The ComScire® CryptoStrong™ Random Number Generator.

Scott A. Wilber[a] and Luis Araujo[b]

Development of quantum computers will soon provide a means of weakening or breaking many currently used encryption methods. ComScire's[1] new CryptoStrong random number generators[2] provide the highest level of unpredictability and reliability available. The Model CS128M includes an entropy source provably surpassing the security of any other known generator, and cryptographic post processing comprising an AES-256 encryption module as defined by both NIST and the German AIS-20/31. Strong tamper resistance prevents reading or changing firmware and hardware design provides high resistance against side channel attacks.

KEYWORDS: Random Number Generator, Entropy, Predictability, Data Security, Cybersecurity, Cryptographic Strength, Cryptography, Post-Quantum, Quantum Computer.

PART 1: ENTROPY SOURCE.

INTRODUCTION.

Random number generators (RNGs) are intended to provide a source of unpredictable numbers or bits for a variety of important applications. These applications notably include generating keys for encrypting data and messages to prevent unauthorized access. For example, online financial transactions, information in the Cloud and government and military secret transmissions. In addition, random numbers form the basis of fairness for all lotteries and electronic "games of chance."

RNGs consist of two fundamental types: 1) pseudorandom generators, which use mathematical algorithms to provide numbers that simulate or give the appearance of randomness, and 2) true random number generators or nondeterministic generators, which require a physical source of entropy to produce actually unpredictable numbers. Pseudorandom generators must be seeded or initialized with a true random number. Nondeterministic generators are essential for every type of encryption and data security.

Many applications of random numbers are critically important and even a partial failure of the RNG used could have catastrophic consequences. Users of these random numbers must be

[a] President of The Quantum World Corporation. swilber@comscire.com
[b] Senior Engineer at The Quantum World Corporation. laraujo@comscire.com
[1] ComScire is a registered trademark of The Quantum World Corporation. https://comscire.com/
[2] Protected by multiple issued and pending US patents and Foreign Patents.

able to rely implicitly on their maximum cryptographic strength[3]. This means that every random bit must be independent identically distributed (i.i.d.) and uniformly distributed, making them absolutely unpredictable beyond chance expectations given unlimited computing power (including future quantum computers). Another way of stating this is that the entropy of a sequence of these bits is equal to the length of the sequence less an infinitesimal amount.[4] NIST formerly defined "Full Entropy" as $H = (1 - \varepsilon)$ per bit, where $0 \le \varepsilon \le 2^{-64}$ [SP 800-133]. In practice we are limited to finite sequence lengths for testing, so a more realistic, though less precise definition is, 'a source of full-entropy bitstrings serves as a practical approximation to a source of ideal random bitstrings of the same length' [SP 800-90A1].

The entropy sources in the PureQuantum® QRNGs sold by ComScire are described in extensive detail by Wilber [Wil13]. They comprise both classical, primarily thermal and shot noise sources, and quantum mechanical sources produced by quantum tunneling in the CMOS transistors. The magnitude of the noise sampled from these sources is modeled and measured to allow accurate calculation of the resultant entropy. The entropy source in the CS128M is the same as in the PureQuantum models – although somewhat expanded – with small adjustments due to 1.1V versus 1.2V core operating voltage and feature dimensions of 28nm versus 65nm of the Cyclone VE and Cyclone III Field-Programmable Gate Array (FPGA) families respectively.[5]

These entropy sources have been extensively tested for years in commercial products, including continuous tests of individual sources amounting to hundreds of trillions of bits. They use no postprocessing so the tests revealed the fundamental quality of the raw output of the entropy sources. Testing to hundreds of terabits would have revealed statistical defects on the order of 0.0000001 (100 parts per billion). No defects were observed.

ENTROPY SOURCES AND THE EFFECT OF POST-PROCESSING.

Virtually all previously described entropy sources (except in ComScire's PureQuantum® generators) have significant statistical defects and a corresponding reduction of total entropy. In order to satisfy the statistical requirements for modern TRNGs, their output sequences are "whitened" by passing imperfect random numbers through a cryptographic hash function. This has the effect of greatly improving their statistical properties.

Conditioning a TRNG sequence having deficient entropy does not necessarily make it entirely unpredictable, especially if post-processing methods are also used to extend or increase the number of output numbers relative to the number of bits of entropy provided to the input. The output numbers are extended using a deterministic algorithm that is periodically reseeded by the true entropy source. The amount of true entropy per output bit is equal to the number of bits of entropy input to the algorithm divided by the number of output bits.

[3] Cryptographic Strength is related to the number of operations a computer must use to break the encryption to reveal plain text. It depends primarily on the number of bits of entropy in the key, but also on the encryption method and the type of computer – classical or quantum mechanical.

[4] An *absolutely perfect* random sequence is theoretically impossible to generate, but it is possible to approach that ideal arbitrarily closely, as shown in this paper.

[5] Cyclone V and Cyclone III are FPGA family names of Intel Corporation. These were formerly made by Altera Corporation, which was acquired by Intel in Dec. 2015.

Statistical tests alone cannot easily quantify the amount of physical or *true* entropy used to produce random output sequences. Output sequences are predominantly pseudorandom when they have been greatly extended using deterministic post-processing. Such sequences are currently beyond the ability of most brute-force attacks, but they are theoretically vulnerable to attack by a sufficiently advanced quantum computer.

The ideal entropy source must provide sequences that are provably unpredictable, given any realistic length of previous bits and unlimited computational power. In addition, the ideal source should provide entropy at a rate high enough to remove the need to algorithmically extend the output sequence. The combination of ideal entropy source and cryptographic post-processing results in maximum possible cryptographic strength, even when attacked by arbitrarily advanced quantum computers.

ENTROPY FROM VARIOUS SOURCES.

Two broad types of noise in the production and measurement of entropy are extrinsic and intrinsic. Extrinsic sources are not directly part of the generator entropy source and are coupled to the source by electromagnetic fields, power supply variations or even mechanical vibrations. Extrinsic noise must not be relied on as an entropy source in a secure random number generator system because of the potential to observe and even inject patterns into the generator circuit. Intrinsic sources are inherent in the generator source and arise from fundamental physical principles. Intrinsic sources in transistors and integrated circuits include shot noise from diffusion and tunneling currents, thermal or Johnson noise, flicker or 1/f noise and generation-recombination noise. Intrinsic noise is chaotic and nondeterministic. Most extrinsic sources can be eliminated or greatly reduced by proper design and shielding of the generator, while intrinsic sources are usually not reducible below their theoretical value.

The design presented in this paper requires measurements of the entropy from the sources being used. Most modern digital ICs are constructed using MOS transistors in a complementary or CMOS configuration. The entropy produced in gates constructed from these transistors is measurable by detecting variations in transition times that produce jitter in an oscillating signal passing through them. This is done by latching the state of a free-running ring oscillator (RO) with an independent clock signal. A ring oscillator is a multi-stage delay line containing an odd number of inverting gates with its output connected to its input. Each gate in the ring oscillator adds a certain amount of jitter to the signal transition as it passes through. The statistical distribution of jitter due to intrinsic sources is approximately normally distributed, and the total cumulative jitter from these sources is the jitter introduced by a single stage multiplied by the square root of the number of stages the transition has passed through before being measured.

DEFINITIONS OF SYMBOLS.

ε is the deviation from ideal entropy, $\varepsilon = 1 - H$.
E_P is the deviation from ideal predictability, $E_P = P - 0.5$
f_{ring} is the oscillation frequency of a ring oscillator, $f_{ring} = 1/(2\, n_{lut}\, \tau_p)$ Hz.
H is entropy.

h^{-1} is the mathematical inverse of the entropy equation.
H_P is entropy as a function of P, $H_P = -\big(PLog_2P + (1-P)Log_2(1-P)\big)$ bits.
H_T is total entropy produced by combining a number of entropic sources or numbers.
J_F is the fractional jitter in a ring oscillator, $J_F = f_{ring}\, J_{LUT}\, \sqrt{2n_{lut}}$ rms.
J_{LUT} is the rms jitter caused by a single LUT.
n is the number of entropic bits to be combined.
N is the number of bits in a sequence.
n_{lut} is the number of LUTs composing a ring oscillator.
$p(x)$ is the probability of x occurring.
P is predictability – the probability of correctly predicting or guessing the next bit in a sequence.
P_C is combined predictability, $P_C = (P_{FP}+1)/2$.
P_F is fractional predictability, $P_F = 2P - 1$.
P_{FI} is the fractional predictability of independent bits to be combined.
P_{FP} is the product of two or more fractional predictabilities.
P_{FR} is the resultant fractional predictability, $P_{FR} = P_{FI}^n$.
P_N is the maximum next bit predictability given an N-bit sequence, $P_N \le ½ + \sqrt{N} \times (P - ½)$.
RO is a ring oscillator.
τ_p is the propagation delay through a single LUT or gate, $\tau_p = 1/(2\, n_{lut}\, f_{ring})$ s.

COMBINING BITS OF ENTROPY.

Entropy and predictability are terms with different meanings in different fields. Shannon entropy [Sha48] is defined for the binary case when only 1 or 0 may be produced as, $H = -(p(1)Log_2p(1) + p(0)Log_2p(0))$, where $p(1)$ and $p(0)$ are the probabilities of a one and a zero occurring respectively and Log_2 is the logarithm in base 2; used when entropy is quantified in bits. For this discussion, $p(1)$ is replaced by predictability, P, defined as the probability of correctly guessing the next bit in a sequence, and $p(0)$ is replaced by $1-P$, where $0.5 \le P \le 1.0$. Substituting predictability for the probabilities, entropy as a function of P is defined, $H_P = -(PLog_2P + (1-P)Log_2(1-P))$. When the value of H is measured or theoretically calculable, $P = h^{-1}$, where h^{-1} is the mathematical inverse of the entropy equation. The inverse is normally performed numerically since there appears to be no closed-form equation for it. The inverse of the entropy equation has two solutions, but only the one where $P \ge 0.5$ is used.

