able to distinguish with a high degree of certainty even it from random even if a continuous "block" of output bits are not available.

4.7. Modular Exponentiation Generator

The MODEXP generator is loosely based on the old Digital Signature Standard (DSS). One sets $x_1 = g^{\text{seed}} \mod p$ and $x_{i+1} = g^{y_i} \mod p$ for $i \geq 1$ where $y_i = x_i \bmod 2^{160}$. The output sequence consists of concatenated x_i values.

Weaknesses. Since full output blocks x_i are available, the sequence has no security. It can also be also trivially distinguished from random since the values satisfy $x_i < p$.

Some parameters are given, but not those that would be needed to assess the statistical quality of the generator. The (p, g) values are the same as the (p, x_0) values for QCG-1. We note that the order of the generator g is not given; if it is very small, then only a small number of different x_i values would be generated.

Checking for backdoors. To see how the backdoor could have been created into a generator of this type, we examine the factorization of the multiplicative order $p-1$. Since the order of g was not given, it was left to the author to discover the factorization[4]:

$$p - 1 = 2^5 * 11 * 47 * 151 * 175916087 * 22940963671 * p_{160} * p_{269},$$

where the two largest prime components are

$$p_{160} = 1213137149285565671196618904624637288561542502557, \quad \text{and}$$

$$p_{269} = 65296826399054038103394881313031787903702201432810974248631298387656923566051172 1.$$

We find that p_{160} is the order of the value g presented: $g^{p_{160}} \equiv 1 \pmod{p}$. A shorter cycle is more likely, with the expected size being $O(\sqrt{n})$ where n is the minimum of 2^{160} and multiplicative order of g.

It is also trivial to find weaker generators. For example,

$$h = \texttt{31fbe4d542ad14935055b18465de2a19}$$
$$\texttt{d261d3fd0625c565d1e17dceaebc479e}$$
$$\texttt{860bcb11a6d6697a0b1fd7172567600a}$$
$$\texttt{8808df985e2c88379bcb3a565db805e4}$$

satisfies $h^{11} \equiv 1 \mod p$ and hence $g = h$ could generate at most 11 different 512-bit x_i values.

4.8. Secure Hash Generator (G-SHA1)

SP 800-22 does not actually describe this generator but suggests looking at [MvOV96, Chap 5, P. 175]. An investigation of the code confirms that actual SHA-1 is not used, but a variant G with no message padding. The comment in STS 2.1.2, line 370 of `src/generators.c` states: *"This is the generic form of the generator found on the last page of the Change Notice for FIPS 186-2"*[5].

We note that this 186-2 generation method was later withdrawn as it produced biased random numbers (rejection sampling was not used, just reduction $\bmod q$).

The "generic form" (implemented in the code but not explained in the documentation) sets $y_0 = \mathit{Xkey}$ and iterates for $x \geq 1$:

$$x_i \leftarrow G(t, y_{i-1})$$
$$y_i \leftarrow y_i + x_i + 1 \bmod 2^{160}$$

Where $G(\mathsf{IV}, M)$ is a SHA-1 variant with start state $t = \mathsf{IV}$ and message M with (nonstandard) zero padding. The x_i values are output. This generator prevents the direct observation of the counter state y_i.

Weaknesses. Since it is not an actual counter mode, it will exhibit a cycle length of roughly $O(\sqrt{N}) \approx 2^{80}$. The security of G-SHA1 is also affected by weaknesses in the SHA-1 compression function. The SHA-1 algorithm was cryptanalytically broken in 2005 [WYY05] and depreciated by NIST in 2011 [BR11]. Many types of full, practical attacks have been demonstrated [SBK+17].

4. The smaller factors were discovered on a personal laptop in a few minutes using GMP-ECM Elliptic Curve Method (ECM) factorization software. The remaining 429-bit composite was factored into $p_{160} * p_{269}$ in about 95 minutes of computation with a 2x16-core (64-thread) AMD EPYC 7302 server using CADO-NFS Number Field Sieve (NFS) software.

5. An apparent reference to "FIPS Publication 186-2 (with Change Notice 1)", dated October 5, 2001. https://csrc.nist.gov/csrc/media/publications/fips/186/2/archive/2001-10-05/documents/fips186-2-change1.pdf

4.9. Blum-Blum-Shub (BBSG)

The name refers to [BBS86]. The system parameters $p, q \equiv 3 \pmod 4$ are primes and $n = pq$ is their product. The generator iterates $s_i \leftarrow s_{i-1}^2 \pmod n$ for $i \geq 1$ from an initial seed s_0. The least significant bit $x_i \leftarrow s_i \bmod 2$ forms the random sequence.

