

Wolfgang Killmann
T-Systems GEI GmbH, Bonn

Werner Schindler
Bundesamt für Sicherheit in der Informationstechnik (BSI), Bonn

A proposal for:
Functionality classes for random number generators¹

Version 2.0

18 September 2011

¹ The authors wish to express their thanks for the numerous comments, suggestions and notes that have been incorporated into this document.

Table of contents

1.	Introduction	7
1.1.	Motivation	7
1.2.	Abbreviations	8
1.3.	Common Criteria (Abbreviations).....	8
1.4.	Terminology	9
1.5.	Symbols.....	16
2.	Basic Concepts	18
2.1.	Randomness	18
2.1.1.	Concept of Randomness and Random Experiments	18
2.1.2.	Random number generators (RNGs)	19
2.2.	Random Numbers in IT Security.....	21
2.2.1.	Usage of Random Numbers in IT Security.....	21
2.2.2.	Basic considerations for RNG types	23
2.2.3.	Design Description of RNG.....	24
2.3.	Mathematical Background	28
2.3.1.	Random variables	28
2.3.2.	Entropy and Guess Work.....	31
2.3.3.	Random mappings	34
2.4.	Stochastics and Statistical Analysis of Physical RNGs.....	36
2.4.1.	Stochastic model	36
2.4.2.	Overview of Statistical Tests	41
2.4.3.	Standard Statistical Tests	44
2.4.4.	Test procedures	54
2.4.5.	Additional Statistical Tests	57
3.	Security Functional Requirements - Family FCS_RNG.....	61
3.1.	Definition of FCS_RNG.....	61
3.2.	Security capabilities of RNG types	62
3.3.	Rationale for definition of the extended component	66
4.	Pre-defined RNG Classes.....	67
4.1.	Overview of pre-defined RNG classes	67
4.2.	General Remarks (Exemplary applications, side-channel attacks, fault attacks)	71
4.3.	Class PTG.1.....	71
4.3.1.	Security functional requirements for the RNG class PTG.1	71

4.3.2.	Application notes	72
4.4.	Class PTG.2.....	74
4.4.1.	Security functional requirements for the RNG class PTG.2.....	74
4.4.2.	Application notes	75
4.4.3.	Further aspects	77
4.5.	Class PTG.3.....	79
4.5.1.	Security functional requirements for the RNG class PTG.3.....	79
4.5.2.	Application notes	80
4.5.3.	Further aspects	82
4.6.	Class DRG.1	84
4.6.1.	Security functional requirements for the RNG class DRG.1	84
4.6.2.	Application notes	84
4.6.3.	Further aspects	87
4.7.	Class DRG.2.....	88
4.7.1.	Security functional requirements for the RNG class DRG.2.....	88
4.7.2.	Application notes	89
4.7.3.	Further aspects	89
4.8.	Class DRG.3.....	90
4.8.1.	Security functional requirements for the RNG class DRG.3	90
4.8.2.	Application notes	91
4.8.3.	Further aspects	91
4.9.	Class DRG.4.....	91
4.9.1.	Security functional requirements for the RNG class DRG.4.....	91
4.9.2.	Application notes	92
4.9.3.	Further aspects	93
4.10.	Class NTG.1	93
4.10.1.	Security functional requirements for the NPTRNG class NTG.1.....	93
4.10.2.	Application notes	94
5.	Examples	96
5.1.	Guesswork for binomial distributed data	96
5.2.	Contingency tables	99
5.3.	Forward and backward secrecy	103
5.4.	Examples of post-processing algorithms.....	107
5.4.1.	Von Neumann unbiasing	107
5.4.2.	Xoring of non-overlapping segments of independent bits.....	108

5.4.3.	Two sources	108
5.4.4.	Uniformly distributed input data for random mappings	109
5.5.	Examples of online test, tot test, and start-up test	111
5.5.1.	An online test of the internal random numbers	111
5.5.2.	A straightforward online test	112
5.5.3.	A more sophisticated online test procedure	113
5.6.	Examples of RNG designs	116
5.6.1.	PTRNG with two noisy diodes	116
5.6.2.	Examples of DRNGs	120
5.6.3.	NPTRNG	127
6.	Literature	130

Tables

Table 1: Attack potential, guessing probability and security bits.....	22
Table 2: Attack potential and guessing passwords.....	22
Table 3: Statistics of random mappings	34
Table 4: Statistics of random permutations.....	35
Table 5: Brief overview of error types of statistical tests.....	43
Table 6: Typical values of χ^2-distribution with 1 degree of freedom	45
Table 7: Typical values of χ^2-distribution with degree of freedom d	46
Table 8: Typical values of χ^2-distribution for runs.....	47
Table 9: Typical values of Normal (Gaussian) $N(0,1)$ for a two-sided test of autocorrelation	50
Table 10: Parameters for entropy test.....	53
Table 11: Recommended parameter settings for the NIST test suite	57
Table 12: Attack potential, Min-entropy, and recommended length of the internal state.....	85
Table 13: Requirements for the parameters in (DRG.1.3) depending on claimed attack potential.....	87
Table 14: Work factor and work factor defect for uniform mappings with equidistributed input.....	111
Table 15: Probability for a noise alarm within a test suite and the expected number of noise alarms per year for different distributions of the das-random numbers.....	115

Figures

Figure 1: Min-entropy, collision-entropy and Shannon-entropy for binary-valued random variables.....	33
Figure 2: Contingency table for counts of consecutive bits strings.....	59
Figure 3: Example of PTRNGs that belong to the pre-defined classes PTG.1 and PTG.2.....	68
Figure 4: Example of a PTG.3 and NTG.1 that belongs to the pre-defined class PTG.3 and NTG.1.....	69
Figure 5: Examples of DRNGs that belong to the pre-defined classes DRG.1 and DRG.2.....	70
Figure 6: Examples of DRNGs that belong to the pre-defined classes DRG.3 and DRG.4.....	70
Figure 7: Probabilities of vectors of length $n = 10$.....	97
Figure 8: Success probability ($p = 0.55, n = 10$).....	98
Figure 9: Basic design of RNG with noisy diodes.....	117
Figure 10: Variant of the basic design of RNG with noisy diodes	117
Figure 11: Examples of self-protection in PTRNG based on noise diodes.....	120
Figure 12: RGB Functional model defined in [NIST800-90].....	121
Figure 13: Functional design of the Linux NPTRNG	128

1. Introduction

1.1. Motivation

- 1 Random Number Generators (RNG) are incorporated in many IT products and play an important role in numerous cryptographic applications. However, the Information Technology Security Evaluation Criteria (ITSEC) and the Common Criteria (CC) do not specify any uniform evaluation criteria for RNG, nor do their corresponding evaluation methodologies (Information Technology Security Evaluation Manual [ITSEM]) and Common Evaluation Methodology [CEM]) specify such criteria.
- 2 The document is intended for use by developers, evaluators and certifiers.
- 3 Chapter 2 introduces this field, addresses basic concepts, and explains foundations that support the understanding of the remaining parts of this document. Chapter 3 defines a CC family FCS_RNG and the extended component FCS_RNG.1 for description of security functional requirements in protection profiles or security targets. Chapter 4 describes pre-defined classes for physical true, non-physical true, deterministic and hybrid random number generators. It sketches RNG specific information and evidence the developer is expected to provide for the assurance components selected in the ST. The basic concepts and evaluation criteria are illustrated by additional examples in chapter 5.
- 4 All software tools referenced in the following paragraphs are freeware. The statistical calculations may be performed using:
 - The BSI test suite for statistical test procedures A and B, which is available on the BSI website [AIS2031Stat].
 - The NIST test suite and guidance documentation [SP800-22], which is available on the NIST RNG project website describing the implemented tests http://csrc.nist.gov/groups/ST/toolkit/rng/documentation_software.html.
 - The statistics program R, which is available on the website www.r-project.org. There are several books (e.g., [SaHe06], [Prus06], [Ligg07]) describing statistical methods together with R scripts implementing these methods.
- 5 This document updates the previous documents [AIS20An] and [AIS31An] used as the evaluation methodology for RNG in the German CC scheme. The families described in parts 2 and 3 relate to the RNG classes described in [AIS20An] and [AIS31An] as follows (coarse comparisons):

RNG class	Comparable to [AIS20] or [AIS31] class	Comments
PTG.1	AIS31, P1	Physical RNG with internal tests that detect a total failure of the entropy source and non-tolerable statistical defects of the internal random numbers
PTG.2	AIS31, P2	PTG.1, additionally a stochastic model of the entropy source and statistical tests of the random raw

RNG class	Comparable to [AIS20] or [AIS31] class	Comments
		numbers (instead of the internal random numbers)
PTG.3	No counterpart	PTG.2, additionally with cryptographic post-processing (hybrid PTRNG)
DRG.1	AIS20, K2, partly K3	DRNG with forward secrecy according to [ISO18031]
DRG.2	AIS20, K3	DRG.1 with additional backward secrecy according to [ISO18031]
DRG.3	AIS20, K4	DRG.2 with additional enhanced backward secrecy
DRG.4	No counterpart	DRG.3 with additional enhanced forward secrecy (hybrid DRNG)
NTG.1	No counterpart	Non-physical true RNG with entropy estimation

1.2. Abbreviations

6 In this document we use the following abbreviations:

RNG	random number generator
DRNG	deterministic RNG
TRNG	true RNG
PTRNG	physical true RNG (short: physical RNG) ²
NPTRNG	non-physical true RNG
das	digitized analog noise signal
iid	independent and identically distributed
pp.	pages
iff	if and only if
{x,y,...}	A list x,y,... of indices, e.g., ADV_FSP.{1,2} stands for “ADV_FSP.1 and ADV_FSP.2”

1.3. Common Criteria (Abbreviations)

PP	Protection Profile
ST	Security Target
EAL	Evaluation Assurance Level
ADV	Assurance Development
TOE	Target of Evaluation
TSF	TOE Security Functionality
SFR	Security Functional Requirement

² To avoid misunderstanding, we do not apply the „straightforward“ abbreviation „PRNG“ because this often stands for „pseudorandom number generator“.

1.4. Terminology

7 In this document we use the following terminology:

8 **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.

9 **Bit string**

A finite sequence of ones and zeroes.

10 **Binomial distribution**

Binomial distribution with parameters n and p , $P\{X = k\} = \binom{n}{k} p^k (1-p)^{n-k}$

11 **Black box**

An idealized mechanism that accepts inputs and produces outputs, which is designed such that an observer cannot see inside the box or determine exactly what is happening inside that box. Contrast with a *glass box*.

12 **Cryptographic boundary**

An explicitly defined continuous perimeter that establishes the physical bounds of a cryptographic module and contains all the hardware, software and/or firmware components of a cryptographic module. [ISO/IEC 19790]

13 **Cryptographic post-processing**

A post-processing algorithm that generates the internal numbers of a TRNG by means of a cryptographic mechanism

14 **das-random number**

Bit string that results directly from the digitization of analogue noise signals (das) in a physical RNG. Das-random numbers constitute a special case of raw random numbers.

NOTE: Assume, for instance, that a PTRNG uses a Zener diode. Regular comparisons of the (amplified) voltage (analogue signal) with a threshold value provide values 0 and 1, which may be interpreted as das-random numbers. In contrast, for ring oscillators on FPGAs it is not obvious how to define the analogue signal. At least in the true sense of the word it may be problematic to speak of 'das random number' in this context.

NOTE: In [AIS31An] for physical RNGs the term 'das-random number' was consistently used. Apart from concrete examples in this document we use the more general term 'raw random number' for both physical and non-physical true RNGs.

15 **Deterministic RNG**

An RNG that produces random numbers by applying a deterministic algorithm to a randomly-selected seed and, possibly, on additional external inputs.

16 Digitization

Derivation process of raw random numbers from raw random signals, usually performed at discrete points in time.

17 Endorsed algorithm

Cryptographic algorithm endorsed by a certification body for certified products; that is, either a) specified in an endorsed standard, b) adopted in an endorsed standard and specified either in an appendix of the endorsed standard or in a document referenced by the endorsed standard, or c) specified in the list of Endorsed security functions.

18 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.

19 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.

20 Entropy

A measure of disorder, randomness or variability in a closed system. The entropy of a random variable X is a mathematical measure of the amount of information gained by an observation of X .

21 Entropy source

A component, device or event that generates unpredictable output values which, when captured and processed in some way, yields discrete values (usually, a bit string) containing entropy (Examples: electronic circuits, radioactive decay, RAM data of a PC, API functions, user interactions). Entropy sources provide randomness for true and hybrid random number generators.

22 External random numbers

Random numbers used by an application (usually the concatenation of output random numbers)³.

23 Finite state machine

A mathematical model of a sequential machine that comprises a finite set of admissible states, a finite set of admissible inputs (seed, and possibly additional input or publicly known input), a finite set of admissible outputs, a mapping from the set of inputs and the sets of states to the set of state transitions (i.e., state transition mapping), and a mapping from the set of inputs and the set of states to the set of outputs (i.e., output function).

24 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.

25 Glass box

An idealized mechanism that accepts inputs and produces outputs. It is designed such that an observer can see inside and determine exactly what is going on. Contrast with a *black box*.

26 Human entropy source

An entropy source that includes a random human component (Examples: key strokes, mouse movement).

27 Hybrid RNG

An RNG that applies design elements from DRNGs and PTRNGs; see also *hybrid DRNG* and *hybrid PTRNG*.

28 Hybrid DRNG

A DRNG accepting external input values besides the seed; i.e., a hybrid DRNG uses an additional entropy source. Identical output sequences demand identical seeds and identical external input values.

29 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.

30 Ideal RNG

A mathematical construct that generates independent and uniformly distributed random numbers. An ideal RNG can be described by a sequence of independent identically distributed

³ External random numbers are outside the scope of this document.

random variables X_t , $t \in T$, that are uniformly distributed on a finite set Ω ; in our context, typically $\Omega = \{0,1\}$ or $\Omega = \{0,1\}^c$.

31 Internal random numbers

For DRNGs: values of the output function; for PTRNGs: random numbers after post-processing. The internal numbers are intended to be output upon request by a user.

32 Kerckhoffs' box

An idealized cryptosystem where the design and public keys are known to an adversary, but in which there are secret keys and/or other private information that is not known to an adversary. A Kerckhoffs' box lies between a black box and a glass box in terms of the knowledge of an adversary.

33 Known-answer test

A method of testing the correctness of a deterministic mechanism by checking whether for given input, the mechanism outputs the correct (known) value.

34 Noise alarm

Consequence of an application of an online test that suggests (e.g., due to a failure of a statistical test) that the quality of the generated random numbers is not sufficiently good.

35 Noise source

Special type of entropy source that consists of dedicated hardware (e.g., an electronic circuit) used by PTRNGs.

36 Non-physical true RNG

A true RNG whose entropy source is not dedicated hardware but e.g., provides system data (RAM data or system time of a PC, output of API functions etc.) or human interaction (key strokes, mouse movement, etc.).

37 Normal (Gaussian) distribution

Normal (Gaussian) distribution with mean μ and variance σ^2 , is defined by

$$P\left\{\frac{X - \mu}{\sigma} \leq x\right\} = \int_{-\infty}^x \frac{e^{-u^2/2}}{\sqrt{2\pi}} du.$$

38 One-way function

A function with the property that it is easy to compute the output for a given input but it is computationally infeasible to find for a given output an input, which maps to this output. [ISO/IEC 11770-3].

39 Online test

A quality check of the generated random numbers while a PTRNG is in operation; usually realized by physical measurements, by a statistical test, or by a test procedure that applies several statistical tests.

40 **Pure DRNG**

A DRNG that does not accept any external input apart from the seed. Identical seed values result in identical output sequences (random numbers).

41 **Physical true 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.

42 **Poisson distribution**

Poisson distribution, where λ is the mean number of events per time interval

$$P(X = k) = \begin{cases} \frac{\lambda^k}{k!} e^{-\lambda} & \text{for } k = 0, 1, 2, \dots \\ 0 & \text{else} \end{cases}$$

43 **Post-processing (algorithm)**

Transformation of raw random numbers that have been derived from the entropy source into the internal random numbers

44 **Pure PTRNG**

A PTRNG without (complex) post-processing. A total failure of a pure PTRNG entropy source typically results in constant output or periodic patterns if no post-processing algorithm is implemented, or in outputs of a weak DRNG if a simple mathematical (non-cryptographic) post-processing algorithm is implemented.

45 **P-value**

The p-value quantifies the probability that the test values are at least as extreme as the particular value, which has just been observed (tail probability) if the null hypothesis is true. If this p-value is smaller than a pre-defined bound, the statistician rejects the null hypothesis.

NOTE: Alternatively, a particular significance level α may be defined before the sample is drawn.

46 **Random number generator (RNG)**

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

47 **Random variable**

Mathematical construction that quantifies randomness. A real-valued **random variable** X is a function that assigns to each outcome in the sample space Ω a value of R , i.e., $X : \Omega \rightarrow R$. More precisely, there exist σ -algebras σ_Ω of Ω and σ_R of R for which X is a $(\sigma_\Omega, \sigma_R)$ -measurable function, i.e., for each $r \in \sigma_R$ holds $X^{-1}(r) \in \sigma_\Omega$.

48 **Raw random number**

Raw random numbers are derived at discrete points in time from raw random signals that are generated by the entropy source of a PTRNG or NPTRNG. Raw random numbers have not been post-processed. Raw random numbers assume discrete values.

NOTE: For particular types of TRNGs it may not be unique, which discrete values (normally bits or bit strings) are interpreted as the raw random numbers. The definition of the raw random numbers may influence their distribution. Of course, for the chosen definition the raw random numbers must fulfil the requirements that are specified in the respective functionality class.

NOTE: For many types of physical RNGs raw random numbers are computed from analogue signals that are generated by the entropy source, motivating the notion of das ('digitized analogue signal') random numbers. Examples are PTRNGs that are based on noisy diodes or oscillators. For PTRNGs that are based on ring oscillators on an FPGA, for instance, the term 'analogue signal' is less adequate (cf. the first note to das random numbers).

49 **Raw random number sequence**

Sequence of discrete random values that have directly been derived by digitization from the output of the entropy source; sequence of raw random numbers.

50 **Raw random signal**

Randomly changing signal that is provided by an entropy source of a PTRNG, which is used to generate raw random numbers.

NOTE: In physical experiments and for electronic circuits raw random signals are often time-continuous and assume values in continuous ranges. For a PTRNG on an FPGA that exploits a ring oscillator the current state of the inverter chain with time jitter might be interpreted as a raw random signal.

51 **Realization (of a random variable)**

Value assumed by a random variable.

52 **Refreshing**

Use of fresh entropy provided by an internal or external source of randomness in the state transition function of a hybrid RNG (covers both reseeding and seed-update).

53 **Reseeding**

Re-initialization of the internal state of an RNG (typically, a DRNG), depending on external input (new seed value), but disregarding the current value of the internal state.

54 **Seed**

Value used to initialize the internal state of an RNG.

55 **Seeding procedure**

Procedure for initialization, re-initialization and refreshing of the internal state of a DRNG as described in the guidance documentation.

56 **Secret parameter**

An input value (optional) to the RNG during initialization.

57 **Seed life**

The period between the initialization of the internal state of an RNG (typically, of a DRNG) with a seed value until reseeding / seed-updating the internal state with the next seed value.

58 **Seed-update**

Renewal of the internal state of an RNG (typically, a DRNG) by considering both the current internal state and external input data.

59 **Signal**

Physical carrier of information.

60 **State**

A state is defined as an instantiation of a random number generator or any part thereof with respect to time and circumstance.

61 **Stationary process**

The sequence of random variables X_1, X_2, \dots is called stationary if for all positive integers k and t , and arbitrary (measurable) sets A_j the following equality holds

$$\Pr\{X_1 \in A_1, \dots, X_k \in A_k\} = \Pr\{X_{t+1} \in A_1, \dots, X_{t+k} \in A_k\}.$$

62 **Stochastic model**

A stochastic model is a mathematical description (of relevant properties) of a TRNG using random variables, i.e., a model of the reality under certain conditions and limitations. A stochastic model used for TRNG analysis shall support the estimation of the entropy of the raw random numbers and finally of the internal random numbers. Moreover, it should allow to understand the factors that may affect the entropy.

63 Thermal noise

Inherent production of spurious electronic signals (also known as white noise) within an electronic component (e.g., an operational amplifier, a reversed biased diode, or a resistor)⁴, not desirable for typical applications

64 Total breakdown of an entropy source

The entropy of the future raw random numbers equals 0.

Note: Depending on the concrete RNG design, a total breakdown of the entropy source may result in constant or short-period sequences of raw random numbers.

65 Total failure test of a noise source

The total failure test of the random noise source detects a total breakdown of random noise source.

66 True RNG

A device or mechanism for which the output values depend on some unpredictable source (noise source, entropy source) that produces entropy.

Note: The class of TRNGs splits into two subclasses (PTRNGs and NPTRNGs).

67 Uniform distribution

A random variable X that assumes values on a finite set M is said to have uniform distribution (or equivalently: X is uniformly distributed) on M if $\Pr\{X = m\} = |M|^{-1}$ for each $m \in M$.

1.5. Symbols

68 In this document we use the following symbols:

$A \rightarrow B$ One-way function of A to B

$\Pr\{X=x\}$ Probability that the random variable X assumes the value x

$\Pr\{x\}$ Probability of the value x (short notation if it is clear which random variable is concerned)

$B(n, p)$ Binomial distribution with parameters n and p

$N(\mu, \sigma^2)$ Normal (Gaussian) distribution with mean μ and variance σ^2

⁴ Typically, in electronic circuits a concentrated effort is exerted to minimize these phenomena. However, this exact phenomenon can be taken advantage of in the production of random bit streams as it results in some unpredictable behaviour and, therefore, may be used as an entropy source.

$P o(\lambda)$	Poisson distribution, where λ is the mean of events per time interval
\oplus	Addition in $\text{GF}(2)$, $0 \oplus 0 = 0, 0 \oplus 1 = 1, 1 \oplus 0 = 1, 1 \oplus 1 = 0$
$X \ Y$	Concatenation of two strings X and Y . The strings X and Y are either both bit strings, or both byte strings.
$\lceil X \rceil$	Ceiling: the smallest integer greater than or equal to X , $\lceil X \rceil = \min\{n \in \mathbb{N} \mid X \leq n\}$
$\lfloor X \rfloor$	Floor: the largest integer less than or equal to X , $\lfloor X \rfloor = \max\{n \in \mathbb{N} \mid n \leq X\}$
$ X $	For a finite set X the notation $ X $ denotes its cardinality. If X is a string $ X $ denotes its length.
$SG(S)$	Symmetric group over the set S , i.e., the group of all permutations over S with composition as group operation.
$SH(S)$	Symmetric semi-group over the set S , i.e., the semi-group of all injective (not necessarily surjective) mappings $S \rightarrow S$ with composition as semi-group operation.
$\pi_w(x)$	The projection of a vector $x = (x_0, x_1, \dots, x_n)$ onto the coordinates $w = \{i_1, i_2, \dots, i_{ w }\} \subseteq \overline{1, n}$. That is, $\pi_w(x) = (x_{i_1}, x_{i_2}, \dots, x_{i_{ w }})$.
\mathbb{N}	Set of natural numbers
\mathbb{R}	Set of real numbers
\mathbb{Z}	Set of integers
$\overline{1, n}$	$\{1, \dots, n\}$

2. Basic Concepts

69 This chapter explains basic mathematical concepts that are applied in the security analysis of RNGs. At first, we describe the concept of randomness, which is the “core” for any RNG. For true random sequences, this refers to the entropy source; and for pseudo-random sequences, to the seed. Probability theory describes and analyzes randomness by means of abstract mathematical objects, modelling randomness by random variables and random processes. Statistics links these abstract mathematical models with real-world RNGs by experiments. These experiments may be used to estimate parameters that describe the models or to test hypotheses deduced from the models.

2.1. Randomness

70 Subsection 2.1.1 provides an intuitive notion of randomness, which will be made precise in a mathematical sense in section 2.3.

2.1.1. Concept of Randomness and Random Experiments

71 The core of any non-deterministic (true) random number generator (TRNG) is the entropy source that, loosely speaking, “generates” randomness.

72 An experiment is called **unpredictable** if the observable outcome of the experiment is (to a certain extent) unknown before it is conducted. After the experiment has been performed, the degree of uncertainty depends on the ability to observe the outcome. In this document we denote the outcome of an experiment as **random** if it is unpredictable, i.e., if it cannot be predicted with certainty. Entropy quantifies the amount of unpredictability relative to the observer.

73 Experiments are called **independent** if the outcomes of previous experiments do not influence the outcome of the current experiment.

74 A random experiment is called **unbiased**, if each admissible outcome has the same chance of occurring.

75 Ideal random experiments are unpredictable, independent and unbiased (ideal randomness). Ideal randomness excludes order and regularity in the sequence of outcomes of repeated experiments unless these occur by chance. Any deviation from these properties, i.e., dependency or bias, makes the experiment less random.

76 The goal of any true RNG is clearly to generate ideal random numbers. However, real-world RNGs can only achieve this goal approximately. The key point of any RNG evaluation is to verify to what extent the TOE guarantees fulfillment of this goal.

77 How can we determine to what extent an experiment is random (bias, dependencies)? Assume that an attacker knows the outcomes of many previous experiments. Why should he not be able to guess future outcomes? It is meaningless to argue about randomness on the basis of a single run of an experiment or on the basis of a small number of experiments. The “randomness” of an experiment can only be observed asymptotically. A statistical test that applies a computable function tests the hypothesis of whether the sequence of outcomes is “typical” in some sense. Ideal random sequences belong to any (before the observation of the experiments) reasonably defined “majority” of sequences with overwhelming probability, not showing any regularity

patterns that can be detected by this statistical test⁵. Any finite collection of statistical tests can only check for finitely many types of regularity. A statistical test may not contradict or it may reject the randomness hypothesis under specific assumptions, but this cannot serve as a proof for the randomness of an arbitrary experiment. Testing the randomness of RNG output sequences is computationally hard under “black box assumptions”. Hence, it is important to understand the nature of the random source to rate the randomness of number generation.

2.1.2. Random number generators (RNGs)

- 78 Generally, an RNG consists of a non-deterministic part (entropy source) that generates non-predictable digital data, and a deterministic part that generates from this data the output sequence of the RNG (random numbers). The non-deterministic part of the RNG exploits a physical entropy source or any other kind of non-physical entropy source to generate a raw random number sequence, which is deterministically post-processed. Either the deterministic part or the non-deterministic part may be omitted, giving a pure PTRNG or a pure DRNG, respectively.

PTRNG

- 79 The core of any physical RNG (PTRNG) is the entropy source, which is used to generate the raw random numbers. By exploiting an analogue signal, a digitization mechanism generates a sequence of digital “raw” data (raw random numbers; usually das-random numbers). Additionally, the PTRNG may comprise a post-processing algorithm that transforms the raw data to internal random numbers.

Note that formally a missing post-processing algorithm can be interpreted as the identity mapping.

- 80 Physical entropy sources are based on physical microscopic random processes. Measurements of these processes result in digital random numbers. Examples of time-discrete physical entropy sources are:
- Radioactive atomic disintegration: The number of decay events (detected particles) per time interval follows a Poisson distribution (cf. [Neue04], section 4.1).
 - Shot entropy of a diode: The shot entropy of a parallel-plane temperature-limited diode is non-deterministic. The number of electrons emitted from the tube’s cathode during a time interval follows a Poisson distribution (cf. [DaR087], section 7-2).

The Poisson distribution implies that the inter-occurrence waiting time between consecutive events is exponentially distributed.

- 81 A large number of discrete random events like e.g. emitted electrons may be observed as analogue entropy signal. Examples of analogue physical entropy sources are (cf. [BuLu08] for examples):
- Thermal resistive entropy: The voltage between resistors varies randomly due to vibration of atoms. Ideally, the thermal entropy signal has the same energy in all

⁵ cf. to Chaitin’s definition of random strings and Martin-Löf tests in e.g. [Cal].

frequency bands (so called “white entropy”). Sampling an ideally-amplified white entropy signal generates a sequence of independent bits.