It is essential to be able to calculate the precise levels of entropy or predictability of bits combined by XOring them together[6]. The predictability of each bit is converted to fractional predictability, $P_F = 2P - 1$. The fractional predictabilities of each of the original bits are then multiplied to produce their product, P_{FP}, and the combined predictability, P_C is finally calculated, $P_C = (P_{FP}+1)/2$. The combined or total entropy, H_T, is calculated using the equation for $H_P = f(P_C)$.

[6] The XOR function, or equivalently the parity function when more than two bits are combined, is used because it was proven to be the most efficient algorithm for improving the randomness properties when combining imperfect random sequences [San86], and because its effect on combining entropy sources has been determined by the research underlying this paper and elsewhere [Dav02].

PHYSICAL MEASUREMENTS AND TOTAL ENTROPY CALCULATION.

An Intel Field-Programmable Gate Array (FPGA) was selected for the CryptoStrong Model CS128M design. A number of measurements and tests were performed using several 5CEFA4U19C8N FPGAs to determine the variables required in the entropy model.

The propagation delay, τ_p, through a single LUT or FPGA "gate" is found by measuring the average frequency of several ring oscillators and calculating $\tau_p = 1/(2\, n_{lut}\, f_{ring})$, where n_{lut} is the number of LUT's in the ring and f_{ring} is the frequency of oscillation. Six physically independent ring oscillators were programmed using 11 non-inverting gates and one inverting gate, arranged in single logic blocks to minimize variations between rings. The average measured ring oscillator frequency was, $\overline{f_{ring}}$ = 198.4±7.1MHz (±1 SD), giving a propagation delay of $\tau_p = 210.0 \pm 7.45$ ps.

The entropy source includes three independent but otherwise identical sources, which are combined into one final source. An offline diagnostics mode can provide raw data from a number of internal test points for external testing. These test points are designated levels 1, 2 and 3, representing three levels in the combination of multiple ring oscillator measurements. Each level includes three different source test points to allow representative statistical measurements throughout the entropy source.

The level 1 test outputs are produced by XOring signal taps from pairs of ring oscillators, each RO having different numbers of LUTs and signal taps, and latching the XOred outputs with a 128 MHz clock. The three available level 1 outputs from two FPGAs were measured to produce a mean entropy[7] of, H = 0.8670525 with corresponding P = 0.7112873 and P_R = 0.4225746. Programs 1a-c in Appendix II give the relationships between oscillator jitter, predictability and entropy. Each pair of rings was fully modeled and solved backward numerically giving the single LUT jitter with a geometric mean value of, J_{LUT} = 34.066353 ps rms.

Level 2 test outputs are produced by XOring 15-level 1 outputs. Six measurements from two FPGAs gave a mean entropy, $H = 1 - \varepsilon$, where $\varepsilon = 5.151157 \times 10^{-11}$, with corresponding P = 0.500004225233 and $P_R = 8.450466 \times 10^{-6}$. Numerical solution gives the mean single-LUT jitter, J_{LUT} = 30.91735 ps rms.

The weighted geometric mean of level 1 & 2 measurements gives, J_{LUT} = 31.34873 ps rms. This value was used in a model of the average level 2 output (taking the average number of RO LUTs and taps) From this the averages used throughout the model were calculated: Average P_R/tap = 0.8466737[8], P/tap = 0.9233368 and H/tap = 0.3903111. The algebraic average number of taps/ring = 2.333333 and the geometric mean of LUTs/ring, n_{lut} = 9.528536 adjusted to 9.4765952 for internal consistency[9].

[7] Entropy is a nonlinear variable so its mean is calculated using the geometric mean of the associated relative predictabilities. Details are described in Appendix I.
[8] The average level 2 output is produced from 70 taps and the single tap P_R is the 70th root of the total P_R.
[9] The geometric mean of n_{lut} is 9.5285357. A slight adjustment (a reduction of about 0.5483%) was determined numerically.

Calculating Total Entropy.

The following steps use the measurements and physical design to calculate the total entropy from the entropy source in the CS128M. The average number of LUTs in the ring oscillators comprising the entropy source, n_{lut}, and the geometric mean of the number of signal taps in the rings are used to simplify the calculations.

- Mean oscillator frequency with n_{lut} = 9.4762952 and τ_p = 210.0 ps yields a mean ring oscillator frequency, $f_{ring} = 1/(2\, n_{lut}\, \tau_p) = 251.254$ MHz.
- J_{LUT} = 31.34873 ps rms: the weighted geometric mean from all statistical measurements.
- Ring oscillator fractional jitter, $J_F = f_{ring}\, J_{LUT}\, \sqrt{2n_{lut}} = 0.03428991$ rms.
- Entropy per sampled tap calculated numerically = 0.3903111
- Predictability per tap is, $P = h^{-1}$ = 0.9233368 and $P_{FI} = 2P - 1$ = 0.8466737.
- Each ring provides an average of 2.33333 signal taps per ring times 2 rings XORed together gives 4.66667 taps per sample (each average level 1 output). This is multiplied by 15 samples from 15 pairs of ring oscillators. One level 2 output is thus produced from n = 70 mean samples per bit at 128 MHz sample rate.
- The resultant fractional predictability at level 2 is, $P_{FR} = P_{FI}^{n} = 0.8466737^{70} = 8.712141 \times 10^{-6}$. P_C = (8.712141×10^{-6} + 1)/2 = 0.500004356070. The average entropy for each level 2 output is 1.0 – ε, where ε = 5.47512 x 10^{-11}.
- The next internal test level (level 3) results from XORing 32 independent level 2 outputs yielding, $P_{FR} = P_{FI}^{n} = 0.8466737^{2240} = 1.213377 \times 10^{-162}$ and P_C = (1.21337×10^{-162} +1)/2 = 0.5 + 6.066887×10^{-163}. The entropy at this level is, H = 1– ε where ε =1.06203×10^{-324}.
- The final output of the entropy source is the result of XORing the three level 3 outputs to produce each final output bit at 128 Mbps. This produces $P_{FR} = P_{FI}^{n} = 0.8466737^{6720} = 1.213377 \times 10^{-486}$ and P_C = (1.7864372×10^{-486} +1)/2 = 0.5 + 8.9321861×10^{-487}. The theoretical total entropy at the final output of the entropy source is, H_T = 1– ε where ε = 2.3021×10^{-972}. The fractional defect in other statistics is on the order of 9 x ×10^{-487}, which makes the output indistinguishable from a sequence of perfectly random bits.

In most of these calculations, the precision is reported as 6 or more digits. These digits are presented for calculation purposes while the actual accuracy may be considerably less. For example, the measured $\overline{f_{ring}}$ was reported as 198.4MHz. However, the calculated standard deviation was 7.1MHz so the accuracy is more correctly 198.4 ±7.1MHz or ± 3.6%.

DISCUSSION.

The entropy and size of statistical defects in previous random number generators and entropy sources could not be qualified beyond what could be directly tested using statistical tests. This was primarily due to a lack of knowledge of how to generate random numbers with a precisely quantifiable amount of entropy [Sar19]. To overcome previous limitations requires a deep understanding of physical entropy sources and how to measure them, and a clear and complete model for using the measured output to achieve any desired level of entropy and statistical defect in an output sequence.

Design equations and specific practical designs for simple, inexpensive yet highest quality nondeterministic random generators are presented. The design is implemented in CMOS integrated circuits, but the principles may be applied to random number generators of virtually any design or entropy source. NIST formerly defined “full entropy” as $H = (1 - \varepsilon)$ bits per bit, where $0 \leq \varepsilon \leq 2^{-64}$ [SP 800-90C, p. 4], that is, $\varepsilon \leq 5.421\times10^{-20}$. The entropy source described here not only meets, but vastly surpasses that requirement without any type of post processing, conditioning or randomness correction.

The deviation of predictability above the ideal value of 0.5 is $E_P = P - ½$. E_P is also the approximate level of statistical defects in the sequence. About $1/x^2$ bits must be tested to observe a statistical defect in a sequence with $x = E_P$. A major result of the model is to show it is possible to generate and qualify random sequences whose statistical defects are too small to be detected by direct testing.

Generators constructed following the design rules in this paper can produce sequences indistinguishable from an ideal nondeterministic sequence – beyond the capability of any theoretical quantum computer to predict beyond chance. The entropy source described in this paper has a theoretical E_P of about 8.9×10^{-487}. Photonic or other common quantum generators cannot begin to approach that level of nondeterminism due to inherent limitations in the entropy measuring components.

The German Federal Office for Information Security (BSI) requires an entropy source of only $H \geq 0.997$, giving $P \leq 0.53223$. These numbers may be passed through a cryptographic hash function to correct their entropy deficit and make them unpredictable. However, a hash function is a deterministic algorithm, usually of open design. Knowing the raw random numbers have significant predictability may allow a future quantum computer with virtually unlimited computational power to crack the hash with some form of collision or brute force attack (See Cimpanu [Cim19] as a suggestive example.).