The input parameters are fixed in the code; each p, q is 512 bits, and s is 509 bits. Note that such fixing these parameters is not necessary if a module supports key generation.

Weaknesses. Since prime generation is slow and knowledge of the factorization, the composite n is often a fixed system parameter. Such a generator can allow access by those who know the fixed secret factors p and q of n.

A backdoor of this type was included in the SP 800-90A random bit generator specification for a while[6]. The `Dual_EC_DRBG` method was based on the public key cryptography of elliptic curves.

The quadratic residuosity and factoring problems underlying BBS are, of course, vulnerable to Shor's quantum algorithm [Sho94]. Hence BBSG is not quantum-safe.

4.10. Micali-Schnorr Generator (MSG)

The name refers to [MS88], [MS91]. The MSG generator is not described in detail in [RSN+10, Appendix D.9]. The STS 2.1.2 code has fixed 512-bit prime factors p and q, and sets the exponent as $e = 11$. A fixed 186-bit seed x_0 is set as well. The output sequence consists 837-bit lower-order chunks z_i, while the high 187 bits are used for the next exponentiation.

$$\begin{aligned} y_i &\leftarrow x_{i-1}^e \bmod n \\ x_i &\leftarrow \lfloor y_i / 2^{837} \rfloor \\ z_i &\leftarrow y_i \bmod 2^{837} \end{aligned}$$

Weaknesses. The security of the Micali-Schnorr scheme relies on the difficulty of factoring n and some additional statistical assumptions. It is vulnerable to Shor's quantum computing attack. The generator requires modular exponentiation and is therefore relatively slow.

A Micali-Schnorr generator `MS_DRBG` was proposed for standardization alongside the backdoored `Dual_EC_DRBG`. It is included in ISO/IEC 18031:2011 [ISO11]. The standard text includes only the "default" composite moduli n, not their factors p, q. These unknown factors are likely to form a backdoor for inverting the random number generator.

4.11. Algebraic Irrational Numbers

Appendix F.3 of SP 800-22 [RSN+10] and the STS 2.1.2 test suite discuss binary expansions of four irrational numbers; constants e and π, $\sqrt{2}$, $\sqrt{3}$. It is not explained why these particular numbers are included. From the perspective of "randomness testing", some of them have distinguishing properties.

We note that "irrational numbers" have been proposed as a means of random number generation [Ste21]. We suspect that this was probably inspired by reading NIST SP 800-22 since STS test are also offered as evidence of security. That proposal is clearly insecure.

Weaknesses. While all of these numbers are irrational (not expressible as a fraction), only e and π are transcendental numbers. The square roots $\sqrt{2}$ and $\sqrt{3}$ are algebraic numbers (roots of integer polynomials), and hence their automatic identification and distinguishing from random is relatively easy[7].

Algorithms based on the LLL (Lenstra-Lenstra-Lovász [LLL82]) method can rapidly find "minimal" polynomial coefficients from the binary expansion (of a part) of an algebraic number. Standard computer algebra systems readily provide access to these methods; they are well known to cryptanalysts. Hence algebraic number binary expansions should not be considered indistinguishable from random, and the process of computing them is not a "one-way function" either.

5. Conclusions

We offer criticism of NIST SP 800-22 Rev 1a (and other "black-box") statistical testing methods for random number generators. We consider the continued existence of this document as a NIST Special Publication as harmful to the security of information systems, as it trivializes security analysis and leads to false confidence.

Implementation validation of (deterministic) pseudo-random generators should be focused on algorithmic correctness; the "randomness" of output follows from this.

Evaluation of entropy sources should focus on the true entropy content (physical and stochastic models), built-in health tests, the reliability of the construction, and its resistance to monitoring and interference.

If the SP 800-22 is to be revised, we suggest the new SP focuses on evaluating stochastic models for entropy sources as the SP 800-90 series currently does not address this issue in depth.

We also presented a brief analysis of the "reference generators" included in the SP 800-22 document. None of the presented generators are secure, yet many pass the statistical tests.

We encourage a systematic review of random bit generators against potential "trapdoor" asymmetric backdoors and quantum computing vulnerabilities. Seemingly backdoored and quantum-insecure generators were included in the 2011 version of the random bit generator standard ISO 18031 [ISO11].