- Diode breakdown entropy: The reverse current through diodes varies randomly due to tunnelling of electrons. The power of the entropy signal is inversely proportional to the frequency.
- Free running oscillators generate digital signals with an edge-to-edge random analogue time drift (jitter). Sampling a fast oscillator by a lower frequency oscillator generates a random bit signal. If the standard deviation of the slow oscillator is considerably greater than the fast period, the sampled bit sequence may be expected to be uncorrelated.

82 A typical goal of algorithmic post-processing may be to extract entropy from the das-random numbers sequence in order to increase the entropy per bit, e.g., to correct a given bias. Note that increasing the entropy per bit demands data compression, reducing the output rate. A cryptographic post-processing algorithm may be viewed as an additional security anchor.

NPTRNG

83 A non-physical true RNG (NPTRNG) uses external signals as entropy source to generate random numbers for output.

84 Examples of such external entropy sources are:

- Processes as disk I/O operations and interrupts (cf. e.g. Linux RNG /dev/random [GuPR06]).
- System data as tick counter since system boot, process and thread IDs, current local time (cf. e. g., function CryptGenRandom of Microsoft Windows CE Enhanced Cryptographic Provider 5.01.01603 [MSCE06]).
- Human interaction as mouse movement and key strokes (cf. PGP key generation [PGP]).

85 The NPTRNG are based on the concept of randomness as lack of information about processes and their outcomes. If a huge amount of data from different sources are collected and mapped onto a shorter sequence (e.g., by a hash function), the output value will appear random to an observer who neither knows the source data nor is able to control them.

DRNG

86 A deterministic RNG (DRNG) generates random numbers with a deterministic algorithm and starts with a randomly selected seed. The output sequence depends on the seed and possibly also on additional external input values.

87 Examples:

- Deterministic random bit generators based on hash functions, as described in [ISO18031], Annex C.
- NIST-recommended DRNG based on hash functions or block ciphers [NIST800-90].

- 88 A DRNG may be viewed as a finite automaton that receives input (seed and possibly also additional external input). The DRNG updates the internal state (possibly also considering additional input) and generates output that depends on the current internal state and possibly on additional input. The DRNG (or more generally, the deterministic part of an RNG) may gain entropy from the seed and possibly from additional input during the operational work (reseeding or refreshing). The seed and the additional input may be provided by different sources.
- 89 A DRNG may be based on the concept of complexity-theoretic randomness (cf. e.g. [Calu02] for details). The sequences generated by a DRNG then shall be computationally indistinguishable from random sequences generated by computational power.

Hybrid RNG

- 90 A hybrid RNG combines the design principles of true and deterministic RNGs, in particular, it consists of an entropy source and a deterministic part. The entropy source of a hybrid PTRNG should provide at least as much entropy as the output random numbers might at most contain⁶. Loosely speaking, this means that the entropy source must generate at least so much entropy that a perfect post-processing algorithm might generate an ideal output sequence. A hybrid DRNG usually gets (considerably) less entropy from the entropy source by reseeding (or refreshing) than the length of its output measured in bits. Roughly speaking, the security of hybrid PTRNGs relies on both the entropy of the output sequences and the computational complexity, while the security of hybrid DRNGs essentially relies on computational complexity.

2.2. Random Numbers in IT Security

2.2.1. Usage of Random Numbers in IT Security

- 91 Many security mechanisms need secrets, e.g., cryptographic keys or authentication data. ‘Unpredictable’ random numbers are ‘ideal’ secrets for IT security applications. The use of RNGs as a security mechanism results in requirements on the random numbers, or more specifically, on their generation.
- 92 In the terminology of the Common Criteria, RNGs are probabilistic mechanisms. The vulnerability analysis assesses the strength of permutational or probabilistic mechanisms and other mechanisms to ensure that they can withstand direct attacks (cf. [CEM], section B.2.1.3, and chapter 5.7 of this document for details).
- 93 Guessing a secret by (i) selecting an admissible value; and (ii) checking whether it is correct, is typical for direct attacks. To increase the success probability, it may be reasonable to formulate and analyze a stochastic model that considers how the secret has been generated, i.e., the probability distribution of the admissible values, e.g., a set of passwords or a key space. The ability to verify guesses depends on the availability of suitable reference data and on the workload of the checking procedure. A cryptographic key may be guessed independent of the TOE. If the attacker knows the cryptographic algorithm and sufficiently many plain text / cipher text pairs, the key can be searched for by means of massive parallel high-speed computations without any cryptanalysis. Passwords may be found out by trial and error, but the password mechanism may limit the number of authentication attempts in time (e.g., if human user input is assumed) and the total number of guesses (e.g., by requirement of the component FIA_AFL.1, cf. [CCV31_2] for details). From the attacker’s point of view, the situation is clearly much more

⁶ Cf. paragraph 119 on page 28 for details.

comfortable if he knows some reference string that has been calculated from the correct password, which allows automatic search.

- 94 Table 1 describes the link between the maximum success probability of a single guess of a cryptographic key, the number of security bits, and the assumed attack potential according to the CC.

Table 1: Attack potential, guessing probability and security bits

Component of the vulnerability analysis			Success probability of a single guess	Security bits
Common Criteria Version 2.3		Common Criteria Version 3.1		
		AVA_VAN.{1, 2} (basic)	$\varepsilon \leq 10^{-12}$	≥ 40 security bits
AVA_SOF.1, low	AVA_VLA.2 (low)	AVA_VAN.3 (enhanced basic)	$\varepsilon \leq 3 \cdot 10^{-15}$	≥ 48 security bits
AVA_SOF.1, medium	AVA_VLA.3 (moderate)	AVA_VAN.4 (moderate)	$\varepsilon \leq 5 \cdot 10^{-20}$	≥ 64 security bits
AVA_SOF.1, high	AVA_VLA.4 (high)	AVA_VAN.5 (high)	$\varepsilon \leq 8 \cdot 10^{-31}$	≥ 100 security bits

- 95 As a general rule, the guessing probability for passwords must not exceed the upper bounds given in Table 2, which depend on the assumed attack potential that is claimed in the security target. If a probabilistic or permutational mechanism relies on entry of data by a human user (e.g., the choice of a password), the worst case should be considered.
- 96 Table 2 describes the link between maximum guessing probability ε for passwords and the assumed attack potential according to the CC.

Table 2: Attack potential and guessing passwords

Component of the vulnerability analysis			Success probability of a single guess	Success probability with blocking after 3 failed attempts	Recommended
Common Criteria Version 2.3		Common Criteria Version 3.1			
		AVA_VAN.{1,2} (basic)	$\varepsilon \leq 10^{-4}$	$\varepsilon \leq 3 \cdot 10^{-4}$	$\varepsilon \leq 10^{-5}$
AVA_SOF.1, low	AVA_VLA.2 (low)	AVA_VAN.3 (enhanced basic)	$\varepsilon \leq 10^{-4}$	$\varepsilon \leq 3 \cdot 10^{-4}$	$\varepsilon \leq 10^{-6}$

Component of the vulnerability analysis			Success probability of a single guess	Success probability with blocking after 3 failed attempts	Recommended
Common Criteria Version 2.3		Common Criteria Version 3.1			
		AVA_VAN.{1,2} (basic)	$\varepsilon \leq 10^{-4}$	$\varepsilon \leq 3 \cdot 10^{-4}$	$\varepsilon \leq 10^{-5}$
AVA_SOF.1, medium	AVA_VLA.3 (moderate)	AVA_VAN.4 (moderate)	$\varepsilon \leq 10^{-5}$	$\varepsilon \leq 3 \cdot 10^{-5}$	$\varepsilon \leq 10^{-7}$
AVA_SOF.1, high	AVA_VLA.4 (high)	AVA_VAN.5 (high)	$\varepsilon \leq 10^{-6}$	$\varepsilon \leq 3 \cdot 10^{-6}$	$\varepsilon \leq 10^{-8}$

2.2.2. Basic considerations for RNG types

- 97 For a reasonably designed RNG, the generated random numbers should be mutually distinct if the random numbers are sufficiently long.
- 98 **R1:** (statistical unobscuredness) The application of statistical (standard) black box tests or test suites does not distinguish the generated random numbers from realizations of uniformly distributed independent random variables. A more challenging formulation of this requirement says that statistical tests cannot distinguish between random numbers and realizations of ideal sequences. (Of course, ‘unfair’ tests, e.g., referring the actual seed value of a DRNG, have to be excluded anyway.)
- 99 **R2 (backward and forward security):** It must (at least practically) be impossible to determine predecessors or successors of known sub-sequences of output random numbers. The guessing probability shall be at most negligibly greater than without the knowledge of the sub-sequence.
- 100 **R3 (enhanced backward security):** Even if an adversary knows the current internal state of the RNG, the publicly known inputs (if any exist), and the current and future random numbers, she shall (at least practically) not be able to determine preceding random numbers; that is, she shall be able to guess these random numbers only with a negligibly greater probability than without this knowledge.

Note that the (weaker) backward security demands that previous random numbers cannot be determined from the current and future random numbers. The knowledge of current or future output random numbers may be relevant for physical RNG with internal memory (used for the post-processing algorithm). For a pure DRNG, the internal state and all the publicly known inputs determine the current and the future random numbers.

- 101 **R4 (enhanced forward security):** Even if an adversary knows the internal state of the RNG, all the publicly known inputs and a sequence of preceding random numbers, she shall (at least practically) not be able to determine the next random number; that is, she shall not be able to guess this random number with non-negligibly greater probability than without this knowledge.

Note that the (weaker) forward security requires that future random numbers cannot be determined from the current and previous output values. Pure DRNG may fulfil forward secrecy if the internal state cannot be determined from the knowledge of the current and the previous output values (random numbers). Forward secrecy under the additional condition that the current internal state is compromised (enhanced forward security) cannot be achieved by pure DRNGs. Enhanced forward security may be achieved by hybrid DRNGs if the internal state is permanently reseeded (or is updated) with data that was generated by a strong entropy source.

- 102 Requirement R1 is usually verified by a fixed set of statistical black box tests and possibly by some additional statistical tests that are tailored to the concrete RNG. For true RNGs without a history-dependent internal state, Requirement R2 is essentially equivalent to the combination of Requirement R3 and Requirement R4.
- 103 Requirement R4 cannot be fulfilled by pure DRNGs, since the internal state clearly determines all subsequent random numbers. Forward secrecy requires sufficient refreshing or reseeding of the internal state.
- 104 Requirement R3 may be dropped for devices that are assumed to be secure against all kinds of attacks that could discover (parts of) the internal state or for devices that are operated in a secure environment. Requirement R4 may be relevant if it cannot be excluded that an adversary has unnoticed access to the device and is able to discover the internal state of the device.

2.2.3. Design Description of RNG

Overview

- 105 The description of the RNG design in general comprises
- (1) the entropy source of the non-deterministic part,
 - (2) the digitization of the raw random signal provided by the entropy source,
 - (3) any post-processing of the raw random number sequence producing the internal random numbers,
 - (4) the deterministic part of the RNG in terms of the internal state, the state transition function ϕ , and the output function ψ ,
 - (5) the seeding, refreshing (or reseeding) mechanism of the deterministic part of the RNG, and
 - (6) any secrets and publicly known input of the deterministic part of the RNG (inclusively, the generation process and how it is used).

Depending on the RNG design, some of these design elements come from external sources or they may be trivial as discussed below.

PTRNG

- 106 The PTRNG design is in general described by
- (1) the internal entropy source that generates raw random signals,

- (2) the digitization mechanism of the raw random signal into the raw random number sequence,
- (3) any post-processing of the raw random number sequence generating the internal random numbers⁷, secrets and publicly known values (if there are any), and
- (4) the online test(s) (applied to the raw random numbers or the internal random numbers), a tot test (shall detect a total failure of the entropy source), and a start-up test.

107 The post-processing algorithm may comprise a cryptographic one-way function to prevent the analysis of the raw random number sequence on the basis of knowledge of the RNG output. A hybrid PTRNG may contain a DRNG for post-processing.

NPTRNG

108 In general the design of a NPTRNG is described by

- (1) the external entropy sources continuously providing digital raw random signals as input to the NPTRNG,
- (2) any secrets and publicly known input values (including the generation process and how it is used) if used by the NPTRNG,
- (3) the pre-processing of the raw random number sequence and publicly known input,
- (4) the deterministic post-processing of the pre-processed input in terms of the internal state, the state transition function ϕ , and the output function ψ , and
- (5) the self-test, if implemented.

109 Usually the entropy source of a NPTRNG provides low-entropy sequences. If directly used for output these sequences must be compressed. However, in many designs these sequences are used to update the internal state of a DRNG. Usually, the core of the post-processing algorithm is a hash function. For a non-physical true RNG, the average entropy of the raw data must at least equal the output length of the internal random numbers in bits (cf. paragraph 111 for details on hybrid RNG).

DRNG

110 In general the design of a DRNG is described by

- the seeding procedure that generates the first internal state of the DRNG,
- the generation of the output and the next internal state of the DRNG, and
- the control system for DRNG instantiation, de-instantiation, and limitation for the amount of random numbers produced after seeding.

The seeding procedure may distinguish between

⁷ Formally, a missing post-processing may be interpreted as the identity mapping.

- the instantiation of the DRNG generating the initial internal state using an entropy input string, and
- the reseeding / refreshing of the DRNG generating the next internal state from the current internal state and (possibly) an external input string.

111 We describe the deterministic part of an RNG by a 6-tuple $(S, I, R, \varphi, \psi, p_A)$, more precisely:

S set of internal states

I input alphabet

R output alphabet

s_0 initial internal state (derived from the seed)⁸

$\varphi: S \times I \rightarrow S$ (state transition function), $s_{n+1} := \varphi(s_n, i_n)$ (1)

$\psi: S \times I \rightarrow R$ (output function), $r_n := \psi(s_n, i_n)$ (2)

p_A probability distribution of the initial internal state s_0 that is derived from the seed⁹.

112 For the description of multistep-behaviour of the 6-tuple we derive the extended transition function φ^* and extended output function ψ^* over $S \times I^*$, $I^* = \bigcup_{k=1}^{\infty} I^k$, where for $s \in S$ and

$i^* = (i_1, \dots, i_k) \in I^*$ and $R^* = \bigcup_{k=1}^{\infty} R^k$ hold

$\varphi^*: S \times I^* \rightarrow S$, $\varphi^*(s, \bar{i}) = \varphi(\varphi(\dots \varphi(\varphi(s, i_1), i_2) \dots, i_{k-1}), i_k)$ (3)

$\psi^*: S \times I^* \rightarrow R^*$,

$\psi^*(s, i^*) = (\psi(s, i_1), \psi(\psi(s, i_1), i_2), \dots, \psi(\psi(\dots \psi(\psi(s, i_1), i_2) \dots, i_{k-1}), i_k))$ (4)

113 In some cases one may require that φ or ψ^* are one-way functions (in a sense discussed below), i.e. that it is easy to compute the output for a given input but it is computationally infeasible to find for a given output an input, which maps to this output. For φ directly follows that $|S \times I|$ shall be sufficiently large preventing exhaustive search of appropriate (s', i') such that $\varphi(s', i') = s''$ for a given s'' , $s'' \in \bigcup_{s \in S} \bigcup_{i \in I} \varphi(s, i)$. For small R the set R^* will contain short

In many cases, the seed equals the first internal state.

output sequences allowing to guess an appropriate (s', i^*) such that $\psi^*(s', i^*) = r^*$ for a given r^* , $r^* \in \bigcup_{s \in S} \bigcup_{i^* \in I^*} \varphi(s, i^*)$. If we require “the extended output function ψ^* being a one-way function” it requires more precisely the one-way feature for sufficiently long output sequences r^* , i.e. $|r^*| > l$, where l is big enough that $|R|^l$ prevents exhaustive search of (s', i^*) .

- 114 The 6-tuple is a semiformal¹⁰ description of the deterministic part of the RNG. It is not necessarily formal, because it may not necessarily allow formal proofs as demanded by formal description languages¹¹. For DRNGs, ‘secrets’ mentioned above may be viewed as part of the seed or the internal state.
- 115 The 6-tuple may define a MEALY machine, where the initial (starting) internal state is a random value derived from a random variable with distribution p_A (this is an extension of the definition e.g., in [HDCM00]).
- 116 Apart from the seed, a DRNG may get additional input data while it is in operation. Without loss of generality, we may assume that an external entropy source generates data $a_1, a_2, \dots \in A := A_0 \cup \{o\}$, where A_0 denotes a finite set of admissible input values, and the value $a_n = o$ is logically equivalent to “no input from an external entropy source in step n ”. Analogously, we assume that $b_1, b_2, \dots \in B := B_0 \cup \{o\}$ denotes a sequence of publicly known data, where B_0 denotes a finite set of admissible input values, and $b_n = o$ is logically equivalent to “no publicly known external input in Step n ”. Note: the publicly known input does not provide any entropy to the RNG, but may affect the internal state and the output of the RNG. In particular, we have $s_{n+1} := \varphi(s_n, a_n, b_n)$ and $r_n := \psi(s_n, a_n, b_n)$. We define $i_n := (a_n, b_n) \in I := A \times B$ for $n \geq 1$, where i_n is the input to φ and ψ in Step n .
- 117 If $a_n = o$ for all $n \geq 1$, we may simplify the model by “neglecting” the set A , i.e., we may set $I = B$. Analogously, we may set $I = A$ if no publicly known values are fed into the DRNG during its life cycle.
- 118 A pure DRNG runs without any external input after seeding, i.e., $i_n = (o, o)$ for all $n \geq 1$. The state function and the output function of the MEALY machine may be simplified to $(S, R, \varphi', \psi', p_A)$ with

$$\varphi' : S \rightarrow S, s_{n+1} := \varphi'(s_n) \text{ and } \psi' : S \rightarrow R, r_n := \psi'(s_n). \quad (5)$$

¹⁰ “semiformal” means “expressed in a restricted syntax language with defined semantics” (cf. CC part 1 paragraph 81).

¹¹ “formal” means expressed in a restricted syntax language with defined semantics based on well-established mathematical concepts (cf. CC part 1 paragraph 51).

Hybrid RNG

119 Whether a hybrid RNG is categorized as a hybrid DRNG (which means that its security essentially is based on computational complexity) or as a hybrid PTRNG (which means that its primary security anchor is based on entropy) is not always clear. It may be difficult or even not clear in concrete cases. Roughly speaking, the classification essentially depends on the relation between the entropy of the seed-update material (that is, the entropy of the reseeding material), and the maximum entropy the internal random numbers may attain, namely $\lg_2|R|$, which is provided by ideal RNGs. Let us assume the following:

- (1) The sequence of additional inputs a_1, a_2, \dots is stationary (or more precisely, the sequence a_1, a_2, \dots is assumed to be generated by stationary random variables A_1, A_2, \dots) and has Min-entropy $h = H(a_i, \dots, a_{i+k-1})$;
- (2) Within k cycles, the state transition function at most slightly reduces the entropy of the internal state (If the mapping $\varphi: S \times \{i\} \rightarrow S$ is a permutation over S , for any fixed $i \in I$ The entropy of the internal state is not reduced, even if an adversary knows all external input values.); and
- (3) ψ is surjective. If $h \geq k \lg_2|R| - \varepsilon$, for a small constant ε ,

the RNG *may* behave like a **hybrid PTRNG** since the non-deterministic part of the RNG provides at least almost as much entropy as the output sequence may have in the best case. Note, however, that this does not prove that the entropy of the output is indeed close to $k \lg_2|R|$ or equivalently, the internal random numbers are (at least almost) uniformly distributed and independent. This clearly depends on the concrete RNG, i.e., on the state transition function and the output function, and demands a solid proof. In contrast, if $h + H(s_i) \ll k \lg_2|R|$, the non-deterministic RNG part does not provide sufficient entropy to ensure that the output sequence can be truly random: The RNG behaves as a **hybrid DRNG**.

2.3. Mathematical Background**2.3.1. Random variables**

120 An **experiment** is any physically or mentally conceivable undertaking that results in a measurable outcome¹². The **sample space** is the set Ω of possible outcomes of an experiment. Unless otherwise stated, in this document we assume the sample space as finite set. The **sample size** of an experiment is the number of possible outcomes of the experiment (= cardinality of the sample space). An **event** is a subset of Ω . A **probability measure** on a *finite* sample space Ω is a function \Pr from the power set of Ω (= set of subsets of Ω) into the interval $[0,1]$ satisfying

$$P\{\Omega\} = 1 \tag{6}$$

¹² We follow the terminology in [HDCM], chapter 7 Discrete Probability.

$$P\left\{\bigcup_{k=1}^n A_k\right\} = \sum_{k=1}^n P\{A_k\} \text{ if the events } \{A_1, A_2, \dots, A_n\} \text{ are mutually disjoint} \quad (7)$$

121 More generally, a **probability space** is a triple (Ω, S, P) , where

- Ω is a set (*not necessarily finite*),
- S is a σ -algebra over Ω (By definition, $\Omega \in S$ and $\emptyset \in S$, and $\Omega - A \in S$ whenever $A \in S$. Moreover, $\bigcup_{i=1}^{\infty} A_i \in S$ for any countable sequence $A_1, A_2, \dots \in S$.)
- P a probability measure on the a σ -algebra S (By definition, $P: S \rightarrow [0,1]$ is a function that assigns to each $s \in S$ some real number between 0 and 1. In particular, $P(\Omega) = 1$, and for mutually disjoint events $A_1, A_2, \dots \in S$, we have $P(\bigcup_{i=1}^{\infty} A_i) = \sum_{i=1}^{\infty} P(A_i)$.)

122 For countable Ω (finite or infinite), we always assume that S equals the power set.

123 From a mathematical point of view, a **random variable** X is a function $X: \Omega \rightarrow V$ that assigns to each element $\omega \in \Omega$ an observable value $v \in V$. More precisely, some fixed σ -algebra S' is assigned to V , and the mapping $X: \Omega \rightarrow \Omega'$ is measurable with regard to the σ -algebras S and S' . A random variable is denoted **discrete** if V is countable.

124 In the following, we interpret random numbers r_1, r_2, \dots as realizations of random variables R_1, R_2, \dots . Dealing with random variables allows the use of probabilities in place of relative frequencies (i.e., in place of empirical probabilities). For ‘real-world’ RNGs, the distribution of the underlying random variables R_1, R_2, \dots is usually unknown. Depending on the concrete design, one may determine a family of distributions to which the true (but unknown) distribution belongs.

125 In the following, we mainly deal with countable (mostly finite) sets V . We skip measurability problems and refer the interested reader to the relevant literature.

126 By definition, random variables X_1, X_2, \dots are said to be **independent** if for any positive integer k and for any sequence $A_1, \dots, A_k \in S$ the equality $P\{X_1 \in A_1, \dots, X_k \in A_k\} = P\{X_1 \in A_1\} \cdot \dots \cdot P\{X_k \in A_k\}$ holds. For discrete random variables X_1, X_2, \dots , this condition simplifies to $P\{X_1 = x_1, \dots, X_k = x_k\} = P\{X_{1+t} = x_1\} \cdot \dots \cdot P\{X_{k+t} = x_k\}$ for any sequence $x_1, \dots, x_k \in V$.

127 A random variable X that assumes values in a finite set V is said to be **uniformly distributed** (or equivalently: unbiased, equidistributed) if it assumes all $v \in V$ with the same probability. Otherwise X is said to be **biased**. An ideal RNG is characterized by a sequence of independent, uniformly distributed random variables.

128 Random variables X_1, X_2, \dots are called **stationary** if $P\{X_1 \in A_1, \dots, X_k \in A_k\} = P\{X_{1+t} \in A_1, \dots, X_{k+t} \in A_k\}$ for any positive integers k and t and for any sequence $A_1, \dots, A_k \in \mathcal{S}$. For discrete random variables, this condition simplifies to $P\{X_1 = x_1, \dots, X_k = x_k\} = P\{X_{1+t} = x_1, \dots, X_{k+t} = x_k\}$ for all $x_1, \dots, x_k \in V$.

129 **Example 1:** Interpret the outcomes of N subsequent tosses of a coin as a realization of random variables T_1, T_2, \dots, T_N . Since a coin has no memory, we may assume that the random variables T_1, T_2, \dots, T_N are independent and identically distributed. This distribution is known as the **BERNOULLI** distribution and denoted by $B(1, p)$, where without loss of generality, $\Pr\{T_j = \text{Head}\} = p$ and $\Pr\{T_j = \text{Tail}\} = 1 - p$. If the coin is fair then the random variables T_1, T_2, \dots, T_N are unbiased, i.e. for all $1 \leq j \leq N$ holds $p = 0.5$. In the ‘real world’ for a particular coin, its parameter p is unknown, but can be estimated by experiments. Let the outcome of the k -th coin toss be denoted as t_k and $h_n = |\{k | t_k = \text{Head}, k \in \overline{1, n}\}|$. For any $\varepsilon > 0$,

$$P\left\{\left|\frac{h_n}{n} - p\right| < \varepsilon\right\} \rightarrow 1 \text{ as } n \rightarrow \infty \text{ (weak law of large numbers).} \quad (8)$$

This allows a precise estimate of the unknown probability p .

130 The real-valued function $f : R \rightarrow R$ is a **density function** if $f(x) \geq 0$ for all $x \in R$ and $\int_{-\infty}^{\infty} f(x)dx = 1$. For a **continuous** real-valued random variable X (i.e., $X : \Omega \rightarrow R$, where R denotes the set of real numbers), there exists a density function f such that $P\{a < X < b\} = \int_a^b f(x)dx$ for all $a < b$.

131 The **mean** (expected value) $E(X)$ of a discrete real-valued random variable X is given by

$$E(X) = \sum_k k P\{X = k\}. \quad (9)$$

For a continuous random variable X with density function $f(x)$ the mean equals

$$E(X) = \int_{-\infty}^{\infty} xf(x)dx. \quad (8)$$

132 If a random variable X assumes binary vectors, $X : \Omega \rightarrow \{0,1\}^n$, however, no meaningful definition for the mean is evident. By identifying the binary vectors $(x_{n-1}, x_{n-2}, \dots, x_0)$ with

$$b(x_{n-1}, x_{n-2}, \dots, x_0) := \sum_{k=0}^{n-1} x_k 2^{x_k},$$

definition (7) could principally be applied. However, any meaningful interpretation of the “expected value of $b(X)$ ” should take into account that the

coordinates contribute differently to this sum. Generally speaking, statistics should always consider V (cf. [SaHe06], section 1.4, for details).

- 133 The **variance** $Var(X)$ of a real-valued random variable X is defined by $Var(X) := E((E(X) - X)^2)$. Its **standard deviation** is given by $\sigma := \sqrt{Var(X)}$.

- 134 Given a probability space (Ω, S, P) , a **stochastic process** with state space Ω is a collection of real-valued random variables $X(t, \omega)$, where $\omega \in \Omega$ and $t \in T$ (“time”), i.e., in short, $\{X_t : t \in T\}$. If $T \subseteq \Delta\mathbb{Z}$ with $\Delta > 0$, the stochastic process is called (time-)discrete.

- 135 A stochastic process is called **stationary** if

$$P(X_{t_1} \in A_1, X_{t_2} \in A_2, \dots, X_{t_k} \in A_k) = P(X_{t_1+\tau} \in A_1, X_{t_2+\tau} \in A_2, \dots, X_{t_k+\tau} \in A_k) \quad (10)$$

for all k -tuples $(t_1, t_2, \dots, t_k) \in T^k$, $\tau > 0$ and all (measurable) subsets A_1, \dots, A_k of R . If the random variables X_j are discrete (9) simplifies to

$$P(X_{t_1} = r_1, X_{t_2} = r_2, \dots, X_{t_k} = r_k) = P(X_{t_1+\tau} = r_1, X_{t_2+\tau} = r_2, \dots, X_{t_k+\tau} = r_k). \quad (11)$$

with real numbers r_1, \dots, r_k .

- 136 A stochastic process is **stationary in wide-sense** if $E(X_t) = E(X_{t+\tau})$, $E(X_{t_1} X_{t_2}) = E(X_{t_1+\tau} X_{t_2+\tau})$ for all $t, t + \tau \in T$. (12)

- 137 A stationary stochastic process is **ergodic** if statistical properties can be deduced from a single, sufficiently long realization of this stochastic process. More precisely, assume that for the (measurable) function $f(x_{t_1}, x_{t_2}, \dots, x_{t_k})$, the mean over the time exists. Then

$$\bar{f}(x_{t_1}, x_{t_2}, \dots, x_{t_k}) = \lim_{T \rightarrow \infty} \frac{1}{2T} \int_{-T}^T f(x_{t_1+t}, x_{t_2+t}, \dots, x_{t_k+t}) dt \text{ and } P(\bar{f} = E(f)) = 1 \quad (13)$$

where $t_1, t_2, \dots, t_k, t_1 + \tau, t_2 + \tau, \dots, t_k + \tau \in T$.