The entropy source in the CS128M produces raw bits with effectively perfect entropy and unpredictability. That is the best possible solution to resist threats arising in the post-quantum age.

Appendix I – Entropy Model Equations.

This section provides a number of equations and direct tests demonstrating the basis of the entropy model used to design ComScire's PureQuantum® QRNG models and the entropy source in the Model CS128M CryptoStrong™ generator. The model demonstrates how to combine a number of measurements of a physical entropy source or sources to achieve desired levels of entropy and predictability.

The fundamental relations of the model are:

1) Shannon entropy, H, for a binary sequence with an alphabet [0, 1] is $H = -(p(1)Log_2 p(1) + p(0)Log_2 p(0))$, where $p(1)$ and $p(0)$ are the probabilities of a one or a zero occurring respectively, and Log_2 is the logarithm in base 2; used when entropy is quantified in bits. This equation is a function of only one independent variable since $p(0) = 1 - p(1)$. For this discussion, $p(1)$ is replaced with predictability, P – defined as the probability of correctly predicting the next bit in a sequence – and $p(0)$ is replaced by $1-P$, where $0.5 \leq P \leq 1.0$. After substituting predictability for the probabilities, entropy as a function of P is $H_P \cong -(PLog_2 P + (1-P)Log_2(1-P))$. This equation becomes progressively more accurate as ε, deviation from the ideal $H_P = 1.0$, becomes very small.
2) Predictability, P, and entropy, H of a sequence[10] of true random numbers are two related representations of its randomness. Each variable may be mathematically transformed into the other. When H is measurable or theoretically calculable, P may be calculated as $P = h^{-1}$, where h^{-1} is the mathematical inverse of the entropy equation. The inverse is performed numerically (see Appendix II) since there appears to be no closed-form solution; however, see the approximations below. The inverse of the entropy equation has two solutions, but only $P \geq 0.5$ is used.
3) Entropy can be measured statistically with reasonable accuracy using, for example, an update of Maurer's "Universal Test" [Cor99], without prior knowledge of the various statistical defects that may be present in the sequence being tested. From these measurements the effective predictability can be calculated.

    The following approximations for H_P, P, ε and E_P have been found[11]:

    a) $H_P \cong 1 - (2/\ln 2)(P - ½)^2$
    b) $P \cong ½ + \sqrt{\ln 2/2}\ \sqrt{1 - H_P}$
    c) $\varepsilon \cong (2/\ln 2){E_P}^2$
    d) $E_P \cong \sqrt{\ln 2/2}\ \sqrt{\varepsilon}$

These approximations conveniently allow H_P and P to be calculated with high accuracy and without the use of extended precision when H_P and P are only moderately close to 1.0 and 0.5 respectively.

H_P is an exact function of P when bias in the counts of 1s and 0s is the only statistical defect.[12]. However, H_P is decreased by the presence of any patterns or statistical defects. The

[10] For uniformity throughout **Part 1** of the paper, H refers to the value on a per-bit basis. This applies when either bits or sequences are mentioned. The number of bits of entropy in a sequence of length n is simply $n\,H$ or $n - n\,\varepsilon$.
[11] Their derivation is omitted for brevity, but their accuracy is easily confirmed.

decrease in H_P from its ideal value of 1.0 is $\varepsilon = 1 - H_P$. The approximation for ε shows it decreases as the square of the deviation of P from its ideal value of 0.5. A significant defect in P of 0.0001, i.e., $P = 0.5001$, gives $\varepsilon = 2.88539 \times 10^{-8}$. It was shown elsewhere the mathematical distance from a uniform distribution of a sequence decreases slowly as ε decreases [Skó15]. The approximation for E_P reveals it decreases as the square root of ε.

4) Relative predictability, P_R, is defined as $P_R = 2P - 1$. When independent bits are XOred together, the relative predictability of the resultant sequence of bits, P_{FR}, is the product of the relative predictabilities of the initial bits. The resultant entropy is easily calculated using the equation for H_P after calculating $P = (P_{FR} + 1)/2$. Or, the entropy may be measured statistically if ε is not too small.

   When there is significant dependence between the XOred bits, usually in the form of autocorrelation between bits measured from a single entropy source, the entropy of the resultant sequence is reduced by the mutual information. A statistical measurement is necessary to provide an accurate assessment of the entropy when such dependence is present. Good agreement between statistical measurements and the calculated H_P indicates the bits are sufficiently independent.
5) A supplemental algorithm is derived from the preceding. The mean of n entropy measurements is calculated by:
   a) Converting each entropy to relative predictability, P_R,
   b) Finding the geometric mean (the nth root of their product) of all P_Rs,
   c) Converting the mean P_R to P,
   d) Using P to calculate the mean entropy.
6) P_N is the maximum next bit predictability given a previous sequence of length N bits with single bit predictability, P. This expression was developed from a study of random walk statistics [Ste01]. In equations describing a one-dimensional random walk, the probabilities of a 1 or a 0 occurring were replaced by predictability, P and $1 - P$ respectively. A fundamental derived relation is, $N = \frac{2P_N - 1}{2P - 1} \ln\frac{1-P_N}{P_n} / \ln\frac{1-P}{P}$, where N is the average number of bits that must be used in a random walk to produce a bit with predictability, P_N, and P_{RN} is the relative predictability of the next bit. Using the fact that $\ln\frac{1-P}{P} \cong -2P_R$ when $P_R \ll 1$, $N \cong \frac{P_{RN}}{P_R}\frac{-2P_{RN}}{-2P_R}$, which simplifies to, $N \cong \left[\frac{P_{RN}}{P_R}\right]^2$.
   Solving, $P_{RN} \cong \sqrt{N}P_R$. Substituting $P_{RN} = 2P_N - 1$ and $P_R = 2P - 1$ and simplifying yields $P_N \cong ½ + \sqrt{N} \times (P - ½)$. An alternate derivation gives, $P_N \leq 1/(1 + ((1-P)/P)^n)$, where $n \cong \sqrt{N}$. The accuracy of the approximation is > 8 digits when $P \leq 0.50001$, and becomes more accurate as $P \rightarrow 0.5$.

[12] Bias, the ratio of 1s to total bits, is by definition $p(1)$ and $p(0) = 1 - p(1)$. Hence, Shannon entropy is exact when only bias is present. If the bits are biased but independent, an iterated procedure [Per92], originally due to Von Neumann, can be used to extract unbiased bits with entropy approaching the total entropy in the original sequence. This procedure fails when any other form of statistical defect is present.

Appendix II – Programs.

US Pat. no. 6,862,605, [Wil05] discusses how to calculate the entropy of a sampled oscillatory signal given rms jitter as a fraction of the oscillatory signal period. (See also: [Sun07]). The entropy is calculated numerically by first calculating the average probability of correctly predicting the next sampled value of the oscillator signal, P, and then using Shannon's entropy equation modified to use the single variable, P. The fractional jitter must be adjusted to an effective jitter, $J_E = J_F\sqrt{f_{osc}/f_{samp}}$, where f_{osc} is the ring oscillator frequency and f_{samp} is the sampling frequency. This adjustment accounts for the fact the effective cumulative jitter at each sample time is that jitter which accumulates since the previous sample. The following Mathematica[13] programs perform the required numerical calculations:

```
prob[mu_, rho_]:=Sum[CDF[NormalDistribution[mu, rho], x+1/2]-                Program 1a
CDF[NormalDistribution[mu, rho], x], {x, -Round[6 rho], Round[6 rho]}]
avgprob[rho_, hf_, lf_]:=(ro=rho Sqrt[hf/lf]; divisions=Max[1000, Ceiling[5/ro]];   Program 1b
If[ro>.9, .5, N[2Sum[prob[mu, ro], {mu, 0, 1/2, 1/(4divisions)}]/(2divisions+1)-
Sum[prob[mu, ro], {mu, 0, 1/2, 1/(2divisions)}]/(divisions+1)]])
H[rho_, hf_, lf_]:=(apr=avgprob[rho, hf, lf];                                Program 1c
(-1/Log[2])(apr Log[apr]+(1-apr)(Log[1-apr])))
```

The function H[rho_, hf_, lf_] calculates entropy, where the arguments, rho, hf and lf are the fractional jitter, J_F, and the ring oscillator and sampling frequencies respectively, mu = 0.0. The function avgprob[rho_, hf_, lf_] calculates the average predictability, P. When the fractional jitter gets smaller the number of divisions used in the function avgprob is automatically increased. $5/J_E$ divisions rounded up to the next higher integer will yield about three significant digits of accuracy for J_E down to 0.00001.

The precision of most programming languages cannot represent the numbers in these calculations. When necessary, both entropy and its inverse were conveniently calculated in Mathematica using extended precision up to 1000 digits.

[13] From Wolfram Research.

Appendix III – Error Analysis.

Values of some parameters are calculated from measurements which may be in error, or change over time, temperature, supply voltage and chip-to-chip process variations. Errors in these parameters will result in an increase or decrease in the entropy produced. For simplicity, only the values of ε are shown in the tables, where the total entropy is $H_T = 1 - \varepsilon$.

Table 1 shows the effect on total entropy (H_T) in the output bits when using a wide range of ring oscillator frequency (f_{ring}) from a low of 80% to a high of 125% of the nominal value used in the paper.

RO Frequency, f_{ring}	Low ($\mathbf{0.8}\times$)	Nominal	High ($\mathbf{1.25}\times$)
$1 - H_T$: ε	1.69×10^{-678}	2.30×10^{-972}	7.15×10^{-1411}

Table 1

Total entropy is calculated from the fractional jitter in the LUT output transitions, J_{LUT}. Table 2 shows the effect on total entropy of varying J_{LUT} over a range of 0.5 to 2 times the value used in the paper.