Bibliography

[BBS86] Lenore Blum, Manuel Blum, and Mike Shub. A simple unpredictable pseudo-random number generator. *SIAM J. Comput.*, 15(2):364–383, 1986. doi:10.1137/0215025.

[BEB03] Robert G. Brown, Dirk Eddelbuettel, and David Bauer. Dieharder: A random number test suite. Software distribution, accessed January 2021, 2003. URL: https://webhome.phy.duke.edu/~rgb/General/dieharder.php.

[BK15] Elaine Barker and John Kelsey. Recommendation for random number generation using deterministic random bit generators. NIST Special Publication SP 800-90A Revision 1, June 2015. doi:10.6028/NIST.SP.800-90Ar1.

6. Press release April 21, 2014, "NIST Removes Cryptography Algorithm from Random Number Generator Recommendations."

7. Many thanks to Sophie Schmieg for a helpful discussion about efficient solutions to this problem.

[BKR+21] Elaine Barker, John Kelsey, Allen Roginsky, Meltem Sönmez Turan, Darryl Buller, and Aaron Kaufer. Recommendation for random bit generator (RBG) constructions. Draft NIST Special Publication SP 800-90C, March 2021.

[BR11] Elaine Barker and Allen Roginsky. Transitions: Recommendation for transitioning the use of cryptographic algorithms and key lengths. NIST Special Publication 800-131A, January 2011. doi:10.6028/NIST.SP.800-131A.

[Bro49] George W. Brown. History of RAND's random digits – summary. Research Paper P-113, RAND Corporation, June 1949. Also appeared in: Monte Carlo Method, Nat. Bur. Stand., Appl. Math. Series 12 (1951), pp. 31-32. URL: https://www.rand.org/pubs/papers/P113.html.

[FHK+88] Alan M. Frieze, Johan Hastad, Ravi Kannan, Jeffrey C. Lagarias, and Adi Shamir. Reconstructing truncated integer variables satisfying linear congruences. *SIAM J. Comput.*, 17(2):262–280, apr 1988. doi:10.1137/0217016.

[Fis96] George Fishman. *Monte Carlo*. Springer, 1996. doi:10.1007/978-1-4757-2553-7.

[Glo20] GlobalPlatform. Tee protection profile version 1.3. GlobalPlatform Technology, Reference: GPD_SPE_021, July 2020. URL: https://globalplatform.org/specs-library/tee-protection-profile-v1-3/.

[Glo21] GlobalPlatform. Secure element protection profile version 1.0. GlobalPlatform Technology, Reference: GPC_SPE_174, February 2021. URL: https://globalplatform.org/specs-library/secure-element-protection-profile/.

[ISO11] ISO. Information technology– security techniques – random bit generation. Standard ISO/IEC 18031:2011, International Organization for Standardization, 2011. URL: https://www.iso.org/standard/54945.html.

[KBS38] Maurice G. Kendall and Bernard Babington-Smith. Randomness and random sampling numbers. *Journal of the Royal Statistical Society*, 101(1):147–166, 1938. doi:10.2307/2980655.

[KBS39] Maurice G. Kendall and Bernard Babington-Smith. Second paper on random sampling numbers. *Supplement to the Journal of the Royal Statistical Society*, 6(1):51–61, 1939. doi:10.2307/2983623.

[KL14] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography, Second Edition*. CRC Press, 2014. doi:10.1201/b17668.

[KS01] Wolfgang Killmann and Werner Schindler. A proposal for: Functionality classes and evaluation methodology for true (physical) random number generators. AIS 31, Version 3.1, English Translation, BSI, September 2001. URL: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Zertifizierung/Interpretationen/AIS_31_Functionality_classes_evaluation_methodology_for_true_RNG_e.html.

[KS11] Wolfgang Killmann and Werner Schindler. A proposal for: Functionality classes for random number generators. AIS 20 / AIS 31, Version 2.0, English Translation, BSI, September 2011. URL: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Zertifizierung/Interpretationen/AIS_31_Functionality_classes_for_random_number_generators_e.html.

[Leh51] Derrick H Lehmer. Mathematical methods in large-scale computing units. *Annu. Comput. Lab. Harvard Univ.*, 26:141–146, 1951.

[LLL82] H.W. jr. Lenstra, A.K. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261:515–534, 1982. doi:10.1007/BF01457454.

[MKAG20] Linda C. Meiser, Julian Koch, Philipp L. Antkowiak, and Robert N. Grass. DNA synthesis for true random number generation. *Nature Communications*, 11(5869), 2020. doi:10.1038/s41467-020-19757-y.