2.3.2. Entropy and Guess Work

- 138 Let X be a random variable over $\Omega = \{\omega_1, \omega_2, \dots, \omega_n\}$, $p(\omega_1) \geq p(\omega_2) \geq \dots \geq p(\omega_n)$. We want to estimate the effort to guess the outcome of an experiment that is viewed as a realization of X . The optimal strategy is to guess the values $\omega_1, \omega_2, \dots$ until the outcome is found. The **λ -work-factor** $w_\lambda(X)$ denotes the minimum number of guesses to get the result with probability λ , $0 < \lambda < 1$, if the optimal strategy is applied. That is,

$$w_\lambda(X) = \min \left\{ k \mid \sum_{i=1}^k p(x_i) \geq \lambda \right\}. \quad (14)$$

- 139 For $\lambda = 0.5$ the work factor of the optimal strategy meets the following inequality [Plia99]

$$\left\lfloor \frac{1}{2 \max_{i \in 1, n} \{p(x_i)\}} \right\rfloor \leq w_{1/2}(X) \leq \lceil (1 - \|\bar{p} - \bar{u}\|) \cdot n \rceil \quad (15)$$

with $\bar{p} = (p(\omega_{i_1}), p(\omega_{i_2}), \dots, p(\omega_{i_n}))$ and $\bar{u} = (1/n, \dots, 1/n)$ (uniform distribution). In particular,

$$\|\bar{p} - \bar{u}\| = p(A) - \frac{|A|}{n}, \quad A = \left\{ a \mid p(a) \geq \frac{1}{n}, \quad a \in \Omega \right\}. \quad (16)$$

Note that $\|\bar{p} - \bar{u}\|$ is invariant on (i_1, i_2, \dots, i_n) .

140 The guess work is the expected number of guesses if the optimal strategy is applied

$$W(X) = \sum_{i=1}^n i p(x_i) \quad (17)$$

For the most general case, the following inequality provides tight bounds for the guess work

$$\frac{n}{2} \|p - u\| \leq \frac{n-1}{2} - W(X) \leq n \|p - u\|. \quad (18)$$

141 Chapter 5.1 provides an example of calculation guess work for binary vectors consisting of independent but biased bits.

142 The logarithm of the denominator (apart from factor 2) in the left side of the inequation (15) is called Min-entropy

$$H_{\min}(X) = -\log_2 \max_{i \in 1, n} \{p(x_i)\} \quad (19)$$

143 The Min-entropy represents a special case of the more general notion of RENYI entropy H_α , where

$$H_\alpha(X) = \frac{1}{1-\alpha} \log_2 \sum_{i=1}^n (\Pr\{X = \omega_i\})^\alpha, \quad 0 \leq \alpha < \infty \quad (20)$$

$$\lim_{\alpha \rightarrow \infty} H_\alpha(X) = -\log_2(\max\{\Pr(X = \omega_i)\}) = H_{\min}(X) \quad (21)$$

For special cases of α , the RENYI entropy yields the inequations

$$H_{\min} < H_2 \leq H_1, \quad H_{\min} < H_2 < 2H_{\min} \quad (22)$$

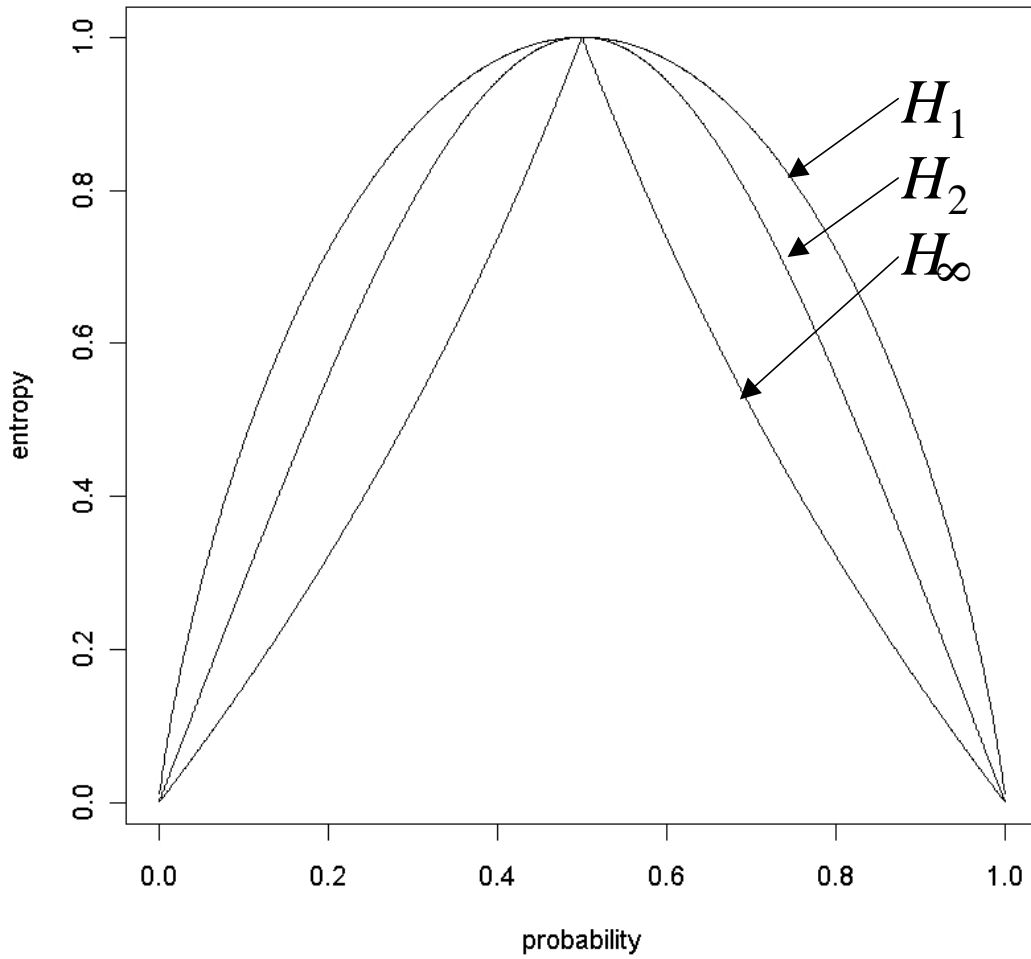


Figure 1: Min-entropy, collision-entropy and Shannon-entropy for binary-valued random variables

- 144 The well-known SHANNON entropy may be interpreted as the special case $\alpha = 1$. In particular, L'Hopital's rule yields

$$H_1(X) = -\sum_{i=1}^n \Pr(X = \omega_i) \log_2(\Pr(X = \omega_i)) \quad (23)$$

- 145 Another important special case of the RENYI entropy is the collision entropy H_2 . Let X and X' be two random independent and identically-distributed variables with values in a finite set Ω . The collision probability is $P\{X = X'\} = \sum_{x \in \Omega} (P\{X = x\})^2$, and the collision entropy equals

$$H_2(X) = -\log_2 \left(\sum_{x \in X} (P\{X = x\})^2 \right). \quad (24)$$

Let p denote the distribution of the random variable X . Then [CFPZ09] states

$$\sum_{x \in X} (P\{X = x\})^2 \geq \frac{1 + 4\|p - u\|^2}{|\Omega|}. \quad (25)$$

- 146 A memoryless binary-valued stationary random source can be described by independent identically $B(1, p)$ -distributed random variables X_1, X_2, \dots, X_n . The guess work for n random bits, or equivalently for the random vector $X = (X_1, \dots, X_n)$, may be estimated by the SHANNON entropy [Maur92]

$$\log_2 w_{1/2}(X) \approx n \cdot H_1(X_1). \quad (26)$$

More generally, for an ergodic stationary binary random source, the relation between the guesswork and the length of a sequence tends asymptotically to the SHANNON entropy [Maur92]

$$\lim_{n \rightarrow \infty} \frac{\log_2 w_\alpha(X)}{n} = H_1(X), \text{ for } 0 < \alpha < 1. \quad (27)$$

We point out that these assumptions cover nearly all physical RNGs.

2.3.3. Random mappings

- 147 Let F be a random variable that assumes values in $V \subseteq \{f': A \rightarrow A\}$, where A is a finite domain. Consider the sequence $t_{n+1} := F(\omega)(t_n)$ with $t_0 = t$ for some $t \in A$ and $n \geq 0$. In terms of the functional graph of $F(\omega)$, this sequence describes a path that ends in a cycle. The functional graphs consist of components, each of which consists of one cycle that is connected with several trees (0 trees is possible). We address some well-known results on statistics of random mappings and permutations.
- 148 Table 3 collects some results on random mappings that are chosen uniformly from $SH(A)$, i.e., the set of all n^n mappings $A \rightarrow A$, $|A| = n$ (cf. [Flod89] for more details).

Table 3: Statistics of random mappings

Characteristic	Expected value as $n \rightarrow \infty$	Definition and comments
Number of components	$\frac{1}{2} \log n$	A component consists of one cycle and several trees connected to this cycle.
Component size	$2n/3$	
Largest component	$\approx 0.7782n$	
Number of cyclic nodes	$\sqrt{\pi n/2}$	$\sqrt{\pi n/2} \approx 1,253314\sqrt{n}$
Cycle length (μ)	$\sqrt{\pi n/8}$	The number of edges in the cycle is called the cycle

Characteristic	Expected value as $n \rightarrow \infty$	Definition and comments
		length of t , denoted $\mu(t)$, $\sqrt{\pi n/8} \approx 0,626657\sqrt{n}$.
Maximum cycle length	$\approx 0,78248\sqrt{n}$	
Tail length (λ)	$\sqrt{\pi n/8}$	The number of edges in the path to the cycle is called the tail length of t , denoted $\lambda(t)$, $\sqrt{\pi n/8} \approx 0,626657\sqrt{n}$.
Maximum tail length	$\approx 1.73746\sqrt{n}$	
Rho length (ρ)	$\sqrt{\pi n/2}$	$\rho(t) = \lambda(t) + \mu(t)$, number of steps until a node on the path repeats, $\sqrt{\pi n/2} \approx 1,253314\sqrt{n}$.
Maximum rho length	$\approx 2.4119\sqrt{n}$	
Tree size	$n/3$	Tree size of a node t means the size of the maximal tree (rooted to the cycle) containing this node t .
Largest tree	$\approx 0.48n$	
Number of terminal nodes	$e^{-1}n$	Number of nodes without predecessor, $e^{-1}n \approx 0,367879n$.
Number of image points	$(1 - e^{-1})n$	$ f(A) $ as number of nodes that have a predecessor, $(1 - e^{-1})n \approx 0,632121n$.
Number of k-th iterate image points	$(1 - r_k)n$, $r_0 = 0$, $r_{k+1} = \exp(-1 + r_k)$	$ f^k(A) $ as number of nodes after application of f^k .
Predecessor size	$\sqrt{\pi n/8}$	The predecessor size of the node t is the size of the tree rooted at node t or equivalent the number of iterated pre-images of t .

149 Table 4 collects some results on random mappings that are chosen uniformly from the set of all $n!$ permutations of A , $|A| = n$, (cf. [Golo64] for more details).

Table 4: Statistics of random permutations

	Expected value as $n \rightarrow \infty$	Distribution as $n \rightarrow \infty$
Number of cycles	$\ln n + C + o(1)$ $C = 0.57721566 \dots$	Number ω_n of cycle of the permutation $P\{\omega_n = k\} = \frac{\exp\left(-\frac{(k - \ln n)^2}{2 \ln n}\right)}{\sqrt{2\pi \ln n}} (1 + o(1))$ normal distribution $(\ln n, \ln n)$.

	Expected value as $n \rightarrow \infty$	Distribution as $n \rightarrow \infty$
Cycle length	$\frac{n}{\ln n + C + o(1)}$	Number ω_{nl} of cycle of length l $P\{\omega_{nl} = k\} = \frac{\exp(-1/l)}{l^k k!}$ POISSON distribution with parameter $\lambda = 1/l$.
Length of the largest cycle	$\approx 0.6243n$	
Expected cycle length of a random element	$\frac{n+1}{2}$	Probability that a random element x lies on a cycle of size k , $k \leq n$, is $P(\omega(x) = k) = \frac{1}{n}$

2.4. Stochastics and Statistical Analysis of Physical RNGs

150 Stochastics and statistical methods are very important tools for the analysis of physical RNGs. In this section we address stochastic models and statistical tests.

2.4.1. Stochastic model

151 A principal task when analyzing physical RNGs is to identify and analyze those characteristics of a system or a process that have (significant) impact on the distribution of random numbers. Other features that are considered to be of subordinate meaning are often neglected. In particular, precise physical models of ‘real-world’ physical RNGs (typically based on electronic circuits) usually cannot easily be formulated and are not analyzed.

152 A stochastic model of a physical RNG shall quantify the distribution of the random numbers. Note, however, that a stochastic model usually is significantly less complicated than a physical model of the RNG. This advantage of stochastic models is linked with loss of information. In fact, unlike a precise physical model, a stochastic model does not provide the distribution of the random numbers as a function of the characteristics of the components of the entropy source. Ideally, a stochastic model provides a class of probability distributions that contains the true but unknown distribution of the internal random numbers (which, at least in a strict sense, depends on the particular device). The stochastic model should at least provide a class of distributions that contains the distribution of the das-random numbers or of ‘auxiliary’ random variables if this allows to establish a lower entropy bound for the internal random numbers. These distribution classes usually depend on one or several parameters that may be estimated on the basis of sampled random numbers.

153 Of course, the stochastic model should be checked against empirical data and, if necessary, be corrected and adjusted. Of course, it is advisable to formulate the model as simple as possible to reach the specified goal.

154 As pointed out above, stochastic models may consider:

- the raw random number sequence,

- ‘auxiliary’ random variables (cf. [KiSc08], for instance), and
 - the internal random numbers.
- 155 Stochastic models of the raw random number sequence or of auxiliary random variables focus on the entropy source and the digitization of its analogue raw random signal in order
- to estimate the entropy of the raw random number sequence, and
 - to assess the factors that may affect the quality of the raw random number sequence.
- 156 Stochastic models of internal random numbers additionally consider the algorithmic post-processing of the digital entropy signal
- to assess the effect of the post-processing, and finally
 - to verify the quality of the RNG output.
- 157 The digitized entropy signal may be used directly as internal random numbers or as input for a non-trivial post-processing algorithm. Of course, the first case may be interpreted as a trivial post-processing algorithm (identity mapping). The goal of non-trivial post-processing algorithms may be to increase the average entropy per bit (which demands data compression), simply to smear simple dependencies of raw random numbers, or to obtain a further security anchor if the post-processing algorithm consists of strong cryptographic primitives.
- 158 The digitized entropy signal (or auxiliary random variables or internal random numbers) may be described by a (time-discrete or time-continuous) stochastic process $(\Omega, S, P(\bar{\pi}))$, where $P(\bar{\pi})$ denotes the distribution of the sample function $X(t, \omega)$, which depends on a set of $\bar{\pi}$, $\bar{\pi} \in \Pi$. The set Π specifies the admissible parameter sets and shall contain the true parameter value $\bar{\pi}$ under operational conditions. As mentioned above, the estimate of the entropy, or at least of lower entropy bounds, grounds on these distributions $P(\bar{\pi})$. The parameter set $\bar{\pi}$ may depend on factors such as the concrete entropy source e , operational conditions of the RNG o , and aging¹³ over the life-time t of the RNG. Formally, the parameter set $\bar{\pi}$ depends on e , o and t , i.e. $\bar{\pi}(e, o, t)$. Usually, these dependencies are difficult to quantify. However, it is essential that the parameter set $\bar{\pi}$ remains in the specified set Π ; otherwise, the stochastic model and consequently, the derived entropy bounds, are no longer valid (at least they are no longer reliable). The online test shall detect if the true parameter $\bar{\pi}$ leaves the ‘agreeable’ part of the parameter set, which may result in insufficient entropy.
- 159 The influence on $\bar{\pi}$ clearly has impact on the environmental protection of the entropy source (e.g., by stabilization of the power supply, filters and sensors) and the assessment of the vulnerability by aimed manipulation attacks on the operational condition of the entropy source.
- 160 The following examples discuss stochastic models for theoretical examples. Practical (more complicated) examples of stochastic models can be found, e.g., in [KiSc08] and [BuLu08].

¹³ Aging is a long-time process and does not prevent assumption of stationarity of the entropy source in shorter time frames.

Example 3: Urn model of ideal RNG

- 161 Example 3 considers an urn model $\mathcal{U}(n)$, with n distinguishable balls (numbered from 1 to n). Drawing a ball provides a random number between 1 and n . The outcome is recorded and the ball is put back into the urn. $\mathcal{U}(n)$ models an ideal RNG that generates independent and uniformly distributed random numbers in the range $\overline{1, n}$.
- 162 One may compare a ‘real-world’ RNG (true or deterministic), that is, its output sequences, with the statistical properties of $\mathcal{U}(n)$. This may give an indicator for similarities and differences with ideal RNGs. The evaluator may count frequencies, consider consecutive output values, etc. to compare the (empirical) distribution of the random numbers with the hypothesis of uniform distribution.
- 163 We note that one may also consider another urn model $\mathcal{U}'(n)$ to describe characteristics of an ideal RNG. The model $\mathcal{U}'(n)$ consists of n numbered urns into which balls are randomly put with regard to uniform distribution. $\mathcal{U}'(n)$ considers the output frequencies for an ideal RNG, but unlike $\mathcal{U}(n)$, it ignores the order of the outcomes (cf. [MeOV97]).
- 164 After k balls have been drawn from an urn with n balls (urn model $\mathcal{U}(n)$), the probability for a collision (i.e., at least one ball has been selected at least twice) is

$$P(n, k) = 1 - \frac{(n-1)(n-2) \cdot \dots \cdot (n-k+1)}{n^{k-1}} \quad (28)$$

For large n and k , STIRLING’S formula yields $n! = \sqrt{2\pi n} n^{n+1/2} \exp\left(-n + \frac{\Theta}{12n}\right)$ with $0 < \Theta < 1$. Consequently,

$$P(n, k) \approx 1 - n^{-k+0.5} k^{-k-0.5} e^{k-n}. \quad (29)$$

If $k = O(\sqrt{n})$ and $n \rightarrow \infty$, then

$$P(n, k) \rightarrow 1 - \exp\left(-\frac{k(k-1)}{2n} + O(n^{-1/2})\right) \approx 1 - \exp\left(-\frac{k^2}{2n}\right) \quad (30)$$

For large n , the expected number n_0 of drawings until the first collision is approximately

$$E(n_0) = \sqrt{\pi n / 2}. \quad (31)$$

- 165 Now consider the case where n balls in an urn are distinguishable, but the balls may be drawn with different probabilities. By p_i , we denote the probability that ball i is drawn, and $\bar{p} = (p_1, p_2, \dots, p_n)$, while i_1, i_2, i_3, \dots denotes the labels of the drawn balls. The conditional probability for a collision in step k under the condition that no collision has occurred so far

clearly equals $\sum_{j=1}^{k-1} p_{i_j}$. For the given sequence i_1, i_2, i_3, \dots , the probability for a collision in most k drawings is

$$G(k, \bar{p}) = 1 - \prod_{l=1}^{k-1} \left(1 - \sum_{j=1}^l p_{i_j} \right). \quad (32)$$

Obviously, $\sum_{j=1}^l p_{i_j} \leq l \cdot \max_{j=1, \dots, n} \{p_j\} \leq l / \left\lfloor 1 / \max_{j=1, \dots, n} \{p_j\} \right\rfloor$, and for $k, l < \left\lfloor 1 / \max_{j=1, \dots, n} \{p_j\} \right\rfloor$ we obtain (cf. formula (18) for the definition of Min-entropy H_{\min})

$$G(k, \bar{p}) \leq P\left(k, \left\lfloor 1 / \max_{j=1, \dots, n} \{p_j\} \right\rfloor\right) = P\left(k, \left\lfloor 2^{H_{\min}(\bar{p})} \right\rfloor\right). \quad (33)$$

Note that $2^{H_{\min}(\bar{p})-1}$ is a lower bound of the $1/2$ -guess-factor $w_{1/2}(\bar{p})$ for distribution \bar{p} .

Example 4: Radioactive decay digitized by a Geiger counter [Neue04]

166 In Example 4 we consider a stochastic model of the PTRNG that consists of a random vector (ξ, ζ, η) . The random variable ξ models the behaviour of the entropy while ζ and η correspond to the das-random numbers and the internal random numbers, respectively.

167 The entropy source is a radioactive source that emits particles that are detected by a Geiger counter. The random variable ξ is described by the random variables $\{T_i \mid i = 1, 2, 3, \dots\}$ that model the intermediate times between consecutive impulses. For a reasonable lifetime of the RNG¹⁴ we may assume that the random variables T_i are independent and identically exponentially distributed with $P\{T_i < t\} = 1 - e^{-\theta t}$, where $1/\theta$ is the expected intermediate time between the impulses. Under this assumption, the number of impulses N within any fixed time interval of length t is POISSON distributed with parameter $\lambda = \theta t$, $P\{N = k\} = \frac{\lambda^k e^{-\lambda}}{k!}$.

168 Under the assumption that the detection mechanism is able to measure the intermediate times exactly, the random variables $\{T_i \mid i = 1, 2, 3, \dots\}$ describe the measured intermediate times. The following Lemma says that the measured intermediate times can be used to generate random numbers

$$R_i = T_{2i} / (T_{2i} + T_{2i+1}) \quad (30)$$

that are uniformly distributed on the unit interval $[0, 1)$.

¹⁴ The mean number of impulses decreases over time, depending on the half-life of the radioactive source; e.g., the half-life of Caesium-137 is 30.23 years.

Lemma 1: Let X and Y be identically exponentially distributed random variables with parameter $\theta > 0$. Then $U := \frac{X}{X + Y}$ is uniformly distributed on $[0,1)$.

169 However, a ‘real-world’ detection mechanism is not able to measure the intermediate times between two impulses of the Geiger counter exactly, but only in multiples of a positive constant (= length of a clock cycle; without loss of generality, we assume that this constant equals 1 within this example). If multiple impulses occur within one clock cycle, they are only counted once. The random variable ζ is given by a POISSON process $\{N_t\}_{t \geq 0}$ which considers the number of measured impulses until time t , i.e., N_t is POISSON distributed with λt .

170 The random variables $\{\hat{S}_i \mid i = 1, 2, 3, \dots\}$ quantify the time (expressed as multiples of the clock length) when the 1st, the 2nd, ... impulse is observed. More precisely, $\hat{S}_0 := 0$ and $\hat{S}_i := \min\{k \mid N_k \geq N_{S_{i-1}} + 1, k \in \mathbb{N}\}$ for $i \geq 1$. The random variables $\{\hat{T}_i \mid i = 1, 2, 3, \dots\}$ describe the random intermediate times between consecutive impulses. The random variables \hat{T}_i are independent and identically distributed. More precisely, $\hat{T}_i - 1$ is geometrically distributed with parameter $1 - e^{-\theta}$. If F_X and F_Y denote the cumulative distribution function of random variables X and Y , the term $\|F_X - F_Y\|_\infty = \sup_{x \in \mathbb{R}} \{F_X(x) - F_Y(x)\}$ quantifies the distance of their distributions (cf. formula (14) above).

Lemma 2: Assume that $X' - 1$ and $Y' - 1$ are independent geometrically distributed random variables with parameter $\theta' = 1 - e^{-\theta}$, while U stands for a uniformly distributed random variable, the unit interval $[0,1)$. Then $U' := \frac{X' - 0.5}{X' + Y' - 1}$ satisfies the inequation

$$\frac{1}{2} \tanh\left(\frac{\theta}{2}\right) \leq \|F_U - F_{U'}\|_\infty \leq 1 - \exp\left(-\frac{\theta}{2}\right). \quad (34).$$

171 Note that the RNG post-processing algorithm may take the property of the detection mechanism into account. Instead of following (30), one may calculate

$$\hat{r}_i = (\hat{t}_{2i} - 0.5) / (\hat{t}_{2i} + \hat{t}_{2i+1} - 1). \quad (35)$$

Lemma 2 allows to estimate the deviation of \hat{R}_i from a uniformly distributed random variable on the unit interval.

Example 5: Random mapping model for a pure DRNG

172 A pure DRNG $(S, R, \varphi', \psi', p_A)$ may be modelled as follows:

- The state transition function $\varphi' : S \rightarrow S$ is uniformly selected from some set Φ (during the design of the DRNG, this selection is fixed for all instantiations of the TOE).

- The output function $\psi': S \rightarrow R$ is uniformly selected from some set Ψ (during the design of the DRNG, this selection is fixed for all instantiations of the TOE).
- The initial state s'_0 is viewed as a realization of a random variable S'_0 that is p_A - distributed on the set S . This random variable describes the instantiation of the DRNG during the operational usage of the TOE. The initial state s'_0 may depend on several non-deterministic data, which may be randomly selected at different time instants and different conditions, e.g., on a cryptographic key that is selected during personalization of the TOE, on a binary seed string selected when the device is powered on, on a binary string that is regularly refreshed, etc.

173 The choice of Φ and Ψ has significant impact on properties of the DRNG. The next paragraphs address some characteristics if Φ or Ψ contain injective¹⁵ (not necessarily surjective¹⁶) mappings or bijective¹⁷ mappings (i.e., permutations). In the following, we briefly speak of “mappings” in the first case and of “permutations” in the second case.

174 Assume that $\Phi = SG(S)$, i.e., that the state transition function ϕ is a permutation that is selected uniformly from all permutations over S . The state transition function saves the entropy of the internal state, since $\phi(S) = S$ for each $\phi \in \Phi$. For $\Phi = SH(S)$, the entropy may be reduced while the DRNG is operated, since $\phi(S) \subseteq S$. In particular, one may expect that the number of admissible internal states has been reduced to $\sqrt{\pi n}/8$ (cf. expected number of cycle nodes) after $\sqrt{\pi n}/2$ steps (cf. Table 3; expected rho length). If the initial internal state has entropy $\lg n$, then the cycle states will have entropy of about $(\lg n)/2$.

175 If ϕ and ψ are permutations, the period of the output values (internal random numbers) equals the length of the cycle of $\phi \circ \psi$ that contains the initial state. If both ϕ and ψ consist of only one cycle, the period of the output values is $|S|$, regardless of p_A . Note that if this DRNG is reseeded (with $p_A =$ uniform distribution on S) before each output of a random number for large $|S|$ and $n = O(\sqrt{|S|})$, the probability for a collision is $\exp(-n^2/2|S|)$ (cf. birthday paradox in [MeOV97]).

2.4.2. Overview of Statistical Tests

176 Statistical testing links a stochastic model to the real-world object of investigation. Without verification by statistical tests, the stochastic model provides only assumptions on the behaviour of a real-world IT system, but no evidence (unless the design is very simple (as repeated coin tossing) and has been understood very well from a theoretical point of view). In particular, this is the case for the analysis of a (physical) RNG. Statistics with empirical data are necessary to develop, confirm, and adjust a stochastic model, and finally to understand the source of randomness that is exploited by the RNG and the stochastic properties of the random numbers.

¹⁵ $f: A \rightarrow B$ is injective if $|f^{-1}(b)| = 1$ for all $b \in f(A)$.

¹⁶ $f: A \rightarrow B$ is surjective if for each $b \in B$, there exists an $a \in A$ with $b = f(a)$.

¹⁷ A function is bijective if it is injective and surjective.