Transition Jitter, J_{LUT}	Low (½ ×)	High ($\mathbf{2}\times$)
$1 - H_T$: ε	1.18×10^{-465}	8.79×10^{-2150}

Table 2

Test level 2 entropy is both measured and calculated using the entropy model. Entropy is then used to calculate, $E_P = P - ½$. Close agreement between measured and calculated values is significant validation of the model. Table 3 shows the effect on total output entropy of varying the level 2 E_P from 0.5 to 2 times the nominal, calculated value. This is effectively the range spanned by the direct statistical measurement, shown for comparison.

Level 2 E_P	Low (½ ×)	Nominal	High ($\mathbf{2}\times$)
Calculated E_P	2.178×10^{-6}	4.356×10^{-6}	8.712×10^{-6}
Measured E_P ±1SD	2.1×10^{-6}	5.1×10^{-6}	8.1×10^{-6}
Final Output ε	3.67×10^{-1030}	2.30×10^{-972}	1.44×10^{-914}

Table 3

These three tables show the directly or indirectly measured variables used to calculate total entropy in the output of the Model CS128M described in this paper. When these variables are changed over a wide range, total entropy varies as shown. The ranges are believed to include worst-case values.

The single worst-case result of $\varepsilon = 1.17\times10^{-465}$ indicates a statistical defect of 2.02×10^{-233} would theoretically be present in the entropy source output sequence. This level of defect is clearly not detectable by any known or theoretical means. Nor could an infinitely powerful quantum computer predict a next bit beyond chance (no better than ½ + 1.28 x 10^{-225})[14] given a previous sequence of 4 Pb (petabits) that would be generated in a full year at 128 Mbps.

Functional Testing versus Operating Temperature.

Table 4 shows the variation of the mean measured entropy at level 1 test points versus temperature of the FPGA. The temperature was measured with a thermocouple in direct contact with the middle of the FPGA heatsink. The accuracy of the temperature is approximately ± 2°C. The final ε was calculated from the inferred initial relative predictability, P_{FI}, at each temperature.

Measured Entropy Vs. Temperature		
T °C	Level 1 H	Final ε (Calc.)
-40	0.858	1.01 10^{-1036}
-35	0.860	2.76 10^{-1045}
-30	0.864	2.53 10^{-1066}
0	0.862	5.02 10^{-1054}
10	0.833	1.10 10^{-940}
20	0.824	1.14 10^{-910}
32	0.802	1.78 10^{-875}
40.5	0.755	4.71 10^{-716}
50	0.814	4.71 10^{-876}
60	0.890	2.46 10^{-1193}
65	0.897	1.21 10^{-1233}
80	0.887	1.99 10^{-1175}
90	0.836	1.69 10^{-951}

Table 4

The temperature of the system board was varied from about –50 to +100°C. The generator system was fully functional at all times as indicated by continuously passing all internal tests and providing test data output. Entropy measurements were made from –40 to +90°C. All values are easily within the believed worst-case ranges of Tables 1 and 2.

[14] Predictability of the next bit given a sequence of length N is found to be (see Appendix I), $P \leq ½ + \sqrt{N} \times E_P$, where E_P is the single bit predictability – ½.

PART 2: CRYPTOGRAPHIC DRBG POST-PROCESSING.

INTRODUCTION.

Part 2 is a step-by-step documentation describing the CryptoStrong™ Model CS128M RNG implementation to comply with NIST Special Publication SP 800-90C and BSI AIS 20/31 guidelines for constructing random numbers. NIST SP 800-90C (2nd Draft), Recommendation for Random Bit Generator (RBG) Constructions, specifies RBG construction consisting of deterministic random bit generators (DRBG) mechanisms, as specified in NIST SP 800-90A, and entropy sources, as specified in SP 800-90B. Likewise, BSI AIS 20/31, Functionality Classes for Random Number Generators (RNG), describes the German Common Criteria (CC) for implementing RNGs grouped in specific hierarchal classes which may consist of a physical RNG (PTRNG) and a deterministic RNG (DRNG).

The following definitions clarify and compare the terminology used by NIST and the German BSI:

NIST Terminology.

Random Bit Generator (RBG): A device or algorithm that is capable of producing a random sequence of (what are effectively indistinguishable from) statistically independent and unbiased bits. An RBG is classified as either a DRBG or an NRBG.

Non-deterministic Random Bit Generator (NRBG): An RBG that always has access to an entropy source and (when working properly) produces output bit strings that have full entropy. Often called a True Random Number (or Bit) Generator.

Deterministic Random Bit Generator (DRBG): An RBG that includes a DRBG mechanism and (at least initially) has access to a randomness source. The DRBG produces a sequence of bits from a secret initial value called a seed, along with other possible inputs. A DRBG is often called a Pseudorandom Bit (or Number) Generator.

Backtracking Resistance: A property whereby an attacker with knowledge of the state of the RBG at some time(s) subsequent to time T would be unable to distinguish between observations of ideal random bit strings and (previously unseen) bit strings that are output by the RBG at or prior to time T.

Prediction Resistance: A property whereby an adversary with knowledge of the state of the RBG at some time(s) prior to T (but incapable of performing work that matches the claimed security strength of the RBG) would be unable to distinguish between observations of ideal random bitstrings and (previously unseen) bitstrings output by the RBG at or subsequent to time T.

BSI Terminology.

Random Number Generator (RNG): A group of components or an algorithm that outputs sequences of discrete values (usually represented as bit strings).

Physical RNG (PTRNG): A RNG where dedicated hardware serves as an entropy source. NOTE: we use the short term “physical RNG” for physical true RNG as well because all physical RNG are true RNG by definition. We use the abbreviation “PTRNG” instead of “PRNG” to avoid confusion with pseudorandom generators.

Deterministic RNG (DRNG): An RNG that produces random numbers by applying a deterministic algorithm to a randomly selected seed and, possibly, on additional external inputs.

Hybrid PTRNG: A PTRNG with a (complex) post-processing algorithm. The goal of (sometimes additional) cryptographic post-processing with memory is to increase the computational complexity of the output sequence. NOTE: A complex algorithmic post-processing algorithm may be viewed as an additional security anchor for the case when the entropy per output bit is smaller than assumed.

Backward secrecy: The assurance that previous output values cannot be determined (i.e., computed or guessed with non-negligible probability) from the current or future output values.

Forward secrecy: The assurance that subsequent (future) values cannot be determined (i.e., computed or guessed with non-negligible probability) from current or previous output values.

Enhanced Backward Secrecy: The assurance that previous output values of a DRNG cannot be determined (i.e., computed or guessed with non-negligible probability) from the current internal state, or from current or future output values. NOTE: The knowledge of the current state of a pure DRNG (with no additional input or with publicly known input) implies knowledge of the current and future output.

Enhanced Forward Secrecy: The assurance that subsequent (future) values of a DRNG cannot be determined (i.e., computed or guessed with non-negligible probability) from the current internal state, or from current or previous output values. NOTE: The enhanced forward secrecy may be ensured by reseeding or refreshing the DRNG internal state, which may be performed automatically or initiated on user demand.

NIST-BSI Terminology Cross Reference.

RBG = RNG
NRBG = PTRNG or Hybrid PTRNG
DRBG = DRNG
Backtracking Resistance = Enhanced Backward Secrecy
Prediction Resistance = Enhanced Forward Secrecy

CS128M IMPLEMENTATION DETAILS

1. CS128M Construction.

The cryptographically strong random number generator CS128M is AIS 20/31 Class PTG.3-compliant Hybrid PTRNG, or SP 800-90C-compliant NRBG as defined by NIST. The AIS 20/31 Class PTG.3 was selected to guarantee the highest RNG security level as defined by AIS 20/31. In pursuance of complying with both standards, the NIST SP 800-90C XOR-NRBG construction was selected in order to be compliant with NIST's RBG construction recommendations and satisfy AIS 20/31 Class PTG.3 requirements.

The construction includes:

1) Entropy source model as defined by SP 800-90B and AIS 20/31.
2) Health tests in accordance with SP 800-90[A/B] and AIS 20/31 standards.
3) An approved SP 800-90A DRNG method that is also AIS 20/31 DRG.3 compliant.