[MS88] Silvio Micali and Claus-Peter Schnorr. Efficient, perfect random number generators. In Shafi Goldwasser, editor, *Advances in Cryptology - CRYPTO '88, 8th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1988, Proceedings*, volume 403 of *Lecture Notes in Computer Science*, pages 173–198. Springer, 1988. doi:10.1007/0-387-34799-2_14.

[MS91] Silvio Micali and Claus-Peter Schnorr. Efficient, perfect polynomial random number generators. *Journal of Cryptographic Engineering*, 3(3):157–172, 1991. doi:10.1007/BF00196909.

[MvOV96] Alfred Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996. URL: http://cacr.uwaterloo.ca/hac/, doi:10.1201/9781439821916.

[NC21] NIST and CCCS. Implementation guidance for FIPS 140-3 and the cryptographic module validation program. CMVP, November 2021. URL: https://csrc.nist.gov/Projects/cryptographic-module-validation-program/fips-140-3-ig-announcements.

[NIA13a] NIAP. Clarification to the entropy documentation and assessment annex. National Information Assurance Partnership (NIAP), Common Criteria Evaluation and Validation Scheme, 2013.

[NIA13b] NIAP. Entropy submission and review process. National Information Assurance Partnership (NIAP), Common Criteria Evaluation and Validation Scheme, 2013.

[NIA22] NIAP. Protection profile for general-purpose computing platforms. National Information Assurance Partnership, Version 1.0, February 2022. URL: https://www.niap-ccevs.org/MMO/PP/PP_GPCP_v1.0.pdf.

[NIS01] NIST. Security requirements for cryptographic modules. Federal Information Processing Standards Publication FIPS 140-2 (With change notices dated October 10, 2001 and December 3, 2002), May 2001. doi:10.6028/NIST.FIPS.140-2.

[NIS19] NIST. Security requirements for cryptographic modules. Federal Information Processing Standards Publication FIPS 140-3, March 2019. doi:10.6028/NIST.FIPS.140-3.

[RAN55] RAND Corporation. *A Million Random Digits with 100,000 Normal Deviates*. Free Press, 1955. URL: https://www.rand.org/pubs/monograph_reports/MR1418.html.

[RSN+10] Andrew Rukhin, Juan Soto, James Nechvatal, Miles Smid, Elaine Barker, Stefan Leigh, Mark Levenson, Mark Vangel, David Banks, Alan Heckert, JamesDray, and San Vo. A statistical test suite for random and pseudorandom number generators for cryptographic applications. NIST Special Publication SP 800-22 Revision 1a, April 2010. doi:10.6028/NIST.SP.800-22r1a.

[SBK+17] Marc Stevens, Elie Bursztein, Pierre Karpman, Ange Albertini, and Yarik Markov. The first collision for full SHA-1. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 570–596. Springer, 2017. doi:10.1007/978-3-319-63688-7_19.

[SCA12] SCA. Randomness test specification. Cryptography Industry Standard of the P.R. China GM/T 0005-2012, March 2012.

[Sho94] Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *35th Annual Symposium on Foundations of Computer Science, Santa Fe, New Mexico, USA, 20-22 November 1994*, pages 124–134. IEEE, 1994. URL: https://arxiv.org/abs/quant-ph/9508027, doi:10.1109/SFCS.1994.365700.

[Ste21] Crown Sterling. Crown sterling cryptographic security protocol. A Technical White Paper (From a Snake Oil Vendor), December 2021. URL: https://f.hubspotusercontent10.net/hubfs/9477568/Crown%20Sterling%20White%20Paper%202021.pdf.

[TBK+18] Meltem Sönmez Turan, Elaine Barker, John Kelsey, Kerry A. McKay, Mary L. Baish, and Mike Boyle. Recommendation for the entropy sources used for random bit generation. NIST Special Publication SP 800-90B, January 2018. doi:10.6028/NIST.SP.800-90B.

[TC218] TC26. (Russian) translation of AIS-31. Technical Committee for Standardization, Cryptographic Protection of Information, 2018. URL: https://tc26.ru/standarts/perevody/perevod-ais-31.html.

[Wal98] John Walker. ENT – a pseudorandom number sequence test program. Open Source Program, October 1998. URL: https://www.fourmilab.ch/random.

[WYY05] Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding collisions in the full SHA-1. In Victor Shoup, editor, *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, volume 3621 of *Lecture Notes in Computer Science*, pages 17–36. Springer, 2005. doi:10.1007/11535218_2.