- 177 Statistical analysis distinguishes between explanatory variables and response variables of an experiment. **Response variables** quantify the outcome of an experiment. For RNGs, response variables may be the random entropy signal, the digitized entropy signal, or internal random numbers. **Explanatory variables** describe the conditions under which the experiment is conducted. For a PTRNG, an explanatory variable may be the temperature, which may affect the entropy source and finally the random numbers.
- 178 An interesting question is to what extent the response variables depend on a variation of the explanatory variables. Such dependencies may be very different (and difficult to quantify). The average voltage of the power supply may reduce the voltage entropy source and cause a bias in the RNG output. A high resolution measurement of power consumption might provide information on single bits of the output data. The analysis of the TOE should consider the question of whether variations of some explanatory variables may cause significant changes of relevant statistical properties of the generated output data, which in turn might be exploited by an attacker. Generally speaking, the permitted ranges for the explanatory variables must be specified such that the quality of the random numbers is guaranteed for all these parameters.
- 179 The data types of the explanatory variables and the response variables may have different types and scales. A variable is called **categorical** if we can only determine whether two values A and B are equal ($A = B$) or different ($A \neq B$). A typical example of a categorical variable is output strings of RNGs. The appropriate interpretation as integers or real numbers depends on their usage, which is normally not linked to the generation process. A variable is called **continuous** if it can take on real numbers (within the limited resolution of measurement device). This allows several operations of continuous variables:
- **Ordinal scale** allows to distinguish data, namely whether $A = B$, $A \neq B$ or $A < B < C$.
 - **Interval scale** allows to distinguish data, namely whether $A = B$, $A \neq B$ or $A < B < C$. Further, the difference $d := A - B$ can be compared; examples of interval scaled data are temperature in °Celsius or date.
 - **Relation scale** allows to distinguish data, namely whether $A = B$, $A \neq B$ and $A < B < C$. Further, the difference $d := A - B$ and quotient $q = A/B$ can be compared; examples of interval scaled data are temperature in °Kelvin, power in Ampere, period in seconds.
- 180 In this section we describe some statistical standard tests for sequences of binary-valued random numbers that have been generated by an RNG. The response variable ‘binary-valued’ random number is categorical, which limits the variety of appropriate statistical methods to counts of bit strings. These statistical tests do not explicitly consider any explanatory variables of the RNG, although the response values clearly may implicitly depend on the explanatory variables (power supply of PTRNG, time since power-up etc.) Other response variables, such as the analog entropy signal, are continuous and require other statistical tests.
- 181 Unless otherwise stated the null hypothesis for the statistical tests below is that the tested data were generated by an ideal RNG. However, since real-world RNGs are never ideal, it is also reasonable to take distributions into account that are not too ‘far’ from independent and uniformly distributed random numbers.

- 182 The statistician may commit two types of errors. We explain both types of errors by an example. The null hypothesis H_0 supposes that an unknown parameter ϑ is contained in some set Θ , $H_0 : \vartheta \in \Theta$, while the alternative hypothesis is given by $H_A : \vartheta \notin \Theta$. Further, let t denote the statistical test value, K_α , the critical set (yielding the rejection of the null hypothesis), and $\alpha = \sup_{\vartheta \in \Theta} P(t \in K_\alpha | \vartheta)$.

Table 5: Brief overview of error types of statistical tests

	Reality	
Null hypothesis	Null hypothesis is true	Null hypothesis is false
Test rejects the null hypothesis H_0	Error Type 1 (with probability α) $P\{t \in K_\alpha H_0\} \leq \alpha$	correct decision (with “power” = probability $1 - \beta$) $P\{t \in K_\alpha H_A\} \geq 1 - \beta$
Test does not reject the null hypothesis H_0	correct decision (with probability $1 - \alpha$) $P\{t \notin K_\alpha H_0\} \geq 1 - \alpha$	Error Type 2 (with probability β) $P\{t \notin K_\alpha H_A\} \leq \beta$

- 183 Roughly speaking, the null hypothesis is rejected if the test data indicate that the null hypothesis is sufficiently unlikely. One may select between two approaches to define the probability of error type 1.

Predefined level significance: The significance level α is selected before the experiments are conducted. By definition, α is the maximum probability among all hypotheses in H_0 that the null hypothesis is rejected (although it is true). The test suite below follows this approach. We note that for many statistical (non-cryptographic) applications $\alpha = 0,05$ and $\alpha = 0,01$ are typical significance levels. For statistical standard tests, the critical regions K_α are tabulated for common values α .

P-value approach: If the p-value is smaller than a pre-defined bound, the statistician rejects the null hypothesis or he continues testing. The NIST test suite [STS] follows this approach.

- 184 Note that a true statistical hypothesis may be falsified: If the statistical test fails, the Null hypothesis is rejected. If the test value is not very unlikely, this does not confirm the Null hypothesis. **Statistical tests cannot confirm the Null hypothesis. But the absence of evidence is not evidence of absence.** The statistician decides whether he continues or stops testing on the basis of the number of conducted tests. The criteria that cause the end of testing depend on the necessary (claimed) assurance that no deviation of the RNG from ideal RNG can be found or used in practical attacks.

- 185 The statistical tests below assume that the RNG output values $r_i, i \in N$, are binary f bit vectors, $r_i = (b_{(r-1)f+1}, b_{(r-1)f+2}, \dots, b_{rf}) \in \{0,1\}^f$. The tests T0 - T8 are used in two statistical tests suites. We specify the input values, and explain the computation of the test value and the evaluation rules. For some distributions, the error type 1 probabilities are given. The statistical tests T0, T6, T7 and T8 are applied to binary sequences that are formed from consecutive random numbers.

2.4.3. Standard Statistical Tests

- 186 This section describes a standard statistical test suite available on the BSI website https://www.bsi.bund.de/ContentBSI/Themen/ZertifizierungundAnerkennung/ZertifizierungnachCCundITSEC/AnwendungshinweiseundInterpretationen/AISCC/ais_cc.html. Tests T1 to T4 are specified in [FI140-1] (power-on tests).
- 187 We assume that the input values of the statistical tests are realizations of random variables, which are denoted by identical but capital letters. Apart from Test T7, the null hypothesis says that the test data were generated from an ideal RNG. Note, however, that in our context, the rejection probabilities for other distributions also are very interesting, since real-world RNGs are not ideal.

Disjointness Test

- 188 The disjointness tests tests the coincidence of non-overlapping patterns in a sequence seen as an urn model.
- 189 **Test T0** (disjointness test)

Input: $w_1, w_2, \dots, w_{2^{16}} \in \{0,1\}^{48}$

Description and Evaluation rule:

The input sequence $w_1, w_2, \dots, w_{2^{16}}$ passes the disjointness test if and only if its elements are mutually disjoint.

Rejection probability for an ideal RNG: 2^{-17} .

Monobit Tests

- 190 The monobit test tests the bias of a $\{0,1\}$ -valued sequence. Let n_0 denote the number of zeros and n_1 the number of ones in a sequence of length n . Then $\delta := (n_0 - n_1)/2n$ gives the (empirical) bias of this sequence. If a sequence is a realization of independent random variables
- the Min-entropy of the sequence of random variables may be estimated by $H_{\min}(n, \delta) = -n \log_2(0.5 + \delta)$, and
 - its SHANNON entropy may be estimated by $H_1 \approx n \left(1 - \frac{2\delta^2}{\log_e 2} \right)$ for small bias.

191 General Monobit Test [MeOV97]

If we denote n_0 the number of zeros and n_1 the number of ones in a sequence of n bits, $n > 10$, then a potential attacker might try to combine information gained from many signatures. $T'_1 = \frac{(n_0 - n_1)^2}{n}$ is approximately χ^2 -distributed with 1 degree of freedom. Table 6 contains the critical regions for typical significance levels. More precisely, the table specifies boundary values χ^2_α , such that $P\{\hat{\chi}^2 \geq \chi^2_\alpha\} \leq \alpha$ for any random variable $\hat{\chi}^2$ that is χ^2 -distributed with 1 degree of freedom¹⁸. Alternatively, one may apply the central limit theorem. More precisely, $T''_1 = \frac{n_1 - 0.5n}{0.5\sqrt{n}}$ is $N(0,1)$ -distributed.

Table 6: Typical values of χ^2 -distribution with 1 degree of freedom

Error probability α	0.05	0.01	0.001	0.0001	0.00001	0.000001
χ^2_α -value for the tail probability p , $d = 1$	3.841459	6.634897	10.827566	15.136705	19.511421	23.928127

192 Test T1 (monobit test [FI140-1])

Input:

Bit sequence $b_1, b_2, \dots, b_{20000} \in \{0,1\}$

Test value:

$$T_1 = \sum_{j=1}^{20000} b_j \quad (36)$$

Evaluation rule:

The bit sequence $b_1, b_2, \dots, b_{20000}$ passes the monobit test if and only if $9654 < T_1 < 10346$.

Rejection probability for an ideal RNG: $\approx 10^{-6}$ (Central Limit Theorem: $9.6 \cdot 10^{-7}$)

Remark: The lower and the upper bound of T_1 relates to the frequency f_0 of zeros (or ones) $f_0 \in (0.4827, 0.5173)$. If the relative frequency equals the (true) probability within these borders the Min-entropy is at least 0.9509269 and the SHANNON entropy is at least 0.9991363.

¹⁸ The quantiles of the χ^2 -distribution may be calculated, e.g., using function `qchisq(p, df)` of the tool R available on the website www.r-project.org.

Block Tests

193 General Block Test [MeOV97]:

Divide a binary sequence of length n in non-overlapping blocks of length m and assume that $\left\lfloor \frac{n}{m} \right\rfloor \geq 5 \cdot 2^m$. Identify the admissible m -bit blocks $(c_{m-1}, c_{m-2}, \dots, c_0)$ with the binary representations $i = \sum_{j=0}^{m-1} c_j 2^j$ and define n_i as number of blocks with value i . Under the null hypothesis, the test statistic

$$T'_2 = \frac{2^m}{k} \left(\sum_{i=0}^{2^m-1} n_i^2 \right) - k \quad \text{with } k := \left\lfloor \frac{n}{m} \right\rfloor \quad (37)$$

is approximately χ^2 -distributed with $d = 2^m - 1$ degrees of freedom. Table 7 contains the critical regions for typical significance levels. More precisely, the table specifies boundary values χ_α^2 such that $P\{\hat{\chi}^2 \geq \chi_\alpha^2\} \leq \alpha$ for any random variable $\hat{\chi}^2$ that is χ^2 -distributed with d degrees of freedom.

Table 7: Typical values of χ^2 -distribution with degree of freedom d

Error probability α	0.05	0.01	0.001	0.0001	0.00001	0.000001
$m = 1, d = 1$	3.8415	6.6349	10.8276	15.1367	19.5114	23.9281
$m = 2, d = 3$	7.8147	11.3449	16.2662	21.1075	25.9018	30.6648
$m = 3, d = 7$	14.0671	18.4753	24.3219	29.8775	35.2585	40.5218
$m = 4, d = 15$	24.9958	30.5779	37.6973	44.2632	50.4930	56.4934
$m = 8, d = 255$	293.2478	310.4574	330.5197	347.6542	362.9888	377.0781

194 **Test T2** (poker test [FI140-1])

Input:

Bit sequence $b_1, b_2, \dots, b_{20000} \in \{0,1\}$

Description:

$c_j = 8b_{4j-3} + 4b_{4j-2} + 2b_{4j-1} + b_{4j}$ and $f[i] = \left| \left\{ j \mid c_j = i \right\} \right|$ for $j = 1, \dots, 5000$.

Test value:

$$T_2 = \frac{16}{5000} \sum_{i=0}^{5000} f[i]^2 - 5000 \quad (38)$$

Evaluation rule:

The sequence $b_1, b_2, \dots, b_{20000}$ passes the poker test if $1.03 < T_2 < 57.4$.

Note: For an ideal RNG, the test value is χ^2 -distributed with 15 degrees of freedom.

Rejection probability for an ideal RNG: $1.014 \cdot 10^{-6}$ (χ^2 -approximation)

Runs Tests

195 General Runs Test [MeOV97]:

The expected number $r(n, i)$ of 0-runs (or 1-runs) of length i in an independent unbiased binary sequence of n bits is

$$r(n, i) = (n - i + 3) / 2^{i+2}. \quad (39)$$

Let $\hat{r}(n, i, \lambda)$ the observed number of λ -runs, $\lambda \in \{0, 1\}$, and k be equal to the largest integer i for which $r(n, i) \geq 5$. The statistic used is

$$T'_3 = \sum_{i=1}^k \frac{(\hat{r}(n, i, 0) - r(n, i))^2}{r(n, i)} + \sum_{i=1}^k \frac{(\hat{r}(n, i, 1) - r(n, i))^2}{r(n, i)}, \quad (40)$$

which approximately follows a χ^2 -distribution with $2k - 1$ degrees of freedom. Table 8 contains the critical regions for typical significance levels. More precisely, the table specifies boundary values χ_α^2 such that $P\{\hat{\chi}^2 \geq \chi_\alpha^2\} \leq \alpha$ for any random variable $\hat{\chi}^2$ that is χ^2 -distributed with d degree of freedom. Additionally, Table 8 provides the link between the run length k and the degrees of freedom d .

Table 8: Typical values of χ^2 -distribution for runs

Error probability α	0.05	0.01	0.001	0.0001	0.00001	0.000001
$k = 1, d = 1$	3.84146	6.63490	10.82757	15.13670	19.51142	23.92813
$k = 2, d = 3$	7.81473	11.34487	16.26624	21.10751	25.90175	30.66485
$k = 3, d = 5$	11.07050	15.08627	20.51501	25.74483	30.85619	35.88819

Error probability α	0.05	0.01	0.001	0.0001	0.00001	0.000001
$k = 4, d = 7$	14.06714	18.47531	24.32189	29.87750	35.25854	40.52183
$k = 5, d = 9$	16.91898	21.66599	27.87716	33.71995	39.34065	44.81094
$k = 6, d = 11$	19.67514	24.72497	31.26413	37.36699	43.20596	48.86564

196 Test T3 (runs test [FI140-1])

Input:

Bit sequence $b_1, b_2, \dots, b_{20000} \in \{0,1\}$

Test values:

$$T_3(\lambda, p) = \left| \left\{ i \in \overline{1, 20000} \mid (b_{i-1}, b_i, \dots, b_{i+\lambda}, b_{i+\lambda+1}), b_{i-1} = b_i \oplus 1, b_i = \dots = b_{i+\lambda} = p, b_{i+\lambda+1} = b_{i+\lambda} \oplus 1 \right\} \right|$$

where, $p \in \{0,1\}$, $b_0 := b_1 \oplus 1$, $\lambda = 1, 2, \dots, 20000$, $b_{20001} := b_{20000} \oplus 1$, i.e., the test values are numbers of 0- and 1-runs. A run is a maximum sub-sequence of consecutive zeroes or ones.

Evaluation rule:

The input sequence $b_1, b_2, \dots, b_{20000}$ passes the run test if and only if

$$\forall p \in \{0,1\} \forall \lambda \in \overline{1,5}: m(\lambda) \leq T_3(\lambda, p) \leq M(\lambda) \text{ and} \quad (41)$$

$$m(6) \leq \sum_{\lambda=6}^{20000} T_3(\lambda, 0), \sum_{\lambda=6}^{20000} T_3(\lambda, 1) \leq M(6) \quad (42)$$

i.e., the number of the occurring runs of zeroes and ones with lengths 1, ..., 5 and the sum of all runs of zeroes and ones with lengths greater than 5 lie within the permitted intervals, as specified below.

Run length λ	Lower bound of the interval $m(\lambda)$	Upper bound of the interval $M(\lambda)$
1	2267	2733
2	1079	1421
3	502	748

Run length λ	Lower bound of the interval $m(\lambda)$	Upper bound of the interval $M(\lambda)$
4	223	402
5	90	223
≥ 6	90	223

Rejection probability for an ideal RNG: 10^{-6}

197 **Test T4** (long run test [FI140-1])

Input:

Bit sequence $b_1, b_2, \dots, b_{20000} \in \{0,1\}$

Description:

A run of length ≥ 34 is called a long run.

Evaluation rule:

The input sequence $b_1, b_2, \dots, b_{20000}$ passes the long run test if and only if

$$\forall p \in \{0,1\} \forall \lambda \in \overline{34, 20000}: T_3(\lambda, p) = 0 \quad (43)$$

i.e., if no long run occurs.

Rejection probability for an ideal RNG: 10^{-6}

Autocorrelation Test

198 Autocorrelation Test [MeOV97]:

Let $s = (s_0, s_1, \dots, s_{n-1}) \in \{0,1\}^n$ be a binary sequence, d a fixed integer, $1 \leq d \leq \lfloor n/2 \rfloor$,

$S_5(n, d) = \sum_{i=0}^{n-d-1} (s_i \oplus s_{i+d})$, where \oplus denotes the XOR-sum operator and Σ a sum of real numbers. We consider the test statistic

$$\hat{T}_5(n, d) = 2 \left(S_5(n, d) - \frac{n-d}{2} \right) / \sqrt{n-d}, \quad (44)$$

which approximately follows an $N(0,1)$ the Normal (Gaussian) distribution with mean 0 and variance 1 if $n-d \geq 10$. A two-sided test should be applied. Table 9 provides typical values for a normal (Gaussian) distribution with mean 0 and variance 1 for two-sided tests. More

precisely, $\mathcal{Q}_{N(0,1)}(\alpha/2)$ denotes the $\alpha/2$ -Quantiles

$$P\left\{\hat{T}_5 \leq -\mathcal{Q}_{N(0,1)}(\alpha/2) \vee \hat{T}_5 \geq \mathcal{Q}_{N(0,1)}(\alpha/2)\right\} \leq \alpha. \quad (45)$$

Table 9: Typical values of Normal (Gaussian) $N(0,1)$ for a two-sided test of autocorrelation

Error probability α	0.05	0.01	0.001	0.0001	0.00001	0.000001
$\mathcal{Q}_{N(0,1)}(\alpha/2)$	1.959964	2.575829	3.290527	3.890592	4.417173	4.891638

199 **Test T5** (autocorrelation test)

Input:

Bit sequence $b_1, b_2, \dots, b_{10000} \in \{0,1\}$

parameter $\tau \in \{1, 2, \dots, 5000\}$

Test value:

$$T_5(\tau) = \sum_{j=1}^{5000} (b_j \oplus b_{j+\tau}) \quad (46)$$

Evaluation rule:

The input sequence $b_1, b_2, \dots, b_{10000}$ passes the autocorrelation test (with shift τ) if and only if $2326 < T_5(\tau) < 2674$.

Rejection probability for an ideal RNG and for each shift parameter $\tau \in \{1, 2, \dots, 5000\}$: 10^{-6}

Multinomial Distributions Tests

200 Multinomial Distribution Tests constitute special cases of contingency tables, which will be discussed in Section 2.4.5.2.

201 **Test T6** (uniform distribution test)

Input:

- parameter k (length of the vectors to be tested)
- parameter n (length of the sequence to be tested)
- parameter a (positive real number)
- sequence of k -bit words (w_1, w_2, \dots, w_n) , $w_i \in \{0,1\}^k$, $i = 1, 2, \dots, n$

Test value:

$$T_6(x) := \frac{|\{j : w_j = x\}|}{n} \text{ for all } x \in \{0,1\}^k \quad (47)$$

Evaluation rule:

The sequence w_1, w_2, \dots, w_n passes uniform distribution test with parameters (k, n, a) if

$$T_6(x) \in [2^{-k} - a, 2^{-k} + a] \text{ for all } x \in \{0,1\}^k \quad (48)$$

Special case $k = 1$:

The sequence w_1, w_2, \dots, w_n passes the uniform distribution test if

$$T_6(0) \in [0.5 - a, 0.5 + a] \quad (49)$$

202 **Test T7** (comparative test for multinomial distributions aka ‘test for homogeneity’)

Assumption:

We assume that the random variables $W_{1j}, W_{2j}, \dots, W_{hj}$ describe h independent repetitions of the j^{th} random experiment, and that these random variables are independent and identically distributed for each fixed $j \in \{1, \dots, n\}$. The distribution of $W_{1j}, W_{2j}, \dots, W_{hj}$ is given by $\bar{p}_j = (p_{0j}, p_{1j}, \dots, p_{s-1,j})$, i.e. $p_{ij} = \Pr\{W_{ij} = t\}$ for $i \in \{1, \dots, n\}, t \in \{0, 1, 2, \dots, s-1\}$.

Significance level: α .

Input:

Vectors $(w_{11}, w_{21}, \dots, w_{h1}), (w_{12}, w_{22}, \dots, w_{h2}), \dots, (w_{1n}, w_{2n}, \dots, w_{hn}) \in \{0, \dots, s-1\}^h$ where $(w_{1j}, w_{2j}, \dots, w_{hj})$ denote the outcome of h independent repetitions of the j^{th} random experiment, $j = 1, 2, \dots, n$.

Null hypothesis:

$$\bar{p}_i = \bar{p}_j \text{ for all } i, j \in \{1, 2, \dots, h\}$$

Test value:

The value $f_i[t] = \left| \left\{ j \in \overline{1, n} \mid w_{ij} = t \right\} \right|$, $i = 1, 2, \dots, h$, $t \in \{0, 1, 2, \dots, s-1\}$ counts the number of occurrences of t in $w_{i1}, w_{i2}, \dots, w_{in}$, and denotes

$$p_t = \frac{f_1[t] + f_2[t] + \dots + f_h[t]}{h \cdot n}, \quad t \in \{0, 1, 2, \dots, s-1\}, \quad (50)$$

the relative frequency for t within all samples. The test value $T_7(h, s)$ is defined as

$$T_7(h, s) := \sum_{i=1, \dots, h} \sum_{t=0, \dots, s-1} (f_i[t] - np_t)^2 / np_t \quad (51)$$

Evaluation rule:

The test fails if $T_7(h, s) > \chi^2(\alpha, (h-1)(s-1))$, where $\chi^2(\alpha, (h-1)(s-1))$ is the rejection bound for a χ^2 -test with $(h-1)(s-1)$ degrees of freedom at significance level α .

Entropy Estimation

- 203 Under specific assumptions defined below (i.e., at least for independent identically distributed binary-valued random variables), CORON's test [CoNa98] is more precise than Maurer's "Universal Statistical" Test [Mau].
- 204 The Approximate Entropy Test [STS] [Ruk2000b] provides another entropy test for independent and identically-distributed variables. The Approximate Entropy Test compares the frequency of overlapping blocks of lengths m and $m+1$ against the expected result for a random sequence.
- 205 **Test T8** (entropy estimation) [CoNa98]:

Assumption:

Let the random variables B_1, B_2, \dots model a stationary binary-valued ergodic random source with a finite memory $M \leq L$.

Input:

- (a) word length L
- (b) Q and K test parameter
- (c) Bit sequence b_1, b_2, \dots, b_n , $n = (Q + K)L$

Test value:

For $i = 1, 2, \dots, Q + K$ we consider non-overlapping words $w_i = (b_{(i-1)L+1}, b_{(i-1)L+2}, \dots, b_{iL})$ of length L . Further, A_i denotes the distance from w_i to the nearest predecessor that assumes the same value,

$$A_n = \begin{cases} n & \text{if no } i < n \text{ exist with } w_n = w_{n-i} \\ \min\{i \mid i \geq 1, w_n = w_{n-i}\} & \text{in all other cases} \end{cases} \quad (52)$$

The test value f_C is defined as

$$f_c = \frac{1}{K} \sum_{n=Q+1}^{Q+K} g(A_n) \text{ with } g(i) = \frac{1}{\log(2)} \sum_{k=1}^{i-1} \frac{1}{k} \quad (53)$$

Entropy estimation:

Under the assumptions from above (stationary binary-valued random source with finite memory), the expected value $E(f_c)$ of test variable f_c is closely related to the entropy increase per L-bit block. If the random variables B_1, B_2, \dots are independent, then $E(f_c) = H_1(W) = LH_1(B_1)$.

Evaluation rule:

Note: If the random variables B_1, B_2, \dots are independent and unbiased, a normal distribution with mean $\mu_c = L$ and variance σ_c^2 provides a good distribution of test variable f_c . More precisely, $\sigma_c = c_c(L, K) \sqrt{\text{Var}(g(A_n)) / K}$, $c_c(L, K) = d(L) + \frac{e(L) \cdot 2^L}{K}$ (54)

Table 10: Parameters for entropy test

L	Variance $\text{Var}(g(A_n))$	$d(L)$	$e(L)$
3	2.5769918	0.3313257	0.4381809
4	2.9191004	0.3516506	0.4050170
5	3.1291382	0.3660832	0.3856668
6	3.2547450	0.3758725	0.3743782
7	3.3282150	0.3822459	0.3678269
8	3.3704039	0.3862500	0.3640569
9	3.3942629	0.3886906	0.3619091
10	3.4075860	0.3901408	0.3606982
11	3.4149476	0.3909846	0.3600222
12	3.4189794	0.3914671	0.3596484
13	3.4211711	0.3917390	0.3594433
14	3.4223549	0.3918905	0.3593316
15	3.4229908	0.3919740	0.3592712

L	Variance $Var(g(A_n))$	$d(L)$	$e(L)$
16	3.4233308	0.3920198	0.3592384
infinite	3.4237147	0.3920729	0.3592016

For $n \geq 23$, the following sum approximates $g(n)$ with an error $< 10^{-8}$:

$$\sum_{j=1}^n j^{-1} = \log n + \gamma + \frac{1}{2n} + \frac{1}{12n^2} + O\left(\frac{1}{n^4}\right), \quad \gamma \approx 0.577216 \text{ (Euler constant)} \quad (55)$$

2.4.4. Test procedures

206 In this section we describe two complex test procedures that are part of the evaluation process. The statistical tests T0 to T8 with the evaluation rules specified in the previous subsection are the basic components of these test procedures.

207 The input data of test procedure A are sequences f of bit random numbers. In a first step, these random numbers are interpreted as binary sequences.

2.4.4.1. Test procedure A

208 **Input:** Step 1: at least $3145728 = 2^{16} \cdot 48$ bits ; Step 2-7: at least $5140000 = 20000 \cdot 257$ bits.

209 Description of test procedure A:

Step 1: Let c_1 denote the smallest integer for which $c_1 \cdot f \geq 48$ and $\pi_{1..48}$ the projection onto the leftmost 48 bits. Apply Test T0 to the sequence $w_1 := \pi_{1..48}(r_1, \dots, r_{c_1}), w_2 := \pi_{1..48}(r_{c_1+1}, \dots, r_{2c_1}), \dots, w_{2^{16}} := \pi_{1..48}(r_{(2^{16}-1)c_1+1}, \dots, r_{2^{16}c_1})$, and apply the evaluation rules specified in the previous subsection.

Step 2: Let c_2 denote the smallest integer with $c_2 \cdot f \geq 20000$ and π_w the projection onto the w^{th} component¹⁹. Generate a sequence of $c_2(f\text{-bit})$ random numbers, interpret the concatenation of these random numbers as a binary sequence, and store the projection onto the leftmost 20000 positions, $s^1 = (b_1^1, b_2^1, \dots, b_{20000}^1)$. Generate a sequence of 20000 $f\text{-bit}$ random numbers, build $s^2 = (b_1^2, b_2^2, \dots, b_{20000}^2), \dots, s^{f+1} = (b_1^{f+1}, b_2^{f+1}, \dots, b_{20000}^{f+1})$, where s^{i+1} consists of the i^{th} bits of the 20000 random numbers. Continue this way (generating sequences of c_2 and of 20000 random numbers, respectively) until s^{257} sets have been built.

¹⁹ In Step 5, the basic tests T1 – T5 are applied to random numbers (which are interpreted as bit strings) and to the traces of the random numbers.

Step 3 – Step 6: Apply the tests T1-T4 to bit sequence $s_i = (b_1^i, b_2^i, \dots, b_{20000}^i)$ for $i = 1, \dots, 257$.

Step 7: Apply Test T5 to bit sequence $s_i = (b_1^i, b_2^i, \dots, b_{20000}^i)$ for $i = 1, \dots, 257$. For each sequence, do the following: Calculate the test values Z_1, \dots, Z_{5000} (see test T5) for $b_1, b_2, \dots, b_{10000}$; determine $\max_{\tau \leq 5000} \{|Z_\tau - 2500|\}$; select τ_0 (randomly in case of several candidates) for which this maximum is assumed; and apply T5 to the sub-sequence $b'_1 := b_{10001}, \dots, b'_{10000} := b_{20000}$ with shift parameter τ_0 .