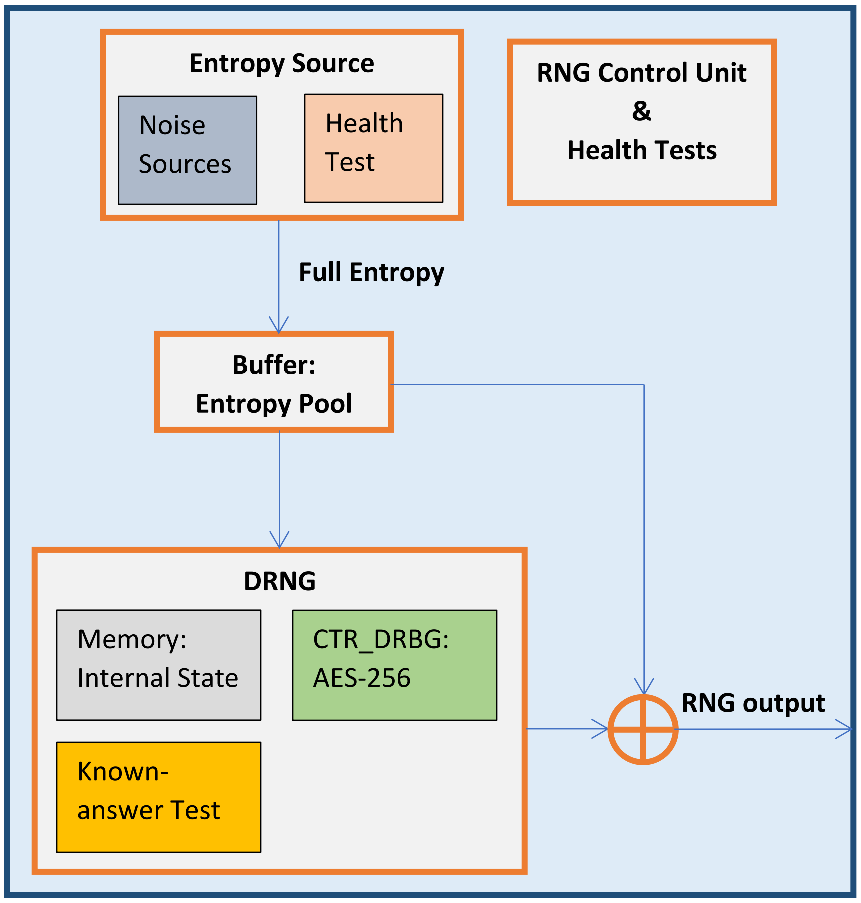


Figure 1: CS128M RNG Construction Model

2. CS128M Entropy Source.

The CS128M entropy source is modeled in accordance with both standards (See **Part 1** for an in-depth description of the entropy source model and entropy estimation). The entropy source sub-boundary contains three redundant noise sources and a health test component. The health test component includes start-up tests and continuous tests on the noise sources.

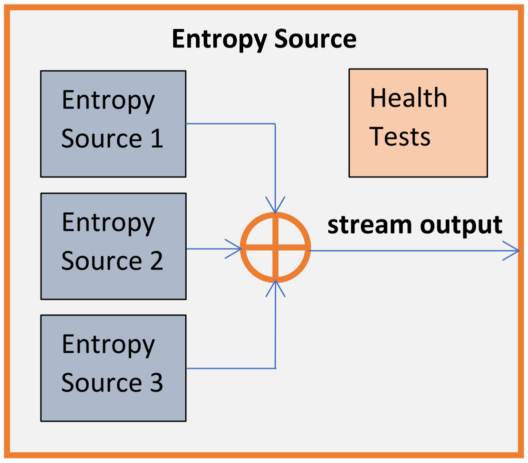


Figure 2: CS128M Entropy Source Sub-boundary

2.1 NIST SP 800-90B Requirements.

This section describes the steps taken to comply with NIST SP 800-90B recommendations for entropy sources. NIST SP 800-90B requires the following implementation:

1) Entropy model validation.
2) Health tests of the internal raw data.
    a. Continuous and start-up tests that meet NIST detection requirements.
        i. Both tests must run over at least 1024 consecutive samples.
        ii. Tests performed on noise sources before any conditioning or post-processing.
    b. On-demand health tests.
    c. The source shall notify the consuming application and halt the output when health test fails.

2.1.1 CS128M Entropy Source Model and Estimation.

See **Part 1** for a comprehensive description of the entropy source model and calculation.

2.1.2 CS128M Entropy Source Health Tests Implementation.

NIST provides two approved health tests: the Repetition Count Test and the Adaptive Proportion Test. NIST allows developer-defined tests that meet the requirements for a substitution of those approved tests. The goal of the Repetition Count Test is to quickly detect catastrophic failures that cause the noise source to be stuck on a single output. The Adaptive Proportion Test is designed to detect a large loss of entropy that might occur from the result of some physical failure or environmental change that affects the noise sources.

Developer-defined alternative tests were implemented that meet the requirements of the NIST approved tests.

1) The startup and continuous monitoring include 1/0 bias, 1st order autocorrelation and an estimated entropy of each of these three sources and the final output.
    a. At startup, random data will not be output until a block of 1,048,576 bits (2^{20} bits) from at least two of the three redundant entropy sources has produced the required minimum estimated entropy level.
    b. Monitoring is continuously run for every subsequent block of 1,048,576 bits.
2) In addition to the startup and continuous testing, the three independent entropy sources raw output data streams and the final combined raw output stream are made available on-demand, offline for direct statistical testing.
3) The 1/0 bias and 1st order autocorrelation tests detect total failure of each of these noise sources and final output stream. The internal hardware monitoring requires at least two of the three entropy sources to have estimated entropy of at least 0.999 bits/bit. If this requirement fails, the output from the RNG is automatically halted. Output bits are also tested for entropy, and the RNG will be halted if the output estimated entropy falls below 0.999 bits/bit.

2.2 AIS 20/31 PTG.3 Entropy Source Requirements.

This section describes the steps taken to comply with AIS 20/31 PTG.3 requirements for entropy source. The class PTG.3 defines requirements for RNGs that must include PTG.1 and PTG.2 definitions with the addition of a cryptographic post-processing algorithm that is DRG.3-conformant (discussed in Section 4.2.1). AIS 20/31 entropy source component requires:

1) A stochastic model of the physical RNG that quantifies the distribution of random numbers.
2) Total failure test of the entropy source.
3) Online tests of raw random numbers and internal random numbers (conditioned).

2.2.1 CS128M Stochastic Model.

See **Part 1** for an in-depth description of the entropy source model and estimation.

2.2.2 CS128M Entropy Source Internal Tests Implementation

Section 2.1.2, CS128M Entropy Source Health Tests, describes the internal tests implemented to detect total failure of the entropy source raw numbers, and online tests that detect non-tolerable defects that may be affected by some physical failure or environmental change. The CS128M entropy source does not require any whitening or conditioning algorithms, therefore no conditioned random numbers are required to be tested.

3. CS128M Entropy Pool.

The entropy pool sub-boundary is a circular buffer implemented to transfer entropy from an approved entropy source to the DRNG and final RNG output. The entropy pool is accessed by the DRNG for seeding initialization/reseeding of its internal states, and accessed by the RNG for generating the final output by XORing 128 bits of fresh full entropy with the DRNG output cipher block (128 bits). The data fills on demand by either DRNG or final RNG XOR output request. During health test mode, the Known-Answer Test module of the DRNG sub-boundary will feed the entropy pool by injecting test vectors used to seed DRNG (RNG output is disabled during test mode).

4. CS128M DRNG.

The DRNG selected is the NIST approved block cipher DRBG mechanism in the counter mode, CTR_DRBG, using the approved cryptographic algorithm AES with security strength of 256 bits (AES-256). The DRNG sub-boundary contains the secret internal states; the AES block cipher mechanism and a health test component, Known-Answer Test (KAT).

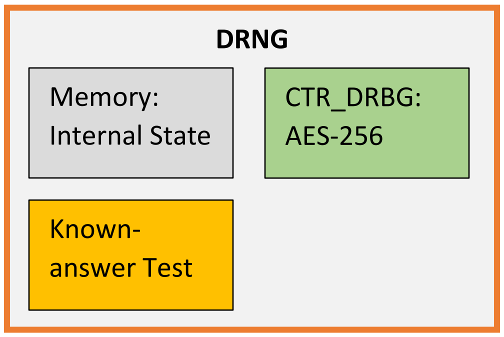


Figure 3: CS128M DRNG Sub-boundary

4.1 NIST SP 800-90A Requirements.

NIST provides a functional model of a DRBG. A DRBG shall implement an approved DRBG mechanism from SP 800-90A and at least one approved randomness source. The DRBG construction includes the following components:

1) Entropy Input source follows NIST 800-90B Recommendations.
    a. Entropy input and the seed shall be kept secret.
    b. The randomness source shall provide input that supports the security strength requested by the DRBG mechanism.
2) Internal state.
    a. Memory of the DRBG and consists of all of the parameters, variables and other stored values that the DRBG mechanism uses or acts upon.
    b. The internal state contains both administrative data (e.g., the security strength) and data that is acted upon and/or modified during the generation of pseudorandom bits (i.e., the working state).
3) DRBG Mechanism.
    a. The instantiate function acquires entropy input to create a seed from which the initial internal state is created.
    b. The generate function generates pseudorandom bits upon request, using the current internal state.
    c. The update function generates new internal state for the next output request.
    d. The reseed function acquires new entropy input and combines it with the current internal state to create a new seed and a new internal state.
    e. The uninstantiate function zeroizes (i.e., erases) the internal state.
    f. The health test function determines that the DRBG mechanism continues to function correctly.

4.1.1 CS128M DRNG Entropy Input.

The CS128M DRNG entropy input source follows NIST 800-90B Recommendations. The entropy source provides randomness that supports the security strength requested by the DRNG mechanism without any entropy conditioning, personalization string or additional input (derivation function). The entropy source input and the seed are always kept secret. (See Section 2. CS128M Entropy Source)

4.1.2 CS128M DRNG Internal State and Mechanism Implementation.

The CS128M DRNG internal state and cryptographic mechanism were implemented to meet NIST 800-90A requirements as described below:

1) CS128M CTR_DRBG DRNG uses the AES cryptographic primitive with security strength of 256 bits (AES-256) in counter mode.
2) DRNG is instantiated with random input from the entropy source prior to output generation.
3) DRNG is reseeded with fresh entropy from the entropy source a little over 15 times per second.
4) The output rate (generate function) of the DRNG does not exceed its input data rate, for every 128 bits input, an output of 128 bits is generated.