210 Evaluation rules for Test Procedure A:

- (a) If all $1285 (= 1 + 257 \cdot 5)$ basic tests have been passed, test procedure A is passed. For an ideal RNG this happens with probability ≈ 0.9987 .
- (b) If more than one basic test has failed, test procedure A is failed. For an ideal RNG this happens with probability ≈ 0 .
- (c) If exactly one basic test has failed, the statistician (evaluator, etc.) applies test procedure A once more to new internal random numbers and applies evaluation rules a) and b). If within a second run of test procedure A one basic test fails, test procedure A is failed. A second repetition of test procedure A is not allowed.

Justification: Test procedure A is applied to output sequences of DRNGs, PTRNGs and NPTRNGs. The goal of test procedure A is to check whether the random numbers (for PTRNGs: internal random numbers) behave statistically inconspicuously. The projection onto the particular components of the internal random numbers shall detect when particular positions of the internal random numbers behave differently than others. For an ideal RNG the probability that test procedure A finally fails is almost 0.

2.4.4.2. Test procedure B

211 **Input:** Sequence of bits $b_1, b_2, \dots \in \{0,1\}$ (Unlike for test procedure A the number of input bits is not fixed but depends on the input sequence.)

212 Description of test procedure B:

Step 1. Apply the uniform distribution test T6 with parameter set $(k, n, a) = (1, 100000, 0.025)$ to $b_1, b_2, \dots, b_{100000}$.

Step 2. Consider the next bits of the input sequence (i.e., those not used for the first test); split the non-overlapping pairs $(b_{100001}, b_{100002}), (b_{100003}, b_{100004}), \dots$ into two disjoint sub-sequences TF_0 and TF_1 , where (b_{2j+1}, b_{2j+2}) belongs to TF_r if and only if $b_{2j+1} = r$; continue this procedure until both TF_0 and TF_1 have at least $n_1 := 100000$ elements; consider the first n_1 many 2-tuples of each sub-sequence and determine the empirical 1-step transition probabilities

$$v_emp_{(r)}(i) := |\{j \leq n_1 : (b_{2j+1}, b_{2j+2}) \in TF_{(r)}, b_{2j+2} = i\}| / n_1.$$

This basic test is passed if and only if $|v_emp_{(0)}(1) + v_emp_{(1)}(0) - 1| < a_1 := 0.02$.

Step 3. Consider the next bits of the input sequence and split the sequence of non-overlapping triplets into 4 disjoint sub-sequences TF_{00}, \dots, TF_{11} , where $(b_{3j+1}, b_{3j+2}, b_{3j+3})$ belongs to TF_{rs} if and only if $(b_{3j+1}, b_{3j+2}) = (r, s)$; continue this procedure until each sub-sequence has at least $n_2 := 100000$ elements; consider the first n_2 many 3-tuples of each sub-sequence and determine the empirical 2-step transition probabilities

$$v_emp_{(r,s)}(i) := |\{j \leq n_2 : (b_{3j+1}, b_{3j+2}, b_{3j+3}) \in TF_{rs}, b_{3j+3} = i\}| / n_2;$$
for each $s \in \{0,1\}$, compare $v_emp_{(0,s)}$ and $v_emp_{(1,s)}$ with test T7 at the significance level $\alpha_2 := 0.0001$ for equality.

Step 4. Consider the next bits of the input sequence and split the sequence of non-overlapping quadruples into 8 disjoint sub-sequences $TF_{000}, \dots, TF_{111}$ where
 $(b_{4j+1}, b_{4j+2}, b_{4j+3}, b_{4j+4})$ belongs to TF_{rst} if and only if $(b_{4j+1}, b_{4j+2}, b_{4j+3}) = (r, s, t)$; continue this procedure until each sub-sequence has at least $n_3 := 100000$ elements; consider the first n_3 many 4-tuples of each sub-sequence and determine the empirical 3-step transition probabilities

$$v_emp_{(r,s,t)}(i) := |\{j \leq n_3 : (b_{4j+1}, b_{4j+2}, b_{4j+3}, b_{4j+4}) \in TF_{rst}, b_{4j+4} = i\}| / n_3;$$
for each $s, t \in \{0,1\}^2$, compare $v_emp_{(0,r,s)}$ and $v_emp_{(1,r,s)}$ with test T7 at the significance level $\alpha_3 := 0.0001$ for equality.

Step 5. Apply the entropy test (test T8) with the parameters $L = 8, Q = 2560$ and $K = 256000$ to the next elements of the input sequence. Test T8 is passed if the test variable $f > 7.976$.

213 Evaluation rule for Test Procedure B:

- a) If all $9 (= 1 + 1 + 2 + 4 + 1)$ basic tests have been passed, test procedure B is passed. For an ideal RNG this happens with probability ≈ 0.9998
- b) If more than one basic test has failed, test procedure B is failed. For an ideal RNG this happens with probability ≈ 0 .
- c) If exactly one basic test has failed, the statistician (evaluator, etc.) applies test procedure B once more to new das-random numbers and applies evaluation rules a) and b). If within the second run of test procedure B exactly one basic test fails, test procedure B is failed. A second repetition of test procedure B is not allowed. For an ideal RNG the probability that test procedure B finally fails is almost 0.

214 Goals and justification: Test procedure B usually is applied to binary-valued das-random numbers of PTRNGs. The goal is to ensure that the entropy per das-bit is sufficiently large. A small bias and slight one-step dependencies are permitted, but no significant more-step dependencies. This means that if the first predecessor of a bit (or the first two predecessors) are identical, all 2-last (or 3-last predecessors) shall induce the same 2-step (or 3-step) transition probability. Moreover, although the stochastic model of the das-random numbers should exclude possible long-term correlations, the evaluator shall search for possible long-term correlations. If these requirements are fulfilled and the one-step transition probabilities are negligible, the test value f from Test T8 provides an entropy estimator.

2.4.5. Additional Statistical Tests

- 215 This section discusses how statistical analysis of the RNG by standard test suites like those described above may be extended by additional test suites or by more specific tests. These statistical tests could be useful, e.g., for the analysis of random entropy sources if the hypothesis of uniform distribution is rejected and digitized random signals need post-processing.

2.4.5.1. NIST RNG test suite

- 216 The U.S. National Institute of Standards and Technology (NIST) developed a test suite for RNGs used for cryptographic purposes. The test suite is available from the NIST RNG project website http://csrc.nist.gov/groups/ST/toolkit/rng/documentation_software.html. It contains 16 tests (cf. for details to [STS] or http://csrc.nist.gov/groups/ST/toolkit/rng/stat_tests.html): Frequency (Monobit) Test, Frequency Test within a Block, Runs Test, Test for the Longest-Run-of-Ones in a Block, Binary Matrix Rank Test, Discrete Fourier Transform (Spectral) Test, Non-overlapping (Aperiodic) Template Matching Test, Overlapping (Periodic) Template Matching Test, Maurer's "Universal Statistical" Test, Linear Complexity Test, Serial Test, Approximate Entropy Test, Cumulative Sums (Cusums) Test, Random Excursions Test, and Random Excursions Variant Test.
- 217 The tests calculate the p-value for the test values observed. If the computed p-value is < 0.01 , then it is recommended to conclude that the sequence is non-random. Otherwise, conclude that the sequence is random. The tester needs 2^{30} (=1073741824) bits to run the NIST test suite with recommended parameters.

Table 11: Recommended parameter settings for the NIST test suite

Test	Configuration item	Setting
All tests	Bits per sequence	1000000
	Number of sequences (sample size)	1073
Frequency Test within a Block	Block length	20000
Non-overlapping template test	Template length	10
Overlapping template	Block length	10
Maurer's "Universal Statistical" test	Test block length L	7
	Initialization steps	1280
Approximate Entropy Test	Block length	8
Linear Complexity Test	Block length	1000
Serial Test	Block length	16

2.4.5.2. Contingency table test

218 This section describes the method of contingency table to test the Null hypothesis of independence of consecutive bits strings in the stationary RNG output by means of a contingency table (also known as “test of homogeneity”). Note that test T7 is a special case. Chapter 5.1 of this document provides a convenient R-script for analysis of binary data.

219 Let $\{B_i\}_{i \in N}$ denote a sequence of stationary random variables that describe the output of the RNG. By definition of independence, a string of y consecutive bits is independent of the x preceding bits if and only if for any binary string $(b_{y+x}, \dots, b_1) \in \{0,1\}^{y+x}$ and any positive integer t (stationarity property)

$$\begin{aligned} P\{(B_{y+x+t}, \dots, B_{1+t}) = (b_{x+y}, \dots, b_1)\} = \\ = P\{(B_{y+x+t}, \dots, B_{x+1+t}) = (b_{x+y}, \dots, b_{1+x})\} \cdot P\{(B_{y+x}, \dots, B_{1+t}) = (b_x, \dots, b_1)\} \end{aligned} \quad (56)$$

holds. Stationarity clearly implies

$$P\{(B_k, \dots, B_1) = (b_k, \dots, b_1)\} = P\{(B_{k+t}, \dots, B_{1+t}) = (b_k, \dots, b_1)\} \quad (57)$$

for any values b_k, \dots, b_1 and for any positive integers k and t . The basic idea is counting bit strings in the RNG output and testing the frequency as an estimate of the probabilities against the formula (55).

220 Suppose we generate a binary sequence $\{b_i\}_{i \in \overline{0, M}}$ and want to test whether y bits depend on x bits generated before. We divide this sequence into n non-overlapping strings $(b_{y+x+l_k-1}, \dots, b_{x+l_k}, b_{x+l_k-1}, \dots, b_{l_k})$ of $x+y$ bits, $k \in \overline{0, n-1}$. Let n_{ij} denote the observed frequency of strings, where the left y bits interpreted as dual number equal i , $i = (b_{y+x+l_k-1}, \dots, b_{x+l_k})_2$, and the right x bits interpreted as dual number equal j , $j = (b_{x+l_k-1}, \dots, b_{l_k})_2$. Denote further $n_{i\bullet} = \sum_{j=0}^c n_{ij}$ and $n_{\bullet j} = \sum_{i=0}^r n_{ij}$, where for short, $r = 2^y - 1$ and $c = 2^x - 1$. Then $n_{i\bullet}/n$ is the relative frequency of the strings that left parts hold $i = (b_{y+x+l_k-1}, \dots, b_{x+l_k})_2$, and $n_{\bullet j}/n$ is the relative frequency of the strings that right parts hold $j = (b_{x+l_k-1}, \dots, b_{l_k})_2$. Following (55), one expects $n_{ij}/n \approx (n_{i\bullet}/n)(n_{\bullet j}/n)$ in case of independence of the left and right bits.

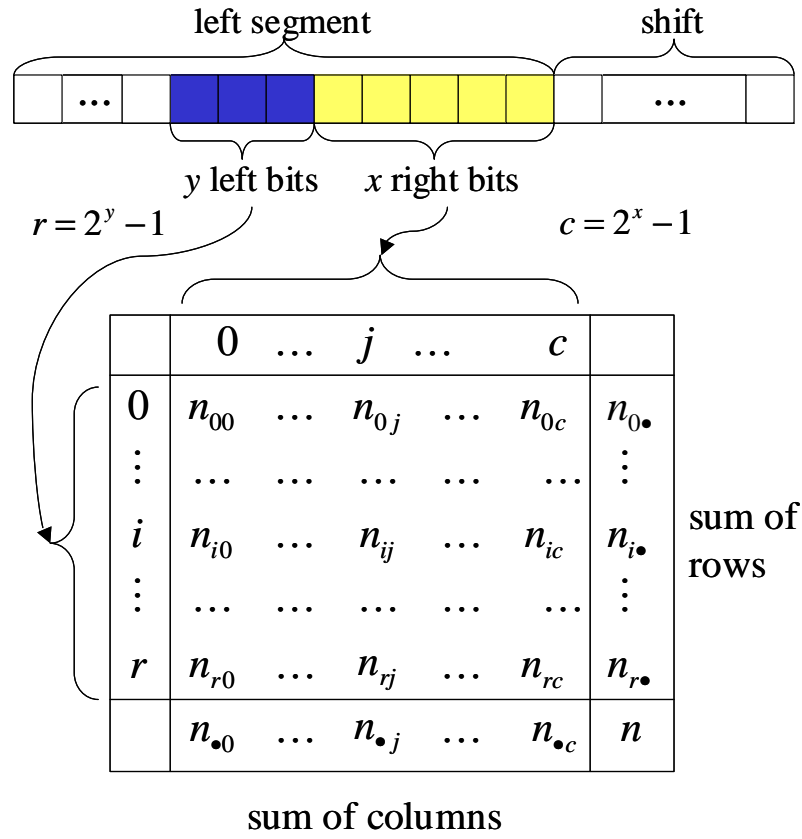


Figure 2: Contingency table for counts of consecutive bits strings

- 221 More precisely, if the bit strings are independent, the test value $\hat{\chi}^2$ will be χ^2 -distributed with rc degrees of freedom,

$$\hat{\chi}^2 = \sum_{i=0}^r \sum_{j=0}^c \left[\frac{\left(n_{ij} - \frac{n_{i\bullet} n_{\bullet j}}{n} \right)^2}{\frac{n_{i\bullet} n_{\bullet j}}{n}} \right] = n \left[\sum_{i=0}^r \sum_{j=0}^c \frac{n_{ij}^2}{n_{i\bullet} n_{\bullet j}} - 1 \right] \quad (58)$$

with density

$$r(x, d) = \begin{cases} \frac{x^{d/2-1} e^{-x/2}}{2^{d/2} \Gamma(d/2)}, & x > 0 \\ 0 & x \leq 0 \end{cases} \quad (59)$$

- 222 Note that for small n and $\hat{\chi}^2 < 0.1$, the χ^2 -distribution is only a coarse approximation. The expected values of n_{ij} for $i = 0, 1, \dots, r$ and $j = 0, 1, \dots, c$ shall be greater than 1. The statistical literature like [Craw07], [SaHe06] may be consulted for further details about usage of contingency tables.

223 Chapter 5.2 discusses examples that apply contingency table tests.

3. Security Functional Requirements - Family FCS_RNG

- 224 This chapter guides PP and ST writers how to describe the security functional requirements for RNG. The extended family FCS_RNG describes SFR for RNGs for cryptographic and other applications in PP and ST.

3.1. Definition of FCS_RNG

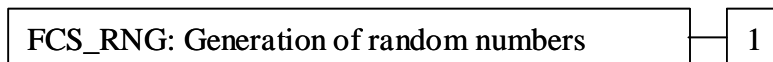
- 225 This section describes the functional requirements for the generation of random numbers, which may be used as secrets for cryptographic purposes or authentication. The IT security functional requirements for a TOE are defined in an additional family (FCS_RNG) of the Class FCS (Cryptographic support).

FCS_RNG Generation of random numbers

Family Behaviour

This family defines quality requirements for the generation of random numbers that are intended to be used for cryptographic purposes.

Component levelling:



FCS_RNG.1 Generation of random numbers, requires that the random number generator implements defined security capabilities and that the random numbers meet a defined quality metric.

Management: FCS_RNG.1

There are no management activities foreseen.

Audit: FCS_RNG.1

There are no actions defined to be auditable.

FCS_RNG.1 Random number generation

Hierarchical to: No other components.

Dependencies: No dependencies.

FCS_RNG.1.1 The TSF shall provide a [selection: *physical, non-physical true, deterministic, hybrid physical, hybrid deterministic*] random number generator that implements: [assignment: *list of security capabilities*].

FCS_RNG.1.2 The TSF shall provide random numbers that meet [assignment: *a defined quality metric*].

3.2. Security capabilities of RNG types

- 226 The PP writer may and the ST writer shall perform the operations by selection of the RNG type and by assignment of security capabilities in the element FCS_RNG.1.1 and the assignment of the quality of the generated random numbers in the element FCS_RNG.1.2. The security capabilities may be assigned by identification of a predefined class that is described by a list of security capabilities and a quality metric of the provided random numbers.
- 227 The PP / ST writer may define the **RNG type** by selecting the type of the RNG (“physical”, “non-physical true”, “deterministic”, “hybrid physical” or “hybrid deterministic”) in the element FCS_RNG.1.1. The RNG types are described in section 2.2.3.
- 228 The PP / ST writer may consider typical **security capabilities** to perform the operation of the element FCS_RNG.1.1 as described below.
- 229 A *physical true* RNG uses dedicated hardware as an entropy source. It typically implements the following security capabilities: CPG.x, x=1, 2, 3, which address internal tests.
- 230 CPG.1: Total failure test of the entropy source. When a total failure has been detected no further random numbers will be output.

A total failure of the entropy source implies that the increase of the overall entropy of the output sequence by future raw random numbers is 0. A total failure test of the random source detects a breakdown of the noise source. The RNG immediately prevents the output of random numbers if a total failure of the random source has been detected. Under suitable circumstances, e.g. if random numbers have been buffered or if entropy is ‘stored’ in the internal state, a larger latency of the tot test may be allowed.

- 231 CPG.2: Online test of the raw random number sequence. When a non-tolerable statistical defect of the raw random number sequence has been detected no further random numbers will be output.

The online test detects non-tolerable statistical defects of the raw random number sequence and prevents the output of future random numbers. Non-tolerable statistical defects violate the quality of random numbers defined in FCS_RNG.1.2. A statistical online test of the raw random number sequence may be very effective to ensure the quality of output random numbers if the post-processing algorithm works correctly. The TSF test capabilities of the post-processing (e.g., known-answer test) or an online test of the internal random numbers should complete the online test of the raw random number sequence.

- 232 CPG.3: Online test of the internal random numbers. When a non-tolerable statistical defect of the internal random numbers has been detected no further random numbers will be output.

The online test detects non-tolerable statistical defects of the internal random numbers. The RNG prevents output of random numbers if non-tolerable statistical defects have been detected. A statistical online test of the internal random numbers may detect a failure of the noise source, of the digitization mechanism, or of the post-processing. If the post-processing algorithm has internal memory, it might produce pseudo-random numbers even after the noise source has completely broken down. The test shall detect non-tolerable statistical defects of the internal random numbers, or it shall be completed by an appropriate test of the raw random number sequence.

- 233 The *hybrid physical* RNG typically implements additional security capabilities CPG.x, x=4, 5, extending those of the physical RNG:

- 234 CPG.4: On average (over the time) the raw random number sequence should provide at least as much entropy to the post-processing algorithm as the output sequence can at most attain. (This upper entropy bound clearly is attained for iid uniformly distributed sequences over the output alphabet of the output function of the same length.)

Let us assume that the raw random number sequence generates entropy \mathcal{E} per time interval where the RNG outputs l internal random numbers over the alphabet R . In average the output sequence can at most contain fresh entropy \mathcal{E} but clearly not more than $l \cdot \log_2 |R|$, which would be obtained by ideal output sequences (i.e., iid uniformly distributed) on R . If $\mathcal{E} \geq l \cdot \log_2 |R|$ on average (over the time), the post-processing algorithm principally might smooth the distribution of the raw random numbers towards the distribution of independent and uniformly distributed random variables. CPG.4 formulates a necessary condition for the generation of true random numbers with maximum entropy. If $\mathcal{E} < l \cdot \log_2 |R|$ on average over the time, the RNG cannot generate random numbers with maximum entropy. To generate output that is (computationally) indistinguishable from uniformly distributed and independent random numbers, the RNG then must extend the raw random number sequence by means of a DRNG (hybrid DRNG, cf. CDG.5).

CPG.5: On average (over the time) the raw random number sequence should provide at least as much entropy to the cryptographic post-processing algorithm as the output sequence can at most attain. (This upper bound clearly is attained for iid uniformly distributed sequences over the output alphabet of the same length.) The capability CPG.5 enhances the capability CPG.4 by means of *cryptographic* post-processing. The capability CPG.4 provides a necessary condition that the internal random numbers can have maximum entropy. Defects that affect statistical properties of the raw random number sequences might reduce their entropy, and then the online test should detect these effects (non-tolerable statistical defects) (cf. CPG.2). A cryptographic post-processing algorithm hardens the internal random numbers against attacks in case of entropy defects that occur in the time between occurrence and detection by the online tests. This feature might be relevant due to limited effectiveness of the online tests.

- 235 A *deterministic* RNG produces random numbers by applying a deterministic algorithm to a randomly-selected seed and, possibly, on additional external inputs. It typically implements security capabilities CDG.x, x=1, 2, 3, 4, 5, as follows:

- 236 CDG.1: The seed of the DRNG has minimum entropy [assignment: *value*] to prevent successful guessing attacks with the assumed attack potential.

The ST writer shall identify the minimum entropy of the seed as a security feature of the RNG. The entropy shall be provided by the non-deterministic part of RNG, e.g., a physical RNG, or by an external source, e.g., during installation (instantiation) of the TOE. The entropy may be defined in terms of Min-entropy or Shannon entropy. This choice should be appropriate for the random source for seeding the RNG.

- 237 CDG.2: The DRNG provides forward secrecy.

Forward secrecy ensures that subsequent (future) values cannot be determined from current or previous output values [ISO18031]. This unpredictability property is a natural security feature of random numbers.

- 238 CDG.3: The DRNG provides backward secrecy.

Backward secrecy ensures that previous output cannot be determined from current or future output values [ISO18031].

- 239 CDG.4: The DRNG provides backward secrecy even if the current internal state is known.

Enhanced backward secrecy ensures that previous output cannot be determined from the known (compromised) current internal state, current or future output values.

- 240 CDG.5: The DRNG provides forward secrecy even if the internal state is known.

Enhanced forward secrecy ensures that subsequent (future) values cannot be determined from known (compromised) current internal state, current or previous output values. Enhanced forward secrecy may be ensured by reseeding or refreshing of the internal state initiated by a specific function of the RNG.

- 241 The *deterministic hybrid* RNG typically implements additional security capabilities CDG. x , $x = 6, 7$, extending those of a pure deterministic RNG:

- 242 CDG.6: The DRNG refreshes the internal state with an internal random source [selection: *on demand, continuously, [assignment: other method]*].

The RNG refreshes the internal state of its deterministic part by non-deterministic input from an internal random source. This capability does not quantify the amount of fresh entropy of the input data. (This requirement is weaker than a reseed; cf. CDG.7).

- 243 CDG.7: The DRNG provides reseeding of the internal state by an internal random source [selection: *on demand, automatically, [assignment: other method]*].

The RNG initializes the internal state of its deterministic part by non-deterministic input from an internal random source. The new initial state shall be independent of the current value of the internal state. Hence, the RNG estimates the entropy of the non-deterministic input to ensure a lower entropy bound for the entropy of the new internal state.

- 244 A *non-physical* RNG uses non-physical external random sources within the operational environment to generate random number output. Depending on the relation between the entropy of the input data and the output data, one may distinguish between non-physical RNG and hybrid non-physical RNG.

A *non-physical true* RNG gets fresh entropy from non-physical external random sources. The estimated entropy of these data, resp. the estimated entropy of an entropy pool, is at least as large as the amount of entropy that would be attained by output sequences that stemmed from an ideal RNG. This condition is similar as for hybrid physical true RNGs, which use internal noise sources, cf. para. 228 on page 67.

- 245 A *non-physical non-deterministic* RNG typically implements the security capability CNG.1 as follows:

- 246 CNG.1: Examination of the external input [assignment: *types of input data*] to ensure sufficient entropy to generate random numbers.

The NPTRNG tests the external input data to ensure that they provide sufficient entropy for the internal state and thus finally for the output data. The entropy estimate for the input data is (usually) based on heuristic assumptions on the nature of the input data. The IT environment of the TOE shall ensure that these assumptions (as described in the security objectives for the IT environment in the PP / ST) are indeed valid. For example, the NTPRNG design may exploit the user's keystrokes as an entropy source, and it computes some entropy estimate on the basis of the pressed keys and the time instants when they are struck. This entropy estimator might not detect a simulation (the repetition) of such keystrokes by malicious software that shall allow to reproduce known internal states.

- 247 A *non-physical true* RNG typically implements an additional security capability on guaranteed entropy as follows:

CNG.2: Each random number has fresh entropy of [assignment: *quantity of entropy*].

This quality metric may assign a lower min-entropy bound per random number, which allows a direct estimate of the minimum guess work needed for the generated random numbers as part of the vulnerability analysis (cf. section 2.3.2 on pp. 32). This quality metric may assign Shannon entropy per random number, which under suitable circumstances (stochastic model!) is easy to estimate with test procedure suite B. This value may be used to estimate the guess work for the generated random numbers in vulnerability analysis e.g. if the bits are independent (cf. paragraph 114 on page 29 for definition of independence).

- 248 The overall entropy of output sequences from true RNGs increases with their length. In contrast, the entropy of output sequences from DRNGs is clearly limited by the seed entropy. Therefore, if the TOE shall generate cryptographic keys that shall resist high attack potential:

- the TSF may call the TRNG until sufficient entropy is collected
- the SFR shall require 100-bit entropy of DRNG in element FCS_RNG.1.2

- 249 The quality of the random numbers produced may be described as follows:

- 250 QRN.1: The RNG generates numbers that are not distinguishable from ideal random numbers by means of [selection: *test procedure A*, [assignment: *other test suite*]].

The quality metric “are not distinguishable from ideal random numbers by means of test procedure A” is easy to test, but its application for the vulnerability analysis might not suffice. This would mean, for instance, that if test procedure A has been selected on basis of the statistical properties that are considered by this test suite the designer should be able to estimate the guesswork for the output sequence.

- 251 QRN.2: Statistical test suites cannot practically distinguish the internal random numbers from output sequences of an ideal RNG.

This quality metric is suitable for the vulnerability assessment, but difficult to test. The demonstration of this quality metric will use other theoretical evidence as well. E.g. if the RNG uses cryptographic post-processing the standard statistical test might not detect defects of the raw random numbers or entropy defects of the output.

3.3. Rationale for definition of the extended component

252 The CC part 2 defines the component FIA_SOS.2, which is similar to FCS_RNG.1, as follows:

FIA_SOS.2 TSF Generation of secrets

Hierarchical to: No other components.

Dependencies: No dependencies.

FIA_SOS.2.1 The TSF shall provide a mechanism to generate secrets that meet [assignment: *a defined quality metric*].

FIA_SOS.2.2 The TSF shall be able to enforce the use of TSF generated secrets for [assignment: *list of TSF functions*].

253 The CC part 2, annex G.3 [CCV31_2], states: “This family defines requirements for mechanisms that enforce defined quality metrics on provided secrets, and generate secrets to satisfy the defined metric“. Even the operation in the element FIA_SOS.2.2 allows listing the TSF functions using the generated secrets. Because all applications discussed in annex G.3 are related to authentication, the component FIA_SOS.2 is also intended for authentication purposes while the term “secret” is not limited to authentication data (cf. CC part 2, paragraphs 39-42).

254 Paragraph 685 in the CC part 2 [CCV31_2] recommends use of the component FCS_CKM.1 to address random number generation. However, this may hide the nature of the secrets used for key generation and does not allow describing random number generation for other cryptographic methods (e.g., challenges, padding), authentication (e.g., password seeds), or other purposes (e.g., blinding as a countermeasure against side channel attacks).

255 The component FCS_RNG addresses general RNG, the use of which includes but is not limited to cryptographic mechanisms. FCS_RNG allows to specify requirements for the generation of random numbers including necessary information for the intended use. These details describe the quality of the generated data where other security services rely on. Thus by using FCS_RNG a ST or PP author is able to express a coherent set of SFRs that include or use the generation of random numbers as a security service.

4. Pre-defined RNG Classes

256 This chapter defines classes of PTRNG, NPTRNG and DRNG for typical use cases. These classes are hierarchically organized. The definition of these classes is accompanied by application notes explaining their security capabilities and quality metrics.

257 This chapter also identifies the minimum information expected from the technical point of view to fulfil the assurance requirements addressed in the PP or ST. The following paragraphs identify the lowest assurance elements describing the relevant evidence. If the ST requires a component hierarchical to the mentioned assurance component, the equivalent element of the hierarchically-higher component will require analogue information.

4.1. Overview of pre-defined RNG classes

258 This section defines pre-defined RNG classes based on the component FCS_RNG.1. It describes the specific evidence necessary for the evaluation of an RNG class based on a chosen EAL according to the CEM.

259 The security functional requirements of the RNG class are described by means of the component FCS_RNG.1 where

- the RNG type is selected,
- the security capabilities are assigned,
- the quality metric is assigned.

260 The developer shall provide specific information describing how the RNG meets the assurance requirements expressed in a PP or ST for the security functional requirements described in FCS_RNG.1.

261 If the ST defines in FCS_RNG.1 that the TOE supports one of the pre-defined RNG classes, the developer is expected to provide specific information and evidence for the assigned security capabilities and quality of the random numbers according to the content and presentation of elements for the assurance components selected in the ST. If the ST requires a component hierarchical to the mentioned assurance component, the equivalent element of the hierarchically-higher component will require analogue information.

262 The pre-defined RNG classes relate to those described in [AIS20] and [AIS31] as follows (coarse comparisons):

RNG class	Comparable to [AIS20] or [AIS31] class	Comment
PTG.1	AIS31, P1	Physical RNG with internal tests that detect a total failure of the entropy source and non-tolerable statistical defects of the internal random numbers
PTG.2	AIS31, P2	PTG.1, additionally a stochastic model of the entropy source and statistical tests of the raw random numbers

RNG class	Comparable to [AIS20] or [AIS31] class	Comment
		(instead of the internal random numbers)
PTG.3	No counterpart	PTG.2, additionally with cryptographic post-processing (hybrid PTRNG)
DRG.1	AIS20, K2, partly K3	DRNG with forward secrecy according to [ISO18031]
DRG.2	AIS20, K3	DRG.1 with additional backward secrecy according to [ISO18031]
DRG.3	AIS20, K4	DRG.2 with additional enhanced backward secrecy
DRG.4	No counterpart	DRG.3 with additional enhanced forward secrecy (hybrid DRNG)
NTG.1	No counterpart	Non-physical true RNG with entropy estimation

263 The following figures illustrate different classes of RNGs. We point out that other realizations of these classes are possible. The pictures show in pink the total failure tests, in red the online tests and in dark green known-answer-tests.

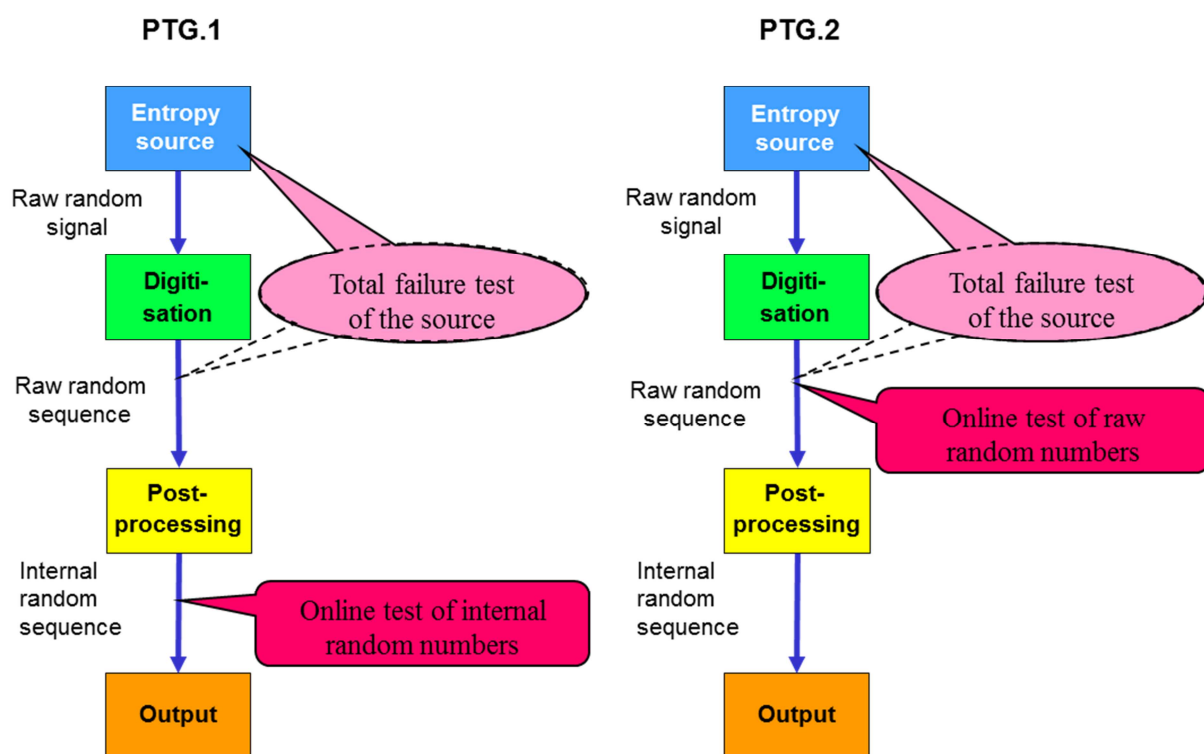


Figure 3: Example of PTRNGs that belong to the pre-defined classes PTG.1 and PTG.2

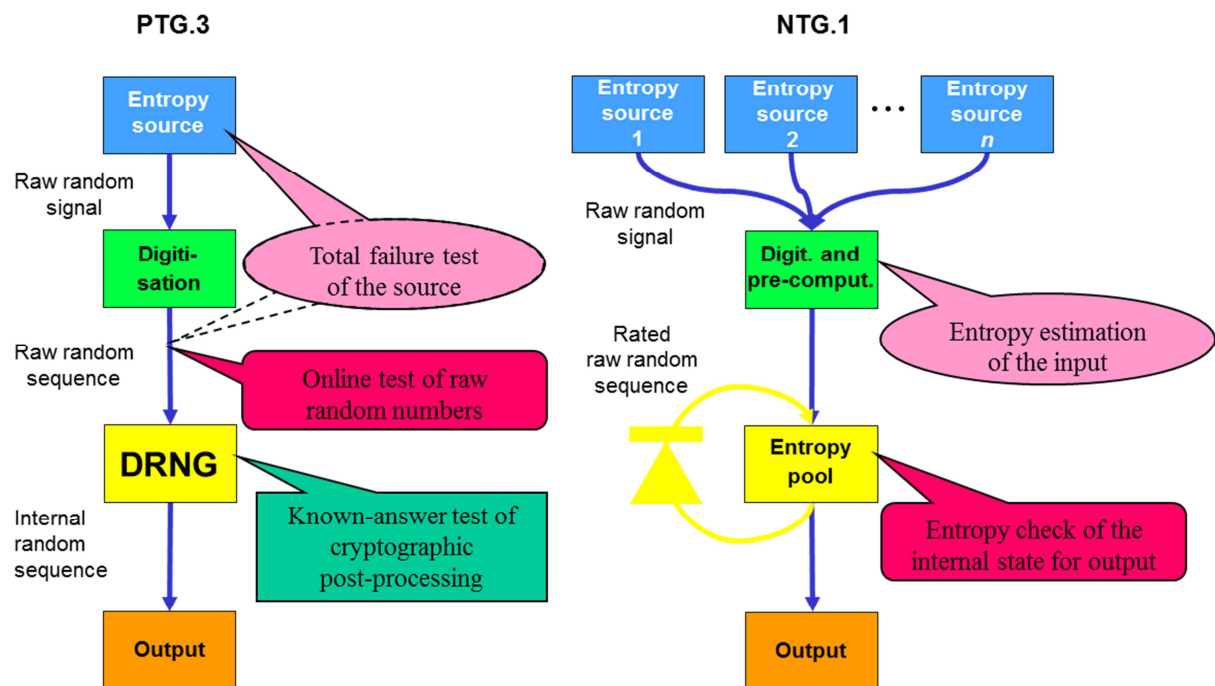


Figure 4: Example of a PTG.3 and NTG.1 that belongs to the pre-defined class PTG.3 and NTG.1

- 264 The DRNG classes are illustrated by the following figures (φ - state transition function, ψ - output function, $A \rightarrow \oplus B$ - symbol for a one-way function for the state transition function φ and the extended output function ψ^*):

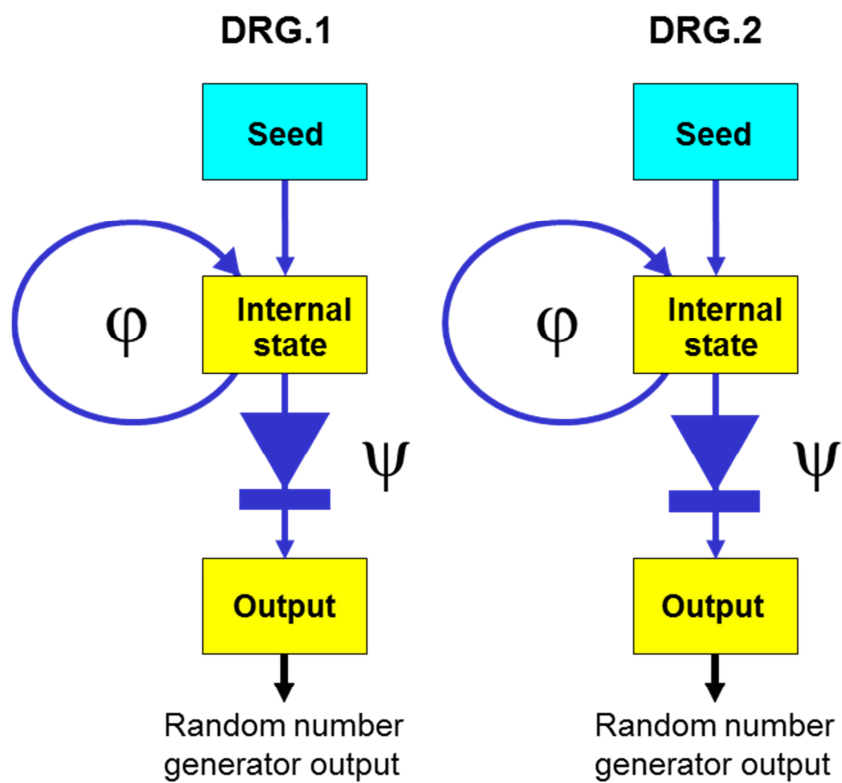


Figure 5: Examples of DRNGs that belong to the pre-defined classes DRG.1 and DRG.2

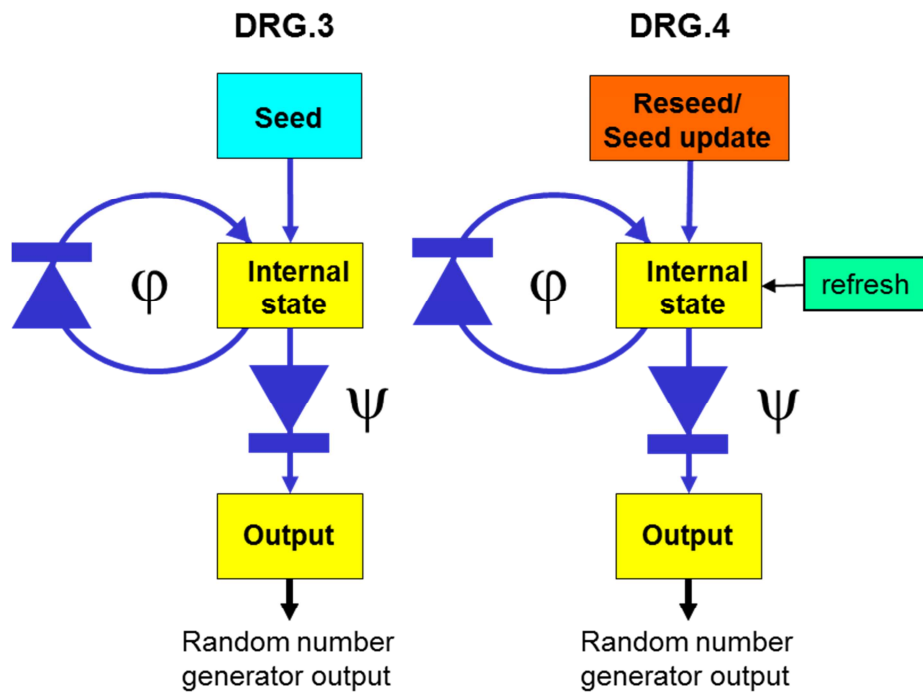


Figure 6: Examples of DRNGs that belong to the pre-defined classes DRG.3 and DRG.4

4.2. General Remarks (Exemplary applications, side-channel attacks, fault attacks)

- 265 In the description of the particular pre-defined RNG classes below possible (exemplary) cryptographic applications are mentioned that shall motivate the class definitions. We point out that these examples are informative only. In such or in related real-world applications there might be special (additional) requirements on the random numbers, which make it necessary or at least advisable to select an RNG from a higher class. Similarly, possible progress in cryptanalysis might implicate the selection of higher RNG classes in the future. The designer of an application is responsible for the choice of an appropriate RNG.
- 266 Implementation attacks, in particular side-channel attacks and fault attacks, constitute serious threats against cryptographic implementations. Principally, also RNGs might be concerned.
- 267 (DRNGs): Although irrelevant from a cryptanalytic (logical) point of view in the light of implementation attacks (in particular, of side-channel attacks) we recommend to use different instantiations of a DRNG for different applications, and maybe even for different tasks within one application (e.g., random numbers remain secret, random numbers remain unknown apart from legitimate users, random numbers might be disclosed later, random numbers are open). Otherwise the attacker might try to perform a side-channel attack, which exploits the known random numbers to recover the internal state, which determines at least all future random numbers.
- 268 (DRNG): To prevent side-channel attacks it might be recommendable not to keep secret parts of the internal state constant. This would be the case, for instance, for a DRNG that uses a block cipher with constant secret key in OFB mode.
- 269 (PTG.3) + (DRG.4): Hybrid RNGs combine security properties of both PTRNGs and DRNGs. One may hope that the combination of both an analogue part and algorithmic post-processing might also help to harden RNG implementations against side-channel attacks and fault attacks.
- 270 From a logical point of view statistical tests on the output data of a cryptographic post-processing algorithm are pointless. However, online tests might serve as additional security measure that might detect fault attacks.

4.3. Class PTG.1

- 271 The class PTG.1 defines requirements for PTRNGs, which might be used to generate random numbers for cryptographic applications, where the random numbers need not meet any unpredictability properties, neither in the past nor in the future.
- 272 The required quality metric of PTG.1 does not prevent that the random numbers might be guessed.

4.3.1. Security functional requirements for the RNG class PTG.1

- 273 The functional security requirements of the class PTG.1 are defined by component FCS_RNG.1 with specific operations as given below.

FCS_RNG.1 Random number generation (Class PTG.1)

FCS_RNG.1.1 The TSF shall provide a *physical*²⁰ random number generator that implements:

(PTG.1.1) *A total failure test detects a total failure of the entropy source immediately when the RNG has started. When a total failure is detected, no random numbers will be output.*

(PTG.1.2) *If a total failure of the entropy source occurs while the RNG is being operated, the RNG [selection: prevents the output of any internal random numbers that depend on raw random numbers generated after the total failure occurred, generates the internal random numbers with a post-processing algorithm of class DRG.1 until the output of further internal random numbers is prevented].*

(PTG.1.3) *The online test detects non-tolerable statistical defects of the internal random numbers. The online test is [selection: triggered externally, applied after regular intervals, applied continuously, applied upon specified internal events]. When a defect is detected, the output of further random numbers is prevented.*

(PTG.1.4) *Within one year of typical use, the probability that an online alarm occurs is in the order of 10^{-6} or larger if the RNG works properly.²¹*

FCS_RNG.1.2 The TSF shall provide [selection: *bits, octets of bits, numbers* [assignment: *format of the numbers*]] that meet:

(PTG.1.5) *Test procedure A [assignment: *additional standard test suites*] does not distinguish the internal random numbers from output sequences of an ideal RNG.²²*

4.3.2. Application notes

274 An RNG of class PTG.1 shall generate true random numbers based on an entropy source. A total breakdown of the entropy source causes zero entropy for future raw random numbers. Moreover, a failure of the digitization of the analogue signal of the post-processing algorithm might cause defects of the internal random numbers. The total failure test and the online test shall detect those types of errors, and the TOE shall prevent output of random numbers of poor quality.

275 The total failure test may detect

(i) the breakdown of the entropy source by analyzing the raw random signal, or

(ii) the total failure of the entropy source including the digitization by analyzing the raw random number sequence.

²⁰ [selection: *physical, non-physical true, deterministic, hybrid physical, hybrid deterministic*]

²¹ [assignment: *list of security capabilities*]

²² [assignment: *a defined quality metric*]

The total failure test must consider the physical principle of the entropy source. In case (i), the total failure test may measure the physical entropy effect. In case (ii), the breakdown analysis of the entropy source usually identifies characteristic patterns in the raw random signal (e.g., constant sequences, meanders) that can be detected quickly by a suitable test.

- 276 If a total failure of the entropy source occurs the total failure test might need some time to detect this failure, and the post-processing mechanism might delay the effect of the non-random raw random numbers on the output random numbers. The FCS_RNG.1.1 clause (PTG.1) explains the response of the RNG when a total failure of the entropy source has been detected: preventing the output of random numbers. FCS_RNG.1.1 clause (PTG.1.2) addresses the time between the occurrence and the detection of a total failure. The selection covers two types of post-processing algorithms:
- 277 Case (i): The post-processing mechanism generates internal random numbers that depend only on some fixed number of raw random signals (e.g., the internal random number might be calculated from a raw random number sequence that is stored in a first-in, first-out memory). After a total failure, the raw random signal loses any randomness, and the generated raw random numbers affect the generated internal random numbers in a deterministic way. After some time, the internal random numbers are generated only from non-random raw random numbers. No later than this point the RNG shall prevent the output of internal random numbers because they have *completely been generated after a total failure of the entropy source*.
- 278 Case (ii): The internal random numbers might principally depend on an unlimited number of raw random numbers (memory!), although the internal state is finite (e.g., the internal random numbers are calculated from raw random numbers that are stored in a feedback shift register). Therefore, the internal random numbers still depend on “truly random” raw random numbers, which have been generated before the entropy source has broken down.. The RNG *generates output with a post-processing mechanism of class DRG.1 until the RNG prevents the output of internal random numbers*, which ensures the quality of the internal random numbers during the time between occurrence and detection of a total failure, as required by element FCS_RNG.1.1 clause (PTG.1.2).
- 279 The online test shall detect non-tolerable statistical defects of the internal random numbers. As distinct from a total failure, these defects usually can be detected only by statistical tests. The FCS_RNG.1.1 clause (PTG.1.3) addresses the conditions under which the online tests shall be executed. The FCS_RNG.1.1 clause (PTG.1.4) ensures that the online test is sharp enough to detect statistical defects. The *course of one year of typical use of the RNG* is defined by the use of the TOE or by additional evidence provided by the developer. If the internal random numbers pass the online test, the TOE design shall ensure that the output random numbers meet the quality described in (PTG.1.5).
- 280 If the tot test and / or the online test are not part of the TOE but to be implemented later as an external security measure the applicant must submit a specification of the test(s), a justification for effectiveness and a reference implementation. The suitability of the tot test and the online test shall be verified based on the reference implementation. In the positive case the RNG is said to be PTG.1 compliant under the condition that the final implementation meets the specification in the user manual (to be checked in a composite evaluation).
- 281 The developer may or may not assign additional standard test suites (i.e. the assignment may be empty) in the element FCS_RNG.1.2 clause (PTG.1.5). The element FCS_RNG.1.2 clause (PTG.1.5) demands that the application of test procedure A and - if assigned – of additional

standard test suites does not reject the null hypothesis “the internal random numbers were generated by an ideal random generator”. The same requirement is demanded for classes PTG.2 and DRG.1. The efforts of testing depend on the claimed resistance (in the ST) against attacks (cf. selected component of the family AVA_VAN). The evaluator may apply additional statistical tests as penetration tests. Note that this requirement does not necessarily imply that the rejection probability for the internal random numbers equals the rejection probability of sequences from ideal RNGs. Moreover, even this enhanced property is weaker than Requirements PTG.3.8, DRG.2.5, DRG.3.5 and DRG.4.7.

4.4. Class PTG.2

282 The class PTG.2 defines requirements for RNGs intended to generate, for example:

- cryptographic keys (e.g., for block ciphers or for RSA),
- random padding bits,
- seeds for DRNGs of the classes DRG.1, DRG.2, DRG.3 or DRG.4, or
- cryptographic applications with similar requirements (in particular, secrecy of the random numbers)

(cf. also par. 265). Roughly speaking, PTG.2 conformant RNGs generate high-entropy random numbers. These random numbers may not be practically indistinguishable from independent uniformly distributed random numbers (output from an ideal RNG). The entropy shall in particular prevent successful guessing attacks. In particular, class PTG.2 includes the applications for class PTG.1.

283 The PTG.2 class specification allows that the internal random numbers may have a small entropy defect, e.g. due to a small bias. When used for the generation of ephemeral keys for DSA signatures or ECDSA signatures, for example, a potential attacker might try to combine information from many signatures, resp. on several ephemeral keys. Although no concrete attack is known to date to stay on the safe side it might be favourable to use a class PTG.3 RNG as a measure of precaution. Similar considerations hold for any other applications, too, where an attacker might be able to collect information on many internal random numbers. The practical relevance of this potential vulnerability clearly depends on the concrete application.

284 The PTG.2 class specification does not require a post-processing algorithm if the raw random numbers are already good enough. However, even then it might be reasonable to apply a post-processing algorithm with memory. The post-processing algorithm might smooth a bias or short-term dependencies. Even if it is not data-compressing the entropy of its internal state might compensate entropy defects of the raw random numbers provided that in the course of the time more random raw bits are fed into the post-processing algorithm than outputted by the PTRNG.

285 For a PTG.2 RNG the post-processing algorithm (if it exists) may not be cryptographic. If the post-processing algorithm belongs to class DRG.2, resp. even to DRG.3, (viewed as a free-running DRNG) this extends the reaction time upon a total failure of the entropy source (PTG.2.2), resp. the PTRNG may even belong to class PTG.3 (cf. PTG.3.6).

4.4.1. Security functional requirements for the RNG class PTG.2

286 Functional security requirements of the class PTG.2 are defined by component FCS_RNG.1 with specific operations as given below.

FCS_RNG.1 Random number generation (Class PTG.2)

FCS_RNG.1.1 The TSF shall provide a *physical*²³ random number generator that implements:

(PTG.2.1) *A total failure test detects a total failure of entropy source immediately when the RNG has started. When a total failure is detected, no random numbers will be output.*

(PTG.2.2) *If a total failure of the entropy source occurs while the RNG is being operated, the RNG [selection: prevents the output of any internal random number that depends on some raw random numbers that have been generated after the total failure of the entropy source, generates the internal random numbers with a post-processing algorithm of class DRG.2 as long as its internal state entropy guarantees the claimed output entropy].*

(PTG.2.3) *The online test shall detect non-tolerable statistical defects of the raw random number sequence (i) immediately when the RNG has started, and (ii) while the RNG is being operated. The TSF must not output any random numbers before the power-up online test has finished successfully or when a defect has been detected.*

(PTG.2.4) *The online test procedure shall be effective to detect non-tolerable weaknesses of the random numbers soon.*

(PTG.2.5) *The online test procedure checks the quality of the raw random number sequence. It is triggered [selection: externally, at regular intervals, continuously, applied upon specified internal events]. The online test is suitable for detecting non-tolerable statistical defects of the statistical properties of the raw random numbers within an acceptable period of time.²⁴*

FCS_RNG.1.2 The TSF shall provide [selection: *bits, octets of bits, numbers* [assignment: *format of the numbers*]] that meet:

(PTG.2.6) *Test procedure A [assignment: additional standard test suites] does not distinguish the internal random numbers from output sequences of an ideal RNG.*

(PTG.2.7) *The average Shannon entropy per internal random bit exceeds 0.997.*

4.4.2. Application notes

287 The class PTG.2 includes the requirements of class PTG.1 for the security capabilities (PTG.1.1) (PTG.1.3) and (PTG.1.4) as defined in the element FCS_RNG.1.1. It reformulates (PTG.1.2) in the form of (PTG.2.2). The security capability of an online test of the internal

²³ [selection: *physical, non-physical true, deterministic, hybrid physical, hybrid deterministic*]

²⁴ [assignment: *list of security capabilities*]

random numbers is replaced by online tests of the raw random number sequence, i.e., tests are oriented towards the entropy source. It adds requirements for the minimum entropy of random numbers.

- 288 The element FCS_RNG.1.1 clause (PTG.2.2) addresses security capabilities that shall ensure the quality of outputted internal random numbers between the occurrence and the detection of a total failure of the entropy source. It allows the output of some internal random numbers that depend on some raw random numbers with zero entropy *if the post-processing algorithm can be viewed as a DRNG of class DRG.2*. If the PTRNG had worked properly before the total breakdown of the entropy source, the entropy of the internal state should be maximal, which limits the maximal number of internal random numbers that may be outputted (PTG.2.7). Let t denote the ratio of the bit length of the internal state of the DRNG and the bit size of the internal random numbers. Under the assumption that the DRNG has appropriate mixing properties the RNG may output at most t internal random numbers that depend on some raw random numbers that have been generated after the total failure of the entropy source in order to ensure sufficient entropy of the internal random numbers. (Recall that PTG.1.2 only demands a post-processing algorithm of the weaker class DRG.1, and there is no upper bound for the outputted internal random numbers.)
- 289 The entropy source might be affected, for instance, by aging, tolerances of components, environmental stress, or events like the failure of particular components (e.g., if the PTRNG comprises more than one entropy source). These factors might decrease the entropy of the raw random signal, but might not result in total failure of the (overall) entropy source. Such defects shall be detected by the online test of the raw random number sequence required in the element FCS_RNG.1.1 clause (PTG.2.3).
- 290 The element FCS_RNG.1.1 requires in clause (PTG.2.3) an online test that detects non-tolerable statistical defects of the *raw random numbers*. The online test of the raw random number sequence in (PTG.2.3) analyses the digitized random signal before post-processing while online tests of the internal random numbers in (PTG.1.3) consider the post-processed random numbers. Note that the online tests for the raw random numbers shall be selected such that the tested statistical properties correspond to possible entropy defects. This is usually not the case for blackbox tests. The rationale for the online tests of the raw random number sequence must be based on stochastic models of the entropy source. The online test procedure shall consider the statistical properties of the raw random numbers. Thus the online tests are usually applied to the raw random numbers. Note, however, that this requirement does not categorically exclude that (in exceptional cases) the online test might operate on analogue values (e.g., to estimate the jitter) or on the internal random numbers. Example: Assume that the RNG generates iid (maybe strongly) biased random bits (\leftarrow stochastic model), and that the post-processing algorithm xors non-overlapping pairs of random bits. Then the internal random numbers are also iid, and the online test could be applied to the internal random numbers as well, indirectly verifying the quality of raw random numbers. In this scenario information on statistical properties of the raw random numbers (here: bias) can easily be translated into information on the raw random numbers, and this essentially corresponds with entropy. However, even then it might be reasonable to test the raw random numbers since a large deviation of the distribution of the raw random numbers implies only a significantly smaller deviation in the internal random numbers, which might be more difficult to detect.
- 291 If the tot test and / or the online test are not part of the TOE but to be implemented later as an external security measure the applicant must submit a specification of the test(s), a justification for effectiveness and a reference implementation. The suitability of the tot test and the online

test shall be verified based on the reference implementation. In the positive case the RNG is said to be PTG.2 compliant under the condition that the final implementation meets the specification in the user manual (to be checked in a composite evaluation).