5) Known-answer test (KAT) is implemented within the DRNG mechanism boundary.
    a. Testing is conducted on each DNRG function prior to the first use (at boot-up after initial entropy health test is passed) and immediately prior to each reseeding of the internal states (reseed flag triggers KAT, which is run before reseed function). The KAT is always run prior to reseeding the internal states to frequently ensure reliability of the output.
    b. If the DRNG fails the KAT, or a catastrophic error is detected during boot-up or normal operation, the DRNG enters an error state and output is halted. When in this error state, user intervention (power cycling of the device) is required to exit the error state, and the DRNG will re-instantiate before producing new output (as long as all health tests pass again).
    c. KAT test vectors used were acquired directly from NIST at https://csrc.nist.gov/projects/cryptographic-algorithm-validation-program/random-number-generators.

The DRNG has fixed parameters without the option for user input:

Definitions for the CS128M CTR_DRBG	**AES-256**
Security Strength	256
Input and output block length (*blocklen*)	128
Key length (*keylen*)	256
Seed Length (*seedlen* = *blocklen* + *keylen*)	384
Min/Max entropy input length	*seedlen*
Number of bits per output	128
Number of outputs between reseeds (*reseed_interval*)	65536

Table 5

4.2 AIS 20/31 DRG.3-Compliant DRNG Requirements.

The class DRG.3 defines requirements for deterministic RNGs. The DRG.3 functional security requirements are defined below:

1) Initiated with random seed from a PTRNG of class PTG.2 as random source.
2) DRNG provides forward secrecy.
3) DRNG provides backward secrecy.
4) DRNG provides enhanced backward secrecy.
5) DRNG output rate cannot exceed input rate.
6) Known-answer test of the cryptographic post-processing algorithm to detect whether the algorithm was implemented and/or continues to operate correctly. Random number output is not allowed if failure is detected.

4.2.1 CS128M DRG.3-compliant DRNG Mechanism.

The AIS 20/31 standard provides a list of suitable DRNG implementation examples such as the NIST SP 800-90A approved methods [AIS 20/31, pp. 120-129]. The selected NIST CTR_DRBG DRNG meets and exceeds the requirements of class DRG.3.

1) The DRNG is initiated with an approved PTRNG of class PTG.2 as random source. See Section 2.2.1 for entropy source implementation.
2) The NIST CTR_DRBG AES-256 DRNG's inherited design provides backward and forward secrecy due to its one-way output function implementation.
3) Enhanced backward secrecy as required by DRG.3 specifications is provided by selecting an approved NIST SP 800-90A DRBG such as the one implemented in the design. All DRBG mechanisms in the SP 800-90A Recommendation have been designed to provide backtracking resistance [SP 800-90A1, p. 23].
4) Although enhanced forward secrecy is not required by PTG.3 specifications, periodic reseeding of the internal states with fresh entropy was implemented to offer additional defense against attacks and hardening of the RNG design. The CS128M construction went beyond the minimum requirements of the PTG.3 specification by combining the PTG.3 class with a DRG.4-compliant DRNG. Note, implementing the DRNG as DRG.4-compliant does not mean the DRNG output is 'extended' in this design.
5) The output rate of our DRNG does not exceed the input rate, as specified by DRG.3 requirements.
6) KAT implemented to test the cryptographic post-processing algorithm (See Section 4.1.2).
7) Finally, the NIST XOR-NRBG construction XORs the DRNG output with fresh entropy for every output of the RNG, which offers the highest level of reliability and protection against attacks.

PART 3: VERIFICATION TESTING.

ComScire QNGmeter: Continuous Random Number Tester.

The ComScire QNGmeter is a continuous real-time statistical tester that uses five powerful and fundamentally different tests on the input data. Unlike other statistical test suites, it is designed to measure the quality of randomness of a continuous sequence of bits up to hundreds of terabits in length. The QNGmeter automatically performs metatests of subsequences, which would have to be done manually by other popular test suites. Every QNG Model CS128M is tested extensively after production and finally just before shipment using the QNGmeter test suite.

The five tests are:

1) 1/0 Balance – nominal expected value is $p(1) = p(0) = 0.5$.
2) Auto Correlation - orders 1 through 32, nominal expected value is 0.5 for all orders.
3) Entropy Test – nominal expected value is H = 1.0, an update of U. Maurer's "Universal Test" [Cor99].
4) Serial Test - (Good, I. J, The serial test for sampling numbers and other tests for randomness, *Proc. Camb. Philos. Soc.* Vol. 49, 1953).
5) OQSO – Overlapping-Quadruples-Sparse-Occupancy test, nominal expected value for the mean = 141909.47 and standard deviation (by simulation) = 294.656 (G. Marsaglia and A. Zaman, *Computers Math. Applic.*, Vol. 26, No. 9, pp 1-10, 1993).

The z-scores, p-values, and chi-square (metatest) p-values are presented for each test. In addition, current test run time information, such as *Bits Tested*, *Elapsed Time*, *Throughput*, and *Bits Tested %,* is displayed by the tester. *Bits Tested* is the total number of bits tested. *Elapsed Time* is the time from the start of the current test run. *Throughput* is the input data rate in bits per second. *Bits Tested %* is the percent of the total bits tested. This value might be less than 100% due to limited CPU resources.

Each test uses blocks of data of varying lengths, depending on the specific test. The 1/0 Balance and Auto Correlation tests use a block size of 65,536 bits. The Serial test has a block size of 262,144 bits. The Entropy test has 4,194,304 bits in a block. The OQSO test uses 10,485,775 bits per block.

A z-score is calculated for every test for each data-block. The z-scores are converted to probabilities with the assumption they are normally distributed. The z-scores of the 1/0 Balance, Auto Correlation and Serial tests and their associated p-values displayed are cumulative for all blocks. The z-scores of the Entropy and OQSO tests are combined by summing the z-scores of all blocks and dividing by the square root of the number of blocks, respectively.

A second level of testing is applied to the p-values calculated from the z-scores for each block of data. The z-scores are expected to be normally distributed and their associated p-values are expected to be uniformly distributed. A chi-square test is applied to the individual p-values from each of the five tests. The chi-square tests are cumulative and their results are displayed as probabilities. If these chi-square p-values converge to 0.0 or 1.0 for any test, the assumption of randomness fails, indicating non-random patterns in the data being tested.

A third level of testing is applied to all of the individual chi-squared tests. A Kolmogorov-Smirnov (KS) test is first applied to the probabilities of chi-squared results of all orders of auto correlation being tested to reduce the auto correlation results to a single probability. A meta-meta

KS test is finally calculated using the auto correlation KS result and the probabilities of the chi-squared metatest results of all the other tests. The meta-meta KS+ and KS- probabilities are displayed. Convergence toward 1.0 or 0.0 indicates failure.

For the hardware validation report, the QNGmeter tests were completed on a QNG Model CS128M using 710 trillion random bits. All metatest results for the device are recorded in the following Table 6.

ComScire QNGmeter 710 Trillion Bits Tested Testing QNG Device S/N QWR80001			
Run Time Information		**Autocorrelation**	
Bits Tested	**710E+12**	**Order**	**p $(\chi^2 \leq x)$**
Time Elapsed	**153:05:26:23**	**1**	**0.411814**
Throughput	**128E+6**	**2**	**0.069432**
Meter	**47.3+**	**3**	**0.121966**
1/0 Balance		**4**	**0.113409**
p (1)	**0.5000000038**	**5**	**0.205724**
p $(z \leq x)$	**0.578920**	**6**	**0.449533**
p $(\chi 2 \leq x)$	**0.965719**	**7**	**0.651279**
Entropy Test		**8**	**0.466550**
H	**1.0000000216**	**9**	**0.312043**
p $(z \leq x)$	**0.922749**	**10**	**0.432477**
p $(\chi^2 \leq x)$	**0.109109**	**11**	**0.315501**
Serial Test		**12**	**0.362202**
p $(z \leq x)$	**0.709419**	**13**	**0.042787**
p $(\chi^2 \leq x)$	**0.091264**	**14**	**0.313442**
OQSO (Monkey Test)		**15**	**0.310035**
p $(z \leq x)$	**0.067075**	**16**	**0.636126**
p $(\chi^2 \leq x)$	**0.366756**	**17**	**0.936125**
AC Meta KS- Test		**18**	**0.634828**
KS-	**0.277938**	**19**	**0.081837**
Meta-Meta KS Test		**20**	**0.285263**
KS+	**0.885590**	**21**	**0.422316**
KS-	**0.306019**	**22**	**0.733124**
		23	**0.879618**
		24	**0.697534**
		25	**0.996296**
		26	**0.539955**
		27	**0.336305**
		28	**0.245446**
		29	**0.528274**
		30	**0.637863**
		31	**0.861949**
		32	**0.836153**

Table 6 – QNGmeter continuous test results for CS128M.

NIST Statistical Test Suite for the Validation of Random Number Generators.

The National Institute of Standards and Technology (NIST) provides a statistical testing suite, specified in Special Publication 800-22rev1a, consisting of 15 tests that were developed to test the randomness of binary sequences generated by a TRNG or PRNG. The NIST Statistical Test Suite (NIST STS) software and documentation can be downloaded from their Cryptographic Toolkit web page.

The NIST STS source code was compiled on a computer running Ubuntu 18.04. A number of tests were completed to confirm the functionality of the software. The test suite contains sample data files of 1,000,000 bits in length to be analyzed. These include the binary expansions of constants *e*, π, √2 and √3. For each sample file, the NIST STS battery of tests were performed and compared to the empirical results found in the SP800-22rev1a documentation Appendix B. Following the confirmation that the test suite is operating properly, a binary file of 1 billion raw random bits in length was generated using our QNG Model CS128M (SN: QWR80001) to be analyzed.

All test results are recorded in the following Table 2. The Block Frequency, Non-overlapping Template Matching, Overlapping Template Matching, Approximate Entropy, Linear Complexity and Serial tests require user prescribed input parameters. The exact values used in these examples have been included in parenthesis beside the name of the statistical test. In the case of the Non-overlapping Templates test, a Kolmogorov-Smirnov test (KS-test) was performed for the collection of 148 *P-values*. In the case of the Random Excursions and Random Excursions Variant tests, KS-tests for the collection of 8 and 18 *P-values*, respectively, have been reported.

NIST Battery of Tests Results	
Statistical Test	**P-value**
Frequency	0.630872
Block Frequency (m = 128)	0.781106
Cumulative Sums-Forward	0.809249
Cumulative Sums-Reverse	0.268917
Runs	0.137282
Long Runs of Ones	0.058243
Rank	0.666245
Spectral DFT	0.074791
Non-overlapping Templates (m = 9)	0.138219
Overlapping Templates (m = 9)	0.408275
Universal	0.200115
Approximate Entropy (m = 10)	0.878618
Random Excursions	0.892380
Random Excursions Variant	0.133669
Linear Complexity (m = 500)	0.570792
Serial (m = 16, $\nabla\Psi^2{}_m$)	0.188601
Serial (m = 16, $\nabla^2\Psi^2{}_m$)	0.122325

Table 7 — NIST Test Suite Results for CS128M.

NIST SP 800-90B Entropy Source Validation.

NIST Special Publication (SP) 800-90B provides a standardized process of validating the entropy source quality. The process includes the following steps:

1) Data Collection
2) Determine the track (IID or Non-IID)
3) Initial Entropy Estimate
4) Restart Tests
5) Update Entropy Estimate
6) Entropy Validation

NIST offers software for the initial entropy estimation, restart tests and update entropy estimation. The source code and documentation is available from NIST GitHub repository[15]. The source code was compiled on a computer running Ubuntu 18.04. The included self-test was performed to confirm the functionality of the software.

1. Data Collection.

A sequential dataset of at least 1,000,000 samples must be obtained directly from the noise source to determine the initial entropy estimate. If the generation of 1,000,000 consecutive samples is not possible, the concatenation of several smaller sets of consecutive samples from the same source is allowed. Smaller sets shall contain at least 1,000 samples.

For the restart tests, the entropy source must be restarted 1,000 times; for each restart, 1,000 samples shall be collected.

2. Determine Entropy Track.

Entropy estimation is completed based on selecting from two different tracks: IID and non-IID. The IID-track applies for entropy sources that provide IID (independent and identically distributed) numbers, whereas the non-IID track applies for entropy sources that do not provide IID numbers.

The CS128M entropy source provides IID numbers (see **Part 1**).

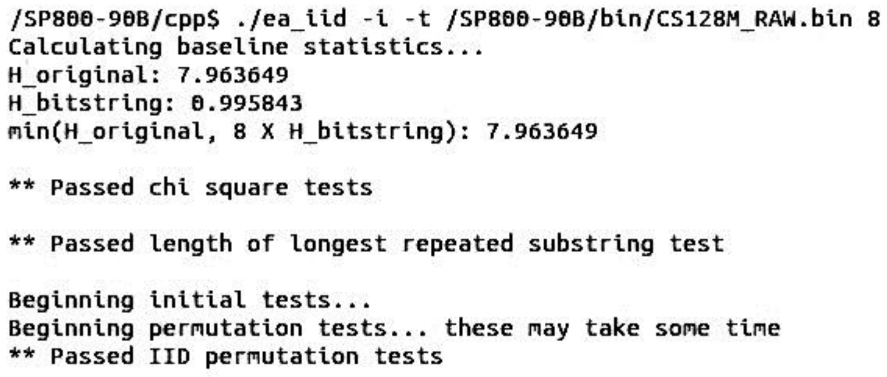
3. Initial Entropy Estimate.

The submitter shall provide an entropy estimate, denoted as $H_{submitter}$, for the noise source outputs, which is based on the submitter's analysis of the noise source. See **Part 1** for in-depth submitter entropy estimation. After determining the entropy estimation track, a min-entropy estimate of the collected sequential dataset of 1,000,000 samples, denoted as $H_{original}$, is calculated using the NIST software. Then, the initial entropy estimate is determined as $H_I = \min(H_{original}, H_{submitter})$. Submitter entropy estimate, NIST initial entropy estimate, the initial min-entropy estimate, and additional statistical tests results are reported in Table 8. Figure 4 is a screenshot of the actual test run.

[15] https://github.com/usnistgov/SP800-90B_EntropyAssessment

NIST SP 800-90B Entropy Assessment	
Initial Entropy Estimate	
Statistical Test	**Results**
$H_{submitter}$	8.000000
$H_{original}$	7.963649
H_I = min ($H_{original}$, $H_{submitter}$)	7.963649
Chi Square Tests	PASS
Length of Longest Repeated Substring Test	PASS
IID Permutation Tests	PASS

Table 8 — NIST Initial Entropy Estimate for CS128M.


```
/SP800-90B/cpp$ ./ea_iid -i -t /SP800-90B/bin/CS128M_RAW.bin 8
Calculating baseline statistics...
H_original: 7.963649
H_bitstring: 0.995843
min(H_original, 8 X H_bitstring): 7.963649

** Passed chi square tests

** Passed length of longest repeated substring test

Beginning initial tests...
Beginning permutation tests... these may take some time
** Passed IID permutation tests
```

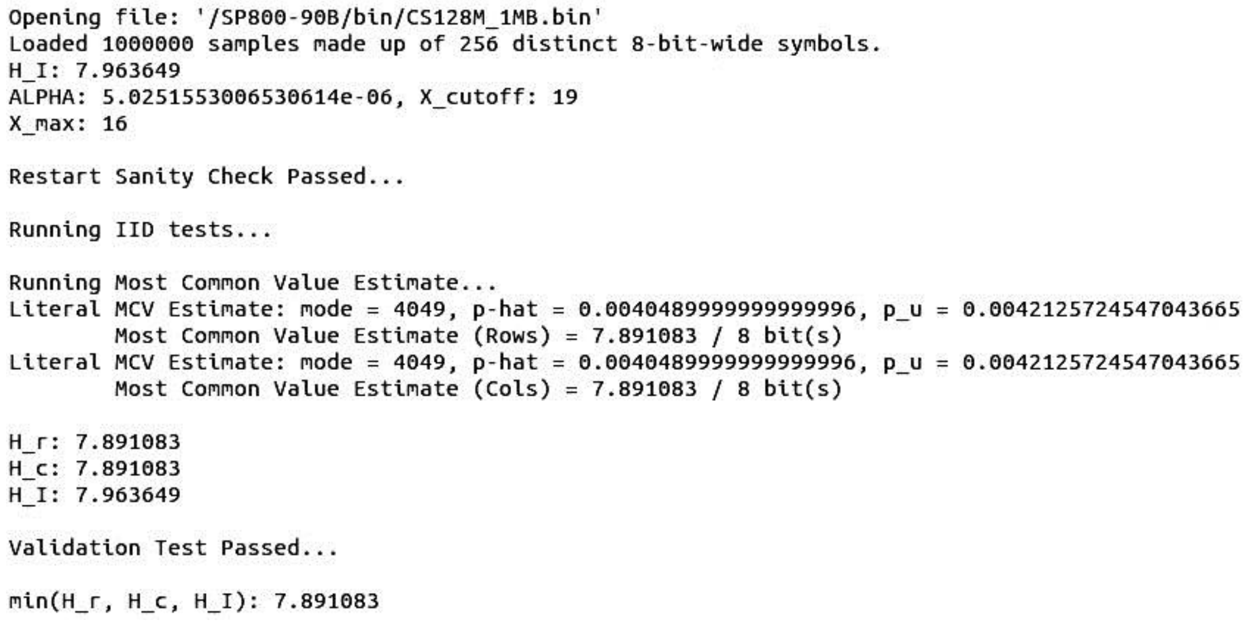
Figure 4: NIST IID-Track Initial Entropy Estimate Test for CS128M

4. Entropy Validation: Restart Tests and Update Entropy Estimate

The restart tests re-evaluate the entropy estimation for the noise source using different outputs from many restarts of the noise source. A matrix *M* of row *r* =1,000 and column *c* = 1,000 is constructed from the collection of restart samples. Sanity check is performed on the matrix *M* prior to calculating entropy estimates on the row and column datasets. The entropy estimates from the row (H_r) and the column (H_c) datasets are expected to be close to the initial entropy estimate H_I. If the minimum of H_r and H_c is less than half of H_I, the validation fails, and no entropy estimate is awarded. Otherwise, the entropy assessment of the noise source is taken as the minimum of the row, the column and the initial estimates, i.e., min (H_r, H_c, H_I). The results are presented in Table 9. Figure 5 is a screenshot of the actual test run.