- 292 If the post-processing algorithm does not reduce the average entropy per bit, the average entropy per internal random bit equals at least the average entropy per raw random bit.
- 293 The term “power-up test” in clause (PTG.2.3) addresses the online test of the raw random number sequence when the RNG is started after the TOE has been powered up (after being powered off), reset, rebooted etc., or after the operation of the RNG has been stopped (e.g., to reduce the power consumption of the TOE). If the PTRNG does not apply a post-processing algorithm (or formally: if it applies the identity mapping), the internal random numbers coincide with the raw random signals. In this case, the clauses (PTG.2.3) and (PTG.2.5) will cover (PTG.1.3) and (PTG.1.4). If the RNG applies a post-processing algorithm, the raw random signals and the internal random numbers usually have different statistical properties. If the raw random number sequence passes the online test and if the post-processing algorithm works correctly, the internal random numbers will have appropriate properties (in particular, sufficient entropy). A temporary or permanent failure in the implementation of the post-processing algorithm might result in non-tolerable entropy defects of the internal random numbers (at least the post-processing algorithm does not work as expected). For many post-processing algorithms, it seems hardly possible to implement effective statistical tests on the internal random numbers, e.g., because post-processing induces complicated dependencies between the internal random numbers. The correctness of the (deterministic) post-processing may be checked while the PTRNG is in operation, e.g., by a known answer test (cf. FPT_TST.1 TSF test).
- 294 The element FCS_RNG.1.2 clause (PTG.2.7) demands that the average Shannon entropy is at least 0.997 per internal random bit. If the raw random numbers are binary-valued the entropy may be checked by the statistical test procedure B as described above. (Note that if the raw random numbers are Markovian, i.e. if there are no higher dependencies than 1-step dependencies, Step 1 and Step 2 of test procedure B indicate an entropy defect per raw random bit of less than 0.002. If the raw bits are iid the entropy defect per bit does not exceed 0.0002.) Note that the Min-entropy is the most generally-applicable entropy measure that can be used to estimate the guesswork in vulnerability analysis, but the min-entropy is difficult to quantify.
- 295 The developer may or may not assign additional test suites (i.e. the assignment may be empty) in the element FCS_RNG.1.2 clause (PTG.2.6). The element FCS_RNG.1.2 clause (PTG.2.6) demands that the application of test procedure A and - if assigned – of additional standard test suites does not reject the null hypothesis “the internal random numbers were generated by an ideal random number generator”. The same requirement is demanded for classes PTG.1 and DRG.1. The efforts of testing depend on the claimed resistance (in the ST) against attacks (cf. selected component of the family AVA_VAN). The evaluator may apply additional statistical tests as penetration tests. Note that this requirement does not necessarily imply that the rejection probability for the internal random numbers equals the rejection probability for sequences from ideal RNGs. Moreover, even this enhanced property is weaker than Requirements PTG.3.8, DRG.2.5, DRG.3.5 and DRG.4.7.

4.4.3. Further aspects

- 296 The developer *shall* provide evidence that the entropy of the internal random numbers is sufficiently large. The evidence comprises a stochastic model (cf. subsection 2.4.1 on pp. 39)

tailored to the TOE design and substantiated by statistical tests. There is only one level of detail in the description of the stochastic model, irrespective of the chosen EAL. Below this guidance describes explicitly two evaluation methods Method A and Method B, which may be applied to PTRNGs that generate 1-bit raw random numbers.

Method A

A.1 On the basis of the stochastic model, the developer shows that the raw random numbers are stationary distributed, and that there are no significant (long-step) dependencies, which are not covered by the statistical tests from test procedure B.

A.2 The raw random numbers pass the statistical test procedure B under all relevant environmental conditions.

A.3 The developer verifies that the post-processing algorithm does not reduce the entropy per bit. Alternatively, the developer provides evidence that the average entropy per internal random number remains sufficiently large.

A.4 The internal random numbers pass the statistical test procedure A (and other statistical standard test suites if applied) under all relevant environmental conditions.

Method B

B.1 On the basis of the stochastic model, the developer shows that the raw random numbers are stationary distributed, and that there are no significant (long-step) dependencies that are not covered by the statistical tests from test procedure B.

B.2 The developer verifies on the basis of the stochastic model that due to the post-processing algorithm the entropy per internal random number is sufficiently large. Under suitable conditions test procedure B might support this goal.

B.3 The internal random numbers pass the statistical test procedures A (and other statistical standard test suites if applied) and test procedures B under all relevant environmental conditions.

- 297 In Evaluation Method A, the raw random numbers pass the statistical test suite B under all relevant environmental conditions. In particular, the average entropy per raw random number is sufficiently large. Hence, the post-processing algorithm need not increase the entropy per bit. The identity mapping is allowed, which means ‘no post-processing’.
- 298 If Evaluation Method B is applied, three cases are possible: (i) test suite B does implicitly verify the entropy of the raw random numbers, and test suite B is passed; (ii) test suite B does implicitly verify the entropy of the raw random numbers, and test suite B fails; or (iii) test suite B cannot be applied or is not applicable to the raw random numbers, e.g. because there is no access to the raw random numbers or the raw random numbers are not binary-valued.
- 299 Evaluation Method A and Evaluation Method B consider the case that the PTRNG generates a single raw random bit per time unit. If the entropy source generates ($k \geq 1$) – bit raw random numbers, additional problems might occur (e.g., dependencies between the bits of each k-bit raw random number, different statistical behaviour of the particular bit traces) and thus must be considered. The evaluation may follow the line of Evaluation Method A or of Evaluation Method B described above. Depending on the concrete PTRNG design, this might require the specification of a new test suite B’, which shall be at least as effective as test suite B under the conditions of Evaluation Method A, or of Evaluation Method B, respectively. The effectiveness of the chosen test suite B’ shall be verified.
- 300 In the definition of the different evaluation methods, environmental conditions are viewed as relevant if they either (i) belong to the specified range of admissible working conditions, or (ii)

lie outside that range but cannot reliably be detected by anti-tamper measures (e.g., by sensors) although they may affect the behaviour of the entropy source.

- 301 Every statistical test considers only particular statistical properties. In particular, there are no generally-applicable blackbox tests that provide reliable entropy estimates for random numbers. To be recognized as effective for a positive verification of security properties, a statistical test must be based on a stochastic model.

4.5. Class PTG.3

- 302 The class PTG.3 defines requirements for RNGs that shall be appropriate for any cryptographic applications, in particular including those for PTG.2. Unlike PTG.2 - PTRNGs the security of PTG.3 - PTRNGs does not only rely on one security anchor but on two security anchors: information-theoretical security ensured on the physical part of the RNG *and* computational security ensured on the properties of the cryptographic post-processing algorithm. In particular, the internal random numbers will not show any bias nor short term dependencies.
- 303 PTG.3 is the strongest class that is defined in this document. PTG.3 conformant PTRNGs may be used for any cryptographic application. Typical PTG.3 applications are the generation of ephemeral keys for DSA signatures and for ECDSA signatures, for instance.
- 304 Class PTG.3 demands a post-processing algorithm with memory that (interpreted as a DRNG) is DRG.3-conformant (cf. chapter 4.8) even if its input data are known at some point in time. In particular, the state transition function ϕ and the extended output function ψ^* of this DRNG are cryptographic one-way functions.

4.5.1. Security functional requirements for the RNG class PTG.3

- 305 Functional security requirements of the class PTG.3 are defined by component FCS_RNG.1 with the specific operations given below.

FCS_RNG.1 Random number generation (Class PTG.3)

FCS_RNG.1.1 The TSF shall provide a *hybrid physical*²⁵ random number generator that implements:

- (PTG.3.1) *A total failure test detects a total failure of entropy source immediately when the RNG has started. When a total failure has been detected no random numbers will be output.*
- (PTG.3.2) *If a total failure of the entropy source occurs while the RNG is being operated, the RNG [selection: prevents the output of any internal random number that depends on some raw random numbers that have been generated after the total failure of the entropy source, generates the internal random numbers with a post-processing algorithm of class DRG.3 as long as its internal state entropy guarantees the claimed output entropy].*
- (PTG.3.3) *The online test shall detect non-tolerable statistical defects of the raw random number sequence (i) immediately when the RNG is started, and (ii)*

²⁵ [selection: *physical, non-physical true, deterministic, hybrid physical, hybrid deterministic*]

while the RNG is being operated. The TSF must not output any random numbers before the power-up online test and the seeding of the DRG.3 post-processing algorithm have been finished successfully or when a defect has been detected.

(PTG.3.4) The online test procedure shall be effective to detect non-tolerable weaknesses of the random numbers soon.

(PTG.3.5) The online test procedure checks the raw random number sequence. It is triggered [selection: externally, at regular intervals, continuously, upon specified internal events]. The online test is suitable for detecting non-tolerable statistical defects of the statistical properties of the raw random numbers within an acceptable period of time.

(PTG.3.6) The algorithmic post-processing algorithm belongs to Class DRG.3 with cryptographic state transition function and cryptographic output function, and the output data rate of the post-processing algorithm shall not exceed its input data rate.

FCS_RNG.1.2 The TSF shall provide [selection: bits, octets of bits, numbers [assignment: format of the numbers]] that meet:

(PTG.3.7) Statistical test suites cannot practically distinguish the internal random numbers from output sequences of an ideal RNG. The internal random numbers must pass test procedure A [assignment: additional test suites].

(PTG.3.8) The internal random numbers shall [selection: use PTRNG of class PTG.2 as random source for the post-processing, have [assignment: work factor], require [assignment: guess work]].

4.5.2. Application notes

306 The security capabilities in element FCS_RNG.1.1 clause (PTG.3.2) ensure the quality of the output in the time period between the occurrence and the detection of a total failure of the entropy source. The internal state of the post-processing algorithm shall ensure that the outputted internal random numbers contain sufficient entropy in this time period. Clause (PTG.3.6) ensures enhanced backward secrecy (cf. (DRG.3.3)) even if the entropy source has broken down and if the internal state is compromised.

307 Clauses (PTG.3.5) and (PTG.3.8) shall ensure that the quality of the internal random numbers is sufficiently large unless a noise alarm occurs.

308 The security capability (PTG.3.8) separates PTG.3-conformant PTRNGs from DRG.4-conformant DRNGs. Essentially, clauses (PTG.3.6) and (PTG.3.8) demand that the average entropy (over the time) of the input data of the algorithmic post-processing algorithm should not be smaller than the average number of internal random bits in the same time period; a small entropy defect might be tolerable. Since the bit length of the internal random numbers is usually much larger than the bit size of the input data of the post-processing algorithm, this requirement might not be fulfilled in short time intervals. However, the entropy of the internal state shall compensate such time-local effects for any time interval, i.e., the entropy of the input data shall not be smaller than the number of internal random bits minus the bit length of the internal state

- of the algorithmic post-processing algorithm. The security capabilities (PTG.3.3) and (PTG.3.5) ensure that the internal random numbers contain enough entropy while the PTRNG is in operation.
- 309 A PTG.3-conformant PTRNG may be viewed as a composition of an “inner” PTRNG with a DRNG post-processing where the output data of the PTRNG serve as input data for the DRNG, which updates / refreshes its internal state. The “inner” PTRNG itself may comprise an ‘inner’ algorithmic post-processing algorithm. In particular, the output data of the inner PTRNG need not necessarily be raw random numbers but may already be algorithmically post-processed.
- 310 If we view a PTG.3-conformant PTRNG as a composition of an inner ‘PTRNG’ part and a DRNG part, we may distinguish two cases: (i) the inner PTRNG is PTG.2-conformant, or (ii) the inner PTRNG is not PTG.2-conformant. For case (i), the RNG may principally be operated in three modes: (a) as a PTG.3-PTRNG, (b) as a PTG.2-PTRNG if the output data of the physical part are used directly, or (c) as a DRG.4-DRNG if the input sequence of the algorithmic post-processing algorithm is ‘extended’. For case (ii), only options (a) and (c) remain. Security requirements and functional requirements of particular cryptographic applications might make such a diversification meaningful. Conformity to the particular classes must be verified in separate evaluation processes. Evaluation results clearly may be used for all these evaluations.
- 311 The post-processing algorithm belongs to class DRG.3 even if the PTRNG random source has totally broken down and an attacker knows or is able to guess its output (i.e., the input of the post-processing algorithm). Of course, this demands that the internal state of the post-processing algorithm was unpredictable at the time of the breakdown, which is the case if the PTRNG had worked properly for at least a short time. In the case of a total breakdown, of course, the PTRNG must not output more internal random bits than the size of the internal state in bits. Otherwise, the RNG belongs to class DRG.3 after this instant.
- 312 Unlike for PTG.3-conformant PTRNGs, DRG.4-conformant DRNGs may ‘extend’ the input data, i.e., DRG.4 conformant DRNGs may compute large output sequences from short input sequences. In particular, there is no minimum entropy bound per internal random bit (cf. chapter 4.6 for details).
- 313 The element FCS_RNG.1.2 clause (PTG.3.7) requires that statistical tests cannot practically distinguish the internal random numbers from output sequences of an ideal RNG. The feature that statistical test suites cannot **practically** distinguish the RNG output from uniformly distributed random bit sequences depends on the claimed in the ST resistance against attacks (cf. selected component of the family AVA_VAN). The effort of testing is defined by the used test suites and the amount of test data. The developer will provide functional tests with test suite A and maybe other appropriate tests suites. The evaluator may additionally apply further statistical tests as penetration tests. These tests may be tailored to the RNG design. Requirement (PTG.3.7) is stronger than (PTG.1.5) and (PTG.2.6)
- 314 The clause (PTG.3.8) provides three methods describing the quality of the output. The work factor and the guess work may be used directly in the vulnerability analysis of the application using the random number output. The selection “use PTRNG of class PTG.2 as random source” together with (PTG.3.6) allows an indirect verification of the output quality.
- 315 If the DRNG post-processing algorithm maps the input data from the inner PTRNG bijectively onto the output space, its entropy remains constant. If the inner PTRNG is PTG.2-conformant,

the (Shannon) entropy per random bit is sufficiently large. Since PTG.2-conformant PTRNGs generate stationary random raw sequences, the Shannon entropy provides an appropriate estimate for the work factor unless the sequences are too short, which is not the case for DRG.3-conformant DRNGs. An example of a bijective DRG.3-conformant post-processing algorithm is a block cipher that is operated in OFB mode where the internal random number is given by the whole ciphertext block. Before a new internal random number is output, a fresh random bit string from the inner PTRNG is XORed to that part of the inner state of the DRNG that stores the previous internal random number (= last ciphertext). Then the updated part of the internal state is encrypted and output (internal random number), and then a one-way function is applied to the updated internal state.

- 316 DRG.3-conformity requires a one-way state transition function. One-way functions usually reduce the entropy per bit unless the length of the input data significantly exceeds the length of the output. One usually models one-way functions as realisations of random mappings. Section 5.4.4 investigates the effect of random mappings on uniformly distributed input data. Output sequences of PTG.2-conformant real-world PTRNGs are not ideal, but should be close enough to the uniform distribution so that it appears reasonable to assume that the figures from section 5.4.4 are also valid for input data from PTG.2-conformant PTRNGs (see section 5.4.4 for further details). An example of this type of post-processing algorithm is the following: Whenever an internal random number shall be output, a fresh n -bit string from the inner PTRNG is XORed to the internal state. Then an m -bit hash value ($m < n$) of the internal state is output, and the internal state is updated by applying (another) one-way function.
- 317 If the tot test and / or the online test are not part of the TOE but to be implemented later as an external security measure the applicant must submit a specification of the test(s), a justification for effectiveness and a reference implementation. The suitability of the tot test and the online test shall be verified based on the reference implementation. In the positive case the RNG is said to be PTG.3 compliant under the condition that the final implementation meets the specification in the user manual (to be checked in a composite evaluation).
- 318 Under certain conditions class PTG.3 allows a composite evaluation. For example, a software developer might use the output of a PTG.2 RNG, which is implemented in hardware on the device, as input for a DRG.3 RNG with memory. In the composite evaluation it has to be checked whether all requirements that concern the post-processing algorithm itself and its interaction with the PTG.2 output are fulfilled. If this is the case the composite RNG (PTG.2 + DRG.3 conformant post-processing) is PTG.3 conformant.

4.5.3. Further aspects

- 319 The developer *shall* provide the evidence required in ATE_FUN.{1,2}.PTG.3.2 clause (PTG.3.8), i.e., the developer shall provide evidence that the entropy of the internal random numbers is sufficiently large. The evidence comprises a stochastic model (cf. subsection 2.4.1 on pp. 39) tailored to the TOE design and substantiated by statistical tests. There is only one level of detail in the description of the stochastic model, irrespective of the chosen EAL. The stochastic model shall consider the situation before the application of the DRG.3 post-processing algorithm, i.e., the input data of the post-processing algorithm (as for PTG.2-conformant PTRNGs) and the effect of this post-processing.
- 320 For the non-post-processed data, this guidance describes Method A* and Method B*, which may be applied if 1-bit raw random numbers are generated. These evaluation methods are related to Method A and Method B for PTG.2-PRNGs.

321 **Method A***

A.1 On the basis of the stochastic model the developer shows that the raw random numbers are stationary distributed, and that there are no significant (long-step) dependencies, which are not covered by the statistical tests from test suite B.

A.2 The raw random numbers pass the statistical test suite B under all relevant environmental conditions.

A.3 The developer verifies that the inner post-processing algorithm (if one exists, resp. if it is different from the identity mapping) does not reduce the entropy per bit. Alternatively, the developer provides evidence that the average entropy per internal random number remains sufficiently large.

A.4 The internal random numbers pass the statistical test procedure A (and other statistical standard test suites if applied) under all relevant environmental conditions.

322 **Method B***

B.1 On the basis of the stochastic model the developer shows that the raw random numbers are stationary distributed, and that there are no significant (long-step) dependencies that are not covered by the statistical tests from test suite B.

B.2 The developer verifies on the basis of the stochastic model and the inner post-processing algorithm (if it exists, resp. if it is different from the identity mapping) that the average entropy per input bit of the DRG.3-post-processing exceeds a certain entropy bound (to be specified).

B.3 The internal random numbers pass the statistical test procedure A (and other statistical standard test suites if applied) under all relevant environmental conditions.

323 Method A* guarantees average of 0.997 bit Shannon entropy per input bit (i.e., 7,976 bit Shannon entropy per input octet) of the post-processing algorithm. To estimate the average entropy per output bit, one may apply results on random mappings or random permutations.

324 Evaluation Method A* and Evaluation Method B* require that the PTRNG generates a single random raw bit per time unit. If the entropy source generates k-bit raw random numbers ($k > 1$), additional problems (e.g., dependencies between the bits of each k-bit raw random numbers, different statistical behaviour of the particular bit traces) might occur and thus must be considered. The evaluation may follow the line of Evaluation Method A* or of Evaluation B* described above. Depending on the concrete PTRNG design, this might require the specification of a new test procedure B', which shall be at least as effective as test procedure B under the conditions of Evaluation Method A*, or of Evaluation Method B*, respectively. The effectiveness of the chosen test procedure B' shall be explained.

325 In the definition of the different evaluation methods, environmental conditions are viewed as relevant if they either (i) belong to the specified range of admissible working conditions, or (ii) lie outside that range but cannot reliably be detected by anti-tamper measures (e.g., by sensors) although they may affect the behaviour of the entropy source.

326 Each statistical test considers only particular statistical properties. In particular, there are no generally-applicable blackbox tests that provide reliable entropy estimates for random numbers. To be recognized as effective for a positive verification of security properties, a statistical test must be based on a stochastic model.

327 Statistical tests, which estimate the entropy of a random sequence, (tacitly) assume that the sequence has specific properties. The rationale behind the evaluation methods A* and B* is that

statistical tests cannot effectively be applied to the internal random numbers because the DRG.3 post-processing algorithm causes complicated dependencies within the internal random number sequence.

4.6. Class DRG.1

328 The class DRG.1 defines requirements for deterministic RNGs. It shall not be possible to distinguish the generated random numbers from output sequences from an ideal RNG by simple statistical blackbox tests. DRG.1 conformant DRNGs provide forward secrecy.

329 An RNG of class DRG.1 might be used for applications that need fresh data that are distinct from previously-generated data with high probability, e.g., to generate challenges in cryptographic protocols or initialization vectors for block ciphers in special modes of operation, provided that previous random numbers need not be protected. DRG.1-conformant DRNGs may be used for zero-knowledge proofs (cf. par. 265).

4.6.1. Security functional requirements for the RNG class DRG.1

330 Functional security requirements of class DRG.1 are defined by component FCS_RNG.1 with specific operations as given below.

FCS_RNG.1 Random number generation (Class DRG.1)

FCS_RNG.1.1 The TSF shall provide a *deterministic*²⁶ random number generator that implements:

(DRG.1.1) *If initialized with a random seed [selection: using a PTRNG of class PTG.2 as random source, using a PTRNG of class PTG.3 as random source, using an NPTRNG of class NTG.1 [assignment: other requirements for seeding]], the internal state of the RNG shall [selection: have [assignment: amount of entropy], have [assignment: work factor], require [assignment: guess work]].*

(DRG.1.2) *The RNG provides forward secrecy.*²⁷

FCS_RNG.1.2 The TSF shall provide random numbers that meet:

(DRG.1.3) *The RNG, initialized with a random seed [assignment: requirements for seeding], generates output for which [assignment: number of strings] strings of bit length 128 are mutually different with probability [assignment: probability].*

(DRG.1.4) *Test procedure A [assignment: additional standard test suites] does not practically distinguish the random numbers from output sequences of ideal RNGs.*²⁸

4.6.2. Application notes

²⁶ [selection: physical, non-physical true, deterministic, hybrid physical, hybrid deterministic]

²⁷ [assignment: list of security capabilities]

²⁸ [assignment: a defined quality metric]

- 331 The vulnerability analysis shall show that an attacker is not able to guess the internal state of the DRNG with the attack potential that is claimed in the ST. The value assigned in clause (DRG.1.1) *shall* meet the attack potential identified in the vulnerability analysis component. The entropy of the initial internal state is an upper bound of the entropy of the generated random number sequence. The entropy of the internal state may decrease over the lifetime of the DRNG instantiation. The internal state of the DRNG *shall* contain sufficient entropy to prevent successful guessing attacks within the lifetime of the DRNG instantiation. Table 12 gives lower entropy bounds for the internal state. Its bit length must at least equal the minimal entropy bound.
- 332 It is natural to generate the seed of a DRNG with a PTG.2- or PTG.3-conformant PTRNG. If the internal state of the DRNG is initialized in this way, and if the internal state is at least 25% larger (in bits) than the Min-entropy bounds given in Table 12, an explicit assessment of the Min-entropy is not necessary. This is justified by the fact that PTG.2-conformant PTRNGs generate stationary output sequences with large Shannon entropy, which ensures a large work factor. Similarly, PTG.3-conformant PTRNGs also generate high-entropy random numbers (see also the application notes for class PTG.3). We point out that tighter entropy bounds for PTG.2 and PTG.3 than the generic 25% margin (allowing smaller internal states) should be possible in most cases but require justification (cf. stochastic model).

Table 12: Attack potential, Min-entropy, and recommended length of the internal state

Component of the vulnerability analysis			Required min-entropy of the internal state	Recommended length of the internal state
CC version 2.3		CC version 3.1		
		AVA_VAN.1, 2 (basic)	≥ 40 bit	≥ 80 bit
AVA_SOF.1, low	AVA_VLA.2 (low)	AVA_VAN.3 (enhanced basic)	≥ 48 bit	≥ 96 bit
AVA_SOF.1, medium	AVA_VLA.3 (moderate)	AVA_VAN.4 (moderate)	≥ 64 bit	≥ 128 bit
AVA_SOF.1, high	AVA_VLA.4 (high)	AVA_VAN.5 (high)	≥ 100 bit	≥ 200 bit

- 333 The (Min-)entropy of the internal state clearly depends on the initialization procedure with a random seed, but also on the state transition function φ and possibly publicly known input. If the state transition function is bijective (i.e., $S = \varphi(S, i)$, for each publicly known input i), it maintains the entropy of the internal state. If the state transition function behaves like a randomly-selected mapping, it will reduce the number of possible internal states for some observed public input data i (i.e. $|\varphi(S, i)| < |S|$), thus reducing the entropy of the internal state (cf. [FI0d89] for details about statistics of random mappings). The publicly-known input does not increase the overall entropy of the system, but it might influence the process of entropy reduction of the internal state over time and make attacks more difficult.
- 334 The security capability (DRG.1.2) forward secrecy means the following: subsequent (future) values cannot be determined or guessed with non-negligible probability from current or

previous output values [ISO18031]. In particular, the design of the DRNG shall prevent a successful guess the internal state, allowing the calculation of future output. The forward secrecy capability requires that the internal state has sufficient entropy to prevent guessing as well as the confidentiality of the current internal state is protected by the design of the DRNG (e.g., one-way extended output function) and the security architecture of the TSF (cf. self-protection accessed in ADV_ARC).

- 335 The security capability (DRG.1.3) requires the DRNG to generate mutually different pseudo-random numbers. The first assignment in the element FCS_RNG.1.2 describes the requirements on the seeding procedure (e.g., the entropy of the seed, how many random numbers can be generated between two seeding procedures). Under this assumption, clause (DRG.1.3) defines the capability of the DRNG to generate an assigned number (let's say k) of fresh random strings with given length of 128 bit being mutually different with at least the defined probability (let's say $1 - \varepsilon$, i.e. ε is the probability of at least one coincidence). If the DRNG generates shorter output values (random numbers) several consecutive output values are concatenated and, if necessary, these joint random bit strings are cut off after 128 bits. The selection of the parameters k and ε depends on the intended application of the DRNG described in the guidance documentation:

- the requirements for seeding shall fit to the intended use cases, and
- during the lifetime of the DRNG instantiation (i.e., during the time between two seeding processes) the DRNG must not produce more random bits than the product of the assigned number k of output strings by their bit length.

The assigned number of strings, string length in bits, and probability shall allow to provide evidence demonstrating that this requirement is fulfilled. The parameters assigned in the element FCS_RNG.1.1 shall meet the attack potential identified in the vulnerability analysis component.

- 336 Table 13 provides necessary conditions to resist attacks, which are based on the repetition of random strings generated by the RNG and are exploitable in the intended environment with the identified attack potential. The TOE might be vulnerable if the RNG is used for purposes that require other properties of the RNG. In this case, the developer shall consider an RNG with additional security features, like class DRG.2 and higher.

Table 13: Requirements for the parameters in (DRG.1.3) depending on claimed attack potential

Component of the vulnerability analysis			Parameter k denotes the number of output strings that shall be mutually different with probability $\geq 1 - \varepsilon$
CC version 2.3		CC version 3.1	
		AVA_VAN.{1, 2} (basic)	$k > 2^{14}$ and $\varepsilon < 2^{-8}$
AVA_SOF.1, low	AVA_VLA.2 (low)	AVA_VAN.3 (enhanced basic)	$k > 2^{19}$ and $\varepsilon < 2^{-10}$
AVA_SOF.1, medium	AVA_VLA.3 (moderate)	AVA_VAN.4 (moderate)	$k > 2^{26}$ and $\varepsilon < 2^{-12}$
AVA_SOF.1, high	AVA_VLA.4 (high)	AVA_VAN.5 (high)	$k > 2^{34}$ and $\varepsilon < 2^{-16}$

- 337 For an ideal RNG the probability for at least one collision within the first k 128 bit output strings is approximately $1 - \exp(-k^2 / 2^{129})$. The expected number of bit strings until the first collision can be approximated by $\sqrt{\pi} 2^{63.5}$ (cf. formulae (30) and (31)).
- 338 The developer may or may not assign additional standard test suites (i.e. the assignment is empty) in the element FCS_RNG.1.2 clause (DRG.1.4). The effort of testing to demonstrate that test procedure A and the assigned test suites do not practically distinguish the RNG output from uniformly distributed random bit sequences depends on the resistance against attacks as claimed in the ST (cf. selected component of the family AVA_VAN). The quality metric (DRG.1.4) is different from (PTG.1.5):
- the class PTG.1 generates **true random** numbers, which cannot be distinguished from ideal random numbers by tests with the test procedure A and - if assigned in clause (PTG.1.5) – with the additional standard test suites,
 - the DRG.1 generates **deterministic random** numbers, but they cannot be distinguished from ideal random numbers by tests with the test procedure A and - if assigned in clause (DRG.1.4) - with the additional standard test suites.

The DRNG gets its initial random state from a randomly selected seed. The most straightforward methods are to use the seed as the initial internal state or to apply the state transition function to the seed. The output string from the initial internal state is part of the deterministically-generated output sequence. The entropy of the output string (and, therefore, of the random numbers generated) cannot be greater than the entropy of the seed. Depending on the seeding procedure of the DRNG, the tests are applied to one or more output strings.

4.6.3. Further aspects

- 339 In many cases it may be practically infeasible to specify the distribution p_A of the first internal state. It suffices to specify a set of distributions that contains p_A if all elements of this set fulfil

the requirements of class DRG.1. Example: The seed entropy exceeds a certain lower entropy bound, e.g. because the seed has been generated with an RNG that is conformant to class PTG.2, PTG.3 or NTG.1. The security architecture description shall describe the secure initialization process of the RNG.