NIST SP 800-90B Entropy Assessment	
Restart Tests and Update Entropy Estimate	
Statistical Test	**Results**
H_I	7.963649
H_r	7.891083
H_c	7.891083
min (H_r, H_c, H_I)	7.891083
Restart Sanity Check	PASS
Entropy Validation Test	PASS

Table 9 — NIST Validation at Entropy Estimate for CS128M.



```
Opening file: '/SP800-90B/bin/CS128M_1MB.bin'
Loaded 1000000 samples made up of 256 distinct 8-bit-wide symbols.
H_I: 7.963649
ALPHA: 5.0251553006530614e-06, X_cutoff: 19
X_max: 16

Restart Sanity Check Passed...

Running IID tests...

Running Most Common Value Estimate...
Literal MCV Estimate: mode = 4049, p-hat = 0.0040489999999999996, p_u = 0.0042125724547043665
        Most Common Value Estimate (Rows) = 7.891083 / 8 bit(s)
Literal MCV Estimate: mode = 4049, p-hat = 0.0040489999999999996, p_u = 0.0042125724547043665
        Most Common Value Estimate (Cols) = 7.891083 / 8 bit(s)

H_r: 7.891083
H_c: 7.891083
H_I: 7.963649

Validation Test Passed...

min(H_r, H_c, H_I): 7.891083
```

Figure 5: NIST Restart Tests and Entropy Validation for CS128M

BSI AIS 31: Standard Statistical Test Suite.

The BSI AIS 31 Standard Statistical Test Suite consists of nine independent tests to examine the randomness of binary sequences generated by the entropy source and the cryptographic post-processing algorithm. The evaluation process is broken into two test procedures, A and B. Test procedure A (Tests T0-T5) is applied to the post-processed final output of the RNG, or internal random numbers. Test procedure B (T6-T8) is applied to the raw output data of the entropy source. The goal is to ensure that the entropy per bit is sufficiently large prior to seeding the post-processing algorithm.

The complete testing suite, including documentation and software, can be downloaded directly from the BSI website[16]. A JAVA program is provided for simple use of the testing suite. The AIS 31 tests require large binary files of raw and internal random numbers, at least 3,145,728 bits for Test T0 and 5,140,000 bits for Tests T1-T5, to be tested. Binary files of raw and internal random data of 2 GB in length each were generated using our QNG Model CS128M (SN: QWR80001) to be analyzed.

For the generated random data file all of the statistical tests were applied and the result recorded in the following Table 10. In the case of the Test T8, Entropy Test, the bits of entropy per byte has been reported.

BSI AIS 31 Battery of Test Results	
Statistical Tests	**Results**
T0 – Disjointness Test	PASS
T1 – Monobit Test	PASS
T2 – Poker Test	PASS
T3 – Runs Test	PASS
T4 – Long Run Test	PASS
T5 – Autocorrelation Test	PASS
T6 – Uniform Distribution Test	PASS
T7 – Comparative Test for Multinomial Distributions	PASS
T8 – Entropy Test	PASS
T8 – Entropy Estimation (bits of entropy per byte)	7.997572

Table 10 — AIS 31 Test Suite Results for CS128M.

[16] https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Zertifizierung/Interpretationen/AIS_31_testsuit_zip.zip

DIEHARD: A Battery of Tests of Randomness.

The DIEHARD Battery of Tests of Randomness, developed by Prof. George Marsaglia, contains a collection of 15 tests to examine the randomness of binary sequences generated by a TRNG or PRNG. The complete testing suite, including documentation and software, can be found from the DIEHARD archived website[17]. Windows executable files are provided for simple use of the testing suite. The DIEHARD tests require a large binary file of random integers, at least 80 million bits, to be tested. Therefore, a binary file of 80 million raw random bits in length was generated using our QNG Model CS128M (SN: QWR80001) to be analyzed.

For the generated random data file all of the statistical tests were applied and the resulting *p-values* recorded in the following Table 11. In the case of the Birthday Spacings, Binary Rank (6x8 matrices), OPSO, OQSO, DNA, Count-the-1's (specified bytes), This is a Parking Lot, The Minimum Distance, 3DSpheres, Overlapping Sums, and Runs (up & down) tests, only the K-S tests are reported here.

DIEHARD Battery of Tests Results	
Statistical Test	**p-value**
Birthday Spacings	0.544478
Overlapping 5-Permutation	0.627684
Binary Rank (31x31)	0.715902
Binary Rank (32x32)	0.977584
Binary Rank (6x8)	0.362809
Bitstream	0.255699
OPSO	0.083799
OQSO	0.788372
DNA	0.388282
Count-the-1's (byte stream)	0.848490
Count-the-1's (specified bytes)	0.822120
This is a Parking Lot	0.016155
The Minimum Distance	0.255680
3DSpheres	0.910887
Squeeze	0.067427
Overlapping Sums	0.501066
Runs (up)	0.646677
Runs (down)	0.217652
Craps (no. of wins)	0.919992
Craps (throws/game)	0.662403

Table 11 — DIEHARD Test Suite Results for CS128M.

[17] https://web.archive.org/web/20160113163414/http://stat.fsu.edu/pub/diehard/diehard.zip

Bibliography.

[AIS 20/31] A proposal for: Functionality Classes for Random Number Generators, *BSI AIS 20 / AIS 31 Version 2.0*, Killmann, W., and Schindler, W., September, 2011, https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Zertifizierung/Interpretationen/AIS_31_Functionality_classes_for_random_number_generators_e.pdf?__blob=publicationFile&v=1

[Cim19] Cimpanu, C., SHA-1 collision attacks are now actually practical and a looming danger, *Zero Day*, May 13, 2019, Downloaded 11/20/19, https://www.zdnet.com/article/sha-1-collision-attacks-are-now-actually-practical-and-a-looming-danger/

[Cor99] Coron, J.-S., On the Security of Random Sources. In H. Imai, & Y. Zeng (Eds.), *Lecture Notes in Computer Science*, Vol. 1560, pp. 29-42, Springer-Verlag, 1999, http://www.jscoron.fr/publications/entropy.pdf

[Dav02] Davies, R., Exclusive OR (XOR) and hardware random number generators, Feb. 28, 2002, Retrieved Nov. 24, 2019 from http://www.robertnz.net/pdf/xor2.pdf

[Per92] Peres, Y., Iterating von Neumann's Procedure for Extracting Random Bits. *The Annals of Statistics*, pp590-597, Vol. 20(1), 1992. https://projecteuclid.org/download/pdf_1/euclid.aos/1176348543

[San86] Santha, M., & Vazirani, U., Generating Quasi-Random Sequences from Semi- Random Sources, *Journal of Computer and System Sciences*, Vol. 83, pp. 75-87, 1986, http://www.eecs.berkeley.edu/~vazirani/pubs/quasi.pdf

[Sar19] Sardi, S., Uzan, H., *et al*, Embedding information in physically generated random bit sequences while maintaining certified randomness, *Europhysics Letters*, 127(6), Nov. 5, 2019. https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&ved=2ahUKEwiY9Zy55YPmAhWN9Z4KHSDwD-wQFjABegQIAhAC&url=https%3A%2F%2Farxiv.org%2Fpdf%2F1911.00001&usg=AOvVaw1BjXvy7xpcdfHBh5A9Ayrx

[Sha48] Shannon, C., A Mathematical Theory of Communication, *Bell System Technical Journal*, Vol. 27, 379-423; 623-656, Jul.; Oct., 1948, https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=7&ved=2ahUKEwi4kuvp04PmAhUYvJ4KHXQPDpoQFjAGegQIBRAC&url=http%3A%2F%2Fwww.nd.edu%2F~powers%2Fame.20231%2Fshannon1948a.pdf&usg=AOvVaw1M_h1uOvsVZq0CM-7WyMkH

[Skó15] Skórski, M., Shannon Entropy versus Renyi Entropy from a Cryptographic Viewpoint, *15th IMA International Conference on Cryptography and Coding*, Dec., 2015, https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=2ahUKEwj1vuTTkevlAhXW7Z4KHX81CDsQFjAAegQIBhAC&url=https%3A%2F%2Feprint.iacr.org%2F2014%2F967.pdf&usg=AOvVaw1RSA_TEjcGUFLeR_97k9b8

[SP 800-133] Recommendation for Cryptographic Key Generation, *NIST Special Publication 800-133*, Barker, E. and Roginsky, A., Dec., 2012, http://dx.doi.org/10.6028/NIST.SP.800-133

[SP 800-90B] Recommendation for the Entropy Sources Used for Random Bit Generation, *NIST Special Publication 800-90B*, Turan, M. S., Barker, E., et al, Jan., 2018, https://doi.org/10.6028/NIST.SP.800-90B

[SP 800-90A1] Recommendation for Random Number Generation Using Deterministic Random Bit Generators, *NIST Special Publication 800-90A Rev. 1*, Barker, E., and Kelsey, J., June, 2015, http://dx.doi.org/10.6028/NIST.SP.800-90Ar1

[SP 800-90C] Recommendation for Random Bit Generator (RBG) Constructions, *DRAFT NIST Special Publication 800-90C*, Barker, E., & Kelsey, J., 1-45, Aug., 2012, http://csrc.nist.gov/publications/drafts/800-90/draft-sp800-90c.pdf

[Ste01] Steele, J., Random Walks and First Step Analysis, pp. 1-9. In: *Stochastic calculus and financial applications*, Springer-Verlag New York, NY, I. Karatzas & M. Yor, Eds., 2001.

[Wil05] Wilber, S., U.S. Patent No. 6,862,605 B2, Mar., 2005, http://www.freepatentsonline.com/6862605.pdf

[Wil13] Wilber, S. A., Entropy Analysis and System Design for Quantum Random Number Generators in CMOS Integrated Circuits, July 5, 2013, Downloaded Dec. 5, 2019, https://comscire.com/files/whitepaper/Pure_Quantum_White_Paper.pdf