4.7. Class DRG.2

340 The class DRG.2 defines requirements for deterministic RNGs. It shall not be possible to distinguish the generated random numbers from output sequences from an ideal RNG by statistical tests, and the generated random numbers sequence shall have at least some minimum amount of Min-entropy (contained in the seed), and backward secrecy is ensured. The class DRG.2 includes the properties of class DRG.1.

341 RNGs of class DRG.2 may be used for the generation of cryptographic keys and parameters, pseudo-random padding bits, etc. (cf. par. 265). The TSF protects the internal state of the RNG from being compromised.

4.7.1. Security functional requirements for the RNG class DRG.2

342 Functional security requirements of the class DRG.2 are defined by component FCS_RNG.1 with specific operations as given below.

FCS_RNG.1 Random number generation (Class DRG.2)

FCS_RNG.1.1 The TSF shall provide a *deterministic*²⁹ random number generator that implements:

(DRG.2.1) *If initialized with a random seed [selection: using a PTRNG of class PTG.2 as random source, using a PTRNG of class PTG.3 as random source, using an NPTRNG of class NTG.1 [assignment: other requirements for seeding]], the internal state of the RNG shall [selection: have [assignment: amount of entropy], have [assignment: work factor], require [assignment: guess work]].*

(DRG.2.2) *The RNG provides forward secrecy.*

(DRG.2.3) *The RNG provides backward secrecy.*³⁰

FCS_RNG.1.2 The TSF shall provide random numbers that meet:

(DRG.2.4) *The RNG, initialized with a random seed [assignment: requirements for seeding], generates output for which [assignment: number of strings] strings of bit length 128 are mutually different with probability [assignment: probability].*

(DRG.2.5) *Statistical test suites cannot practically distinguish the random numbers from output sequences of an ideal RNG. The random numbers must pass test procedure A [assignment: additional test suites].*³¹

²⁹ [selection: physical, non-physical true, deterministic, hybrid physical, hybrid deterministic]

³⁰ [assignment: list of security capabilities]

4.7.2. Application notes

- 343 Class DRG.2 includes the requirements of class DRG.1 for the security capability (DRG1.1) and the quality of the random numbers, (DRG.1.2) and (DRG.1.3). The application notes for class DRG.1 are valid for class DRG.2, as well. The (DRG.2.4) parameters must meet the conditions in Table 13.
- 344 The class DRG.2 requires assigned quality of internal state in (DRG.2.1). The clauses (DRG.2.2) and (DRG.2.3) require forward and backward secrecy according to [ISO18031], i.e., that unknown previous or future output values cannot be determined from known output values (random numbers). The developer should select a cryptographic function for the one-way extended output function and should consider (but is not requested to choose) a cryptographic one-way function for the state transition function, as well. The whole internal state is viewed as input, including cryptographic keys. Hence, keyed bijections, typically coming from strong block ciphers, also count as one-way functions (cf. section 5.3 for details).
- 345 If the DRNG is intended for the generation of cryptographic keys, the entropy in the element FCS_RNG.1.1 clause (DRG.2.1), should meet the security level of the cryptographic algorithm. If no cryptographic weaknesses are known, the security level of a symmetric cryptographic algorithm is assumed to be equal to its key length. For example, if the assignment in the element FCS_RNG.1.1 clause (DRG.2.1) assigns ≥ 128 bit Min-entropy to its internal state, the DRNG may be used to generate AES-128 bits. If the internal state contains less entropy, the AES key generation by means of such a DRNG might be viewed as a potential vulnerability of the cryptosystem.
- 346 The element FCS_RNG.1.2, clause (DRG.2.5), requires that statistical tests cannot **practically** distinguish the random numbers from output sequences of an ideal RNG. The effort of testing in order to demonstrate that the statistical test suites cannot practically distinguish the RNG output from uniformly distributed random bit sequences depends on the resistance against attacks as claimed in the ST (cf. selected component of the family AVA_VAN). The effort of testing is defined by the used test suites and the amount of test data. The developer shall provide functional tests with test procedure A and maybe other appropriate tests suites or specific tests that are tailored to the particular DRNG. The evaluator may provide additional statistical test as penetration tests. Of course, clause (DRG.2.5) excludes ‘unfair’ tests that exploit the knowledge of the given internal state.
- 347 Note that the one-way property of the output function is a necessary condition for forward secrecy, but not a sufficient condition for good statistical properties of the DRNG output. For example, if the DRNG outputs the hash value of the internal state, the output is expected to be indistinguishable from the output of an ideal RNG. If the DRNG output function concatenates statistically weak strings (e.g., a sequence number of the output) to this hash values, this might no longer be true.

4.7.3. Further aspects

- 348 In many cases it may be practically infeasible to specify the distribution p_A of the first internal state. It suffices to specify a set of distributions that contains p_A if all elements of this set fulfil the requirements of class DRG.2. Example: The seed entropy exceeds a certain lower entropy bound, e.g. because the seed has been generated with an RNG that is conformant to class

³¹ [assignment: *a defined quality metric*]

PTG.2, PTG.3 or NTG.1. The security architecture description shall describe the secure initialization process of the RNG.

4.8. Class DRG.3

349 The class DRG.3 defines requirements for deterministic RNGs. It shall not be possible to distinguish the generated random numbers from output sequences from an ideal RNG by statistical tests, and the generated random numbers sequence shall have at least some minimum amount of Min-entropy (contained in the seed), and enhanced backward secrecy is ensured. The class DRG.3 includes the requirements of class DRG.2.

350 RNGs of class DRG.3 might be used for the generation of cryptographic keys and parameters, pseudo-random padding bits, etc. (cf. par. 265). Any compromise of the internal state of the DRNG shall be detected, and re-seeding shall be enforced before further use of the RNG.

4.8.1. Security functional requirements for the RNG class DRG.3

351 Functional security requirements of the class DRG.3 are defined by component FCS_RNG.1 with specific operations as given below.

FCS_RNG.1 Random number generation (Class DRG.3)

FCS_RNG.1.1 The TSF shall provide a *deterministic*³² random number generator that implements:

(DRG.3.1) *If initialized with a random seed [selection: using a PTRNG of class PTG.2 as random source, using a PTRNG of class PTG.3 as random source, using an NPTRNG of class NTG.1 [assignment: other requirements for seeding]], the internal state of the RNG shall [selection: have [assignment: amount of entropy], have [assignment: work factor], require [assignment: guess work]].*

(DRG.3.2) *The RNG provides forward secrecy.*

(DRG.3.3) *The RNG provides backward secrecy even if the current internal state is known.*³³

FCS_RNG.1.2 The TSF shall provide random numbers that meet:

(DRG.3.4) *The RNG, initialized with a random seed [assignment: requirements for seeding], generates output for which [assignment: number of strings] strings of bit length 128 are mutually different with probability [assignment: probability].*

(DRG.3.5) *Statistical test suites cannot practically distinguish the random numbers from output sequences of an ideal RNG. The random numbers must pass test procedure A [assignment: additional test suites].*³⁴

³² [selection: physical, non-physical true, deterministic, hybrid physical, hybrid deterministic]

³³ [assignment: list of security capabilities]

4.8.2. Application notes

- 352 The class DRG.3 includes the requirements of class DRG.2 for the security capabilities (DRG.2.1), (DRG.2.2), and (DRG.2.3), and the quality metrics in (DRG.2.4) and (DRG.2.5). The application notes for class DRG.2 are valid for the class DRG.3 as well. It adds requirements for security capabilities referring to enhanced backward secrecy in (DRG.3.3).
- 353 While (DRG.2.2) and (DRG.2.3) require forward and backward secrecy (i.e., unknown output value cannot be determined from known output values), the security capabilities (DRG.3.2) and (DRG.3.3) additionally require enhanced backward secrecy. This means that previous output values cannot even be determined with knowledge of the current internal state and current and future output values. Enhanced backward secrecy might be relevant, for instance, for software implementations of a DRNG when the internal state has been compromised while all random numbers generated in the past shall remain secret (e.g., cryptographic keys).
- 354 Clause (DRG.3.3) essentially requires a cryptographic state transition function. DRG.3-conformant designs with non-cryptographic output functions may exist. However, it is recommended to apply a cryptographic output function.

4.8.3. Further aspects

- 355 In many cases it may be practically infeasible to specify the distribution p_A of the first internal state. It suffices to specify a set of distributions that contains p_A if all elements of this set fulfil the requirements of class DRG.3. Example: The seed entropy exceeds a certain lower entropy bound, e.g. because the seed has been generated with an RNG that is conformant to class PTG.2, PTG.3 or NTG.1. The security architecture description shall describe the secure initialization process of the RNG.

4.9. Class DRG.4

- 356 Class DRG.4 defines requirements for hybrid deterministic RNGs that primarily rely on the security imposed by computational-complexity, which is ‘enhanced’ by additional entropy from a physical true RNG. RNGs of class DRG.4 clearly may be used for the same cryptographic applications as DRG.3-conformant DRNGs, and additionally for applications that require enhanced forward secrecy.
- 357 Class DRG.4 is based on class DRG.3 but may not use an external source of randomness for the seeding process. RNGs of class DRG.4 contain an internal source of randomness for seeding and reseeding, resp. seed-update (to ensure forward secrecy).

4.9.1. Security functional requirements for the RNG class DRG.4

- 358 Functional security requirements of the class DRG.4 are defined by component FCS_RNG.1 with specific operations as given below.

FCS_RNG.1 Random number generation (Class DRG.4)

³⁴ [assignment: *a defined quality metric*]

FCS_RNG.1.1 The TSF shall provide a *hybrid deterministic*³⁵ random number generator that implements:

(DRG.4.1) *The internal state of the RNG shall [selection: use PTRNG of class PTG.2 as random source, have [assignment: work factor], require [assignment: guess work]].*

(DRG.4.2) *The RNG provides forward secrecy.*

(DRG.4.3) *The RNG provides backward secrecy even if the current internal state is known.*

(DRG.4.4) *The RNG provides enhanced forward secrecy [selection: on demand, on condition [assignment: condition], after [assignment: time]].*

(DRG.4.5) *The internal state of the RNG is seeded by an [selection: internal entropy source, PTRNG of class PTG.2, PTRNG of class PTG.3, [other selection]].*³⁶

FCS_RNG.1.2 The TSF shall provide random numbers that meet:

(DRG.4.6) *The RNG generates output for which [assignment: number of strings] strings of bit length 128 are mutually different with probability [assignment: probability].*

(DRG.4.7) *Statistical test suites cannot practically distinguish the random numbers from output sequences of an ideal RNG. The random numbers must pass test procedure A [assignment: additional test suites].*³⁷

4.9.2. Application notes

359 Class DRG.4 includes the requirements of class DRG.3 for the security capabilities (DRG.3.1) and (DRG.3.3) and the quality metrics of (DRG.3.4) and (DRG.3.5). The assignment in clause (DRG.4.1) should meet the conditions presented in Table 13 (cf. also the application notes for DRG.2). The assignment in clause (DRG.4.6) should meet the conditions of Table 13. (DRG.4.1) and (DRG.4.6) do not depend on an external entropy source because the RNG is seeded by the internal random source identified in (DRG.4.5). Under this consideration, the application notes for class DRG.3 are applicable for the class DRG.4.

360 DRG.4 includes the forward secrecy according to [ISO18031], i.e., subsequent (future) values cannot be determined from current or previous output values, and adds requirements for enhanced forward secrecy (DRG.4.4), i.e., after the identified event or time, the subsequent (future) output values cannot be determined from current or previous output values, even if the current internal state is compromised.

361 The selection in FCS_RNG.1.1 clause (DRG.4.4) depends on the implementation of the reseeding process, resp. of seed update process. The TOE may provide forward secrecy on demand, e.g., if the RNG is used for the generation of sensitive cryptographic keys like a

³⁵ [selection: *physical, non-physical true, deterministic, hybrid physical, hybrid deterministic*]

³⁶ [assignment: *list of security capabilities*]

³⁷ [assignment: *a defined quality metric*]

signature-creation key in a smart card. The TOE may provide forward secrecy on condition or after time, e.g., if the RNG gets continuously fresh entropy from the internal entropy source. The assignments shall consider the seeding procedure and the entropy, which is provided by the internal physical true RNG.

- 362 The security capability of forward secrecy in (DRG.4.4) requires fresh entropy that is provided by the internal source of randomness for reseeding or seed-updating the internal state. If “*internal entropy source*” is selected in clause (DRG.4.5), the RNG shall implement mechanisms (entropy estimator) to ensure that the internal entropy source has provided sufficient entropy to ensure forward secrecy. A combination of an online test of the internal entropy source and the condition selected in (DRG.4.4) may ensure a (suitably large) lower entropy bound. If “*physical true RNG of class PTG.2*” or “*physical true RNG of class PTG.3*” is selected, these tests are required as security capabilities of the PTG class (cf. to class definition above). Note that the selection in clauses (PTG.4.1) and (PTG.4.5) shall be consistent if an internal entropy source is used for seeding.
- 363 The (first) seeding of the internal state might be done within a personalization process with an external entropy source. This ensures that the internal state is unknown from the beginning even if forward secrecy is assured only on demand and if the first application does not apply for forward secrecy.

4.9.3. Further aspects

- 364 In many cases it may be practically infeasible to specify the distribution p_A of the first internal state. It suffices to specify a set of distributions that contains p_A if all elements of this set fulfil the requirements of class DRG.4. Example: The seed entropy exceeds a certain lower entropy bound, e.g. because the seed has been generated with an RNG that is conformant to class PTG.2 or PTG.3. The security architecture description shall describe the secure initialization process of the RNG.
- 365 The security architecture must protect the internal state of a DRNG as one aspect of self-protection. If the internal state has been compromised backward secrecy DRG.4.3 ensures the secrecy of all previous random numbers while enhanced forward secrecy DRG.4.4 ensures the secrecy of the random numbers that will be generated after the next reseeding, resp. the next seed update. However, if an attacker knows the current internal state he may calculate all output values that are generated before the next reseeding, resp. the next seed update.

4.10. Class NTG.1

- 366 The class NTG.1 defines requirements for non-physical true RNGs that rely on information-theoretical security (similar as physical RNGs) but use external input signals as entropy source. Additionally, a suitable cryptographic post-processing algorithm shall provide a second security anchor.

4.10.1. Security functional requirements for the NPTRNG class NTG.1

- 367 Functional security requirements of the class NTG.1 are defined by the component FCS_RNG.1 with specific operations as given below.

FCS_RNG.1 Random number generation (Class NTG.1)

FCS_RNG.1.1 The TSF shall provide a *non-physical true*³⁸ random number generator that implements:

(NTG.1.1) *The RNG shall test the external input data provided by a non-physical entropy source in order to estimate the entropy and to detect non-tolerable statistical defects under the condition [assignment: requirements for NPT RNG operation].*

(NTG.1.2) *The internal state of the RNG shall have at least [assignment: Min-entropy]. The RNG shall prevent any output of random numbers until the conditions for seeding are fulfilled.*

(NTG.1.3) *The RNG provides backward secrecy even if the current internal state and the previously used data for reseeding, resp. for seed-update, are known.*³⁹

FCS_RNG.1.2 The TSF shall provide random numbers that meet:

(NTG.1.4) *The RNG generates output for which [assignment: number of strings] strings of bit length 128 are mutually different with probability [assignment: probability].*

(NTG.1.5) *Statistical test suites cannot practically distinguish the internal random numbers from output sequences of an ideal RNG. The internal random numbers must pass test procedure A [assignment: additional test suites].*

(NTG.1.6) *The average Shannon entropy per internal random bit exceeds 0.997.*⁴⁰

4.10.2. Application notes

368 A non-physical true RNG comprises three parts:

- the input pre-computation block, which computes the input for the internal DRNG from several external input signals provided by (usually several) entropy sources,
- the entropy pool, which collects entropy and computes the output,
- the control block, which prevents the output of random numbers until the RNG has sufficient entropy to ensure the randomness of the output.

369 The class NTG.1 combines security capabilities of deterministic RNGs and security capabilities similar to those of physical true RNGs. By clause (NTG.1.3) the entropy pool with its updating mechanism and output function (viewed as a DRNG) is DRG.3-conformant.

370 The security capability (NTG.1.1) checks the external input signals from the entropy sources with regard to total failure and non-tolerable weaknesses. Usually, an ‘entropy counter’ (applying heuristic rules) is kept to provide plausibility that enough fresh entropy is mixed up with the current internal state. The entropy counter reduces the (estimated) entropy of the internal state by m whenever m bits are output. If the value of the entropy counter is smaller

³⁸ [selection: *physical, non-physical true, deterministic, hybrid physical, hybrid deterministic*]

³⁹ [assignment: *list of security capabilities*]

⁴⁰ [assignment: *a defined quality metric*]

than m the output of an m -bit string is prohibited. One says the input is “rated for entropy estimation”.

- 371 Online tests for NPTRNGs, however, will usually be very different from online tests for classes PTG.2 or PTG.3 because locally the input data for NPTRNGs may provide only low entropy; they might be biased or strongly dependent. It is usually impossible to formulate a precise stochastic model for input data of NPTRNGs.
- 372 The security capability (NTG.1.2) is the same as (DRG.2.4). The security capability (NTG.1.3) of enhanced backward secrecy is the same as (DRG.3.4). The class NTG.1 includes the requirements for the quality of the random numbers (DRG.1.3), (DRG.2.5), and (PTG.2.7).

5. Examples

373 This chapter discusses several examples that shall illustrate the theoretical concepts, which have been explained in the previous chapters.

5.1. Guesswork for binomial distributed data

374 In chapter 2.3.2 we introduced the concept of guess work. The next example shows how to calculate the guess work for a specific scenario..

Example 1: Guess work for vectors with independent biased bits

375 Assume that the attacker guesses realizations of binary vectors $\bar{X} = (X_1, X_2, \dots, X_n)$ with independent components X_i , $P\{X_i = 1\} = p$, $P\{X_i = 0\} = 1 - p$, $i = 1, 2, \dots, n$. Then

$$P\{\bar{X} = \bar{b}\} = p^{h(\bar{b})} \cdot (1 - p)^{n-h(\bar{b})} \quad (17)$$

with $\bar{b} = (b_1, b_2, \dots, b_n) \in \{0, 1\}^n$, while $h(\bar{b})$ denotes the HAMMING weight of \bar{b} . For $p > 0.5$, probability (17) increases with the HAMMING weight. One clearly begins with the guess $\bar{1}$ consisting of n ones, then one checks all vectors with HAMMING weight n-1, all vectors with HAMMING weight n-2, etc. until the searched vector has been found. In the least favourable case, we must check $N, N = 2^n$, vectors.

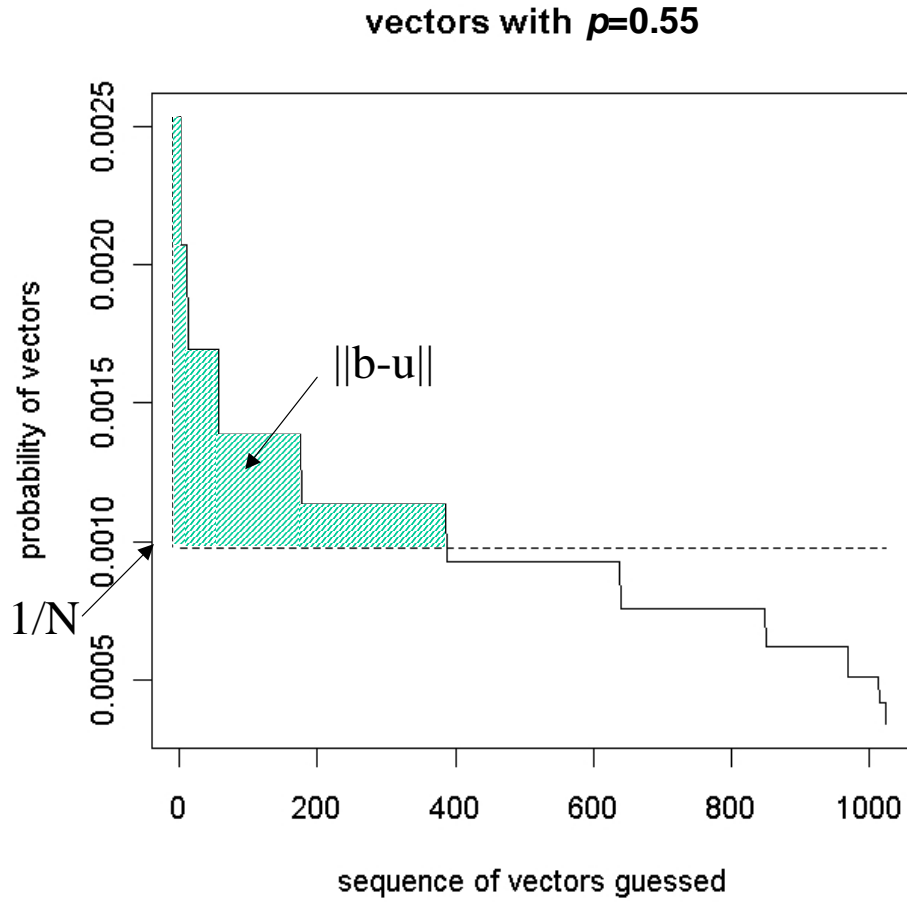


Figure 7: Probabilities of vectors of length $n = 10$

The term $\|b-u\|$ quantifies the “advantage” over the uniform distribution one has when considering all vectors that are assumed with probability larger than 2^{-n} . For $p > 0.5$ we define

$$k = \left\lfloor n \frac{\lg(1-p) + \lg 2}{\lg(1-p) - \lg p} \right\rfloor,$$

i.e., k is the down-rounded solution x of the equation $2^{-n} = p^x (1-p)^{n-x}$. From this, one immediately obtains the inequations

$$p^k \cdot (1-p)^{n-k} \leq 2^{-n} \leq p^{k+1} \cdot (1-p)^{n-k-1}.$$

In our example

$$k = \left\lfloor 10 \frac{\lg(0.45) + \lg 2}{\lg(0.45) - \lg(0.55)} \right\rfloor = \lfloor 5.250419 \rfloor = 5$$

$$0.55^5 \cdot 0.45^5 \approx 0.0009287012 \leq 2^{-10} \approx 0.0009765625 \leq 0.55^6 \cdot 0.45^4 \approx 0.001135079$$

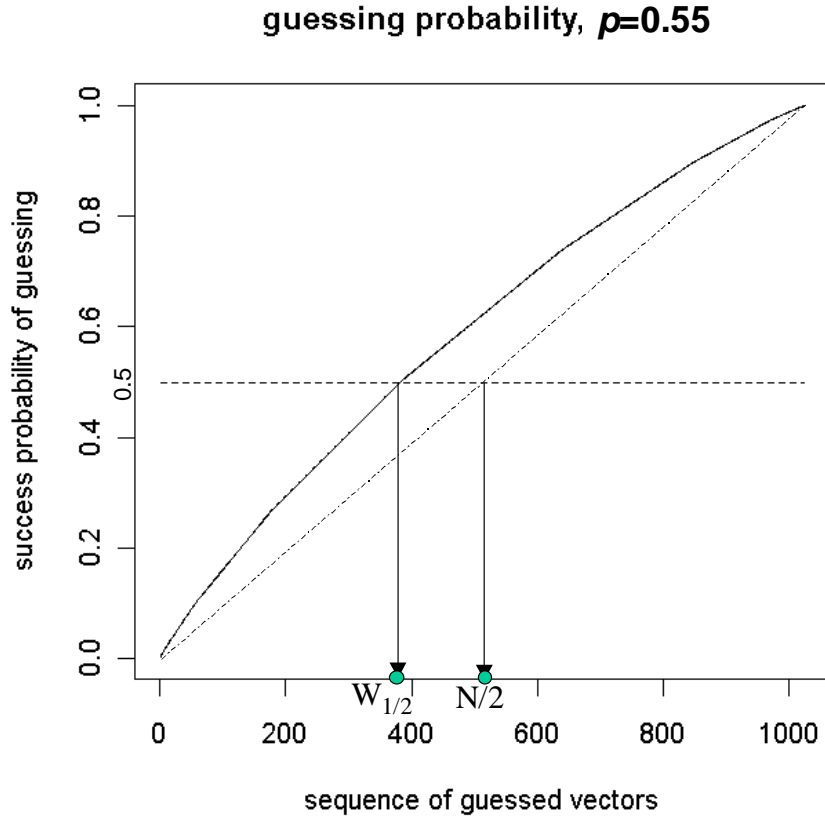


Figure 8: Success probability ($p = 0.55$, $n = 10$)

376 The Figure 8 shows the success probability as a function of the number of vectors that have already been checked. The dotted line represents the success probability for the uniform distribution, which has work factor 2^{n-1} .

It is well known that $P\{h(X) = k\} = \binom{n}{k} p^k (1-p)^{n-k}$ (binomial distribution). The statistics program R may be used to evaluate the above formula for concrete parameters:

$$\text{dbinom}(k, n, p) \text{ calculates } P\{h(X) = k\} = \binom{n}{k} p^k (1-p)^{n-k}$$

pbinom(m,n,p) calculates the cumulative distribution function $P\{h(X) \leq m\} = \sum_{k=0}^m \binom{n}{k} p^k (1-p)^{n-k}$

pbinom(m,n,p,lower.tail=F) calculates $P\{h(X) > m\} = 1 - \sum_{k=0}^m \binom{n}{k} p^k (1-p)^{n-k}$

qbinom(q,n,p) calculates the quantile function, given by $\min\{m \mid P\{h(X) \leq m\} \geq q\}$

where k and m are vectors of quantiles⁴¹, *size* denotes the number of observations (in our example, n), and *prob* stands for a vector of probabilities (in our example, p).

377 Alternatively, we may apply the binomial distribution to the number of 0's instead of applying it to the number of 1's. Note that we begin with a vector that contains no 0, etc.. For $n = 10$ and $p = 1 - 0.55 = 0.45$ we get

$$pbinom(3,10,0.45) = 0.2660379, pbinom(4,10,0.45) = 0.5044046$$

$$pbinom(3,10,0.5) * (2^{10}) = 176 \text{ (i.e number of vectors with maximum 3 times "0")}$$

Each vector with 4 times "0" has a probability of $(0.45^4) * (0.55^6) = 0.001135079$.

$$(0.5 - pbinom(3,10,0.45)) / ((0.45^4) * (0.55^6)) = 206.1196$$

Indeed we get

$$pbinom(3,10,0.45) + 206 * (0.45^4) * (0.55^6) = 0.4998643$$

$$pbinom(3,10,0.45) + 207 * (0.45^4) * (0.55^6) = 0.5009994$$

Finally

$$w_{0.5} = pbinom(3,10,0.5) + \left\lceil \frac{0.5 - pbinom(3,10,0.45)}{0.45^4 \cdot 0.55^6} \right\rceil = 383.$$

5.2. Contingency tables

Example 2: Contingency table 2x4

378 Assume that one observes two sequences of 400 events and compiles contingency tables as shown below. Applying the build in R χ^2 test function `chisq.test`, one obtains results as given in the two tables (3 degrees of freedom):

Observed data, case 1

⁴¹ The α -quantil of the random variable is the value p_α that $P\{X \leq p_\alpha\} = \alpha$.

	0	1	2	3	Sum of rows
0	41	40	51	50	182
1	41	62	54	61	218
Sum of column	82	102	105	111	Sum in general: 400

R script:

```
count_tab1 <- matrix(c(41,41,40,62,51,54,50,61),nrow=2,ncol=4)
      [,1] [,2] [,3] [,4]
[1,]  41  40  51  50
[2,]  41  62  54  61
count_tab1; chisq.test(count_tab1, correct=TRUE)
      Pearson's Chi-squared test
data:  count_tab1
X-squared = 2.7028, df = 3, p-value = 0.4398
```

The chi-squared = 2.7028 and the p-value = 0.4398. The test does not reject the Null hypothesis on independence.

Observed data, case 2

	0	1	2	3	Sum of rows
0	61	59	61	34	215
1	44	36	46	59	185
Sum of column	105	95	107	93	Sum in general: 400

R script:

```
count_tab2 <- matrix(c(61,44,59,36,61,46,34,59),nrow=2,ncol=4)
      [,1] [,2] [,3] [,4]
[1,]  61  59  61  34
[2,]  44  36  46  59
count_tab2; chisq.test(count_tab2, correct=TRUE)
      Pearson's Chi-squared test
data:  count_tab2
X-squared = 14.9783, df = 3, p-value = 0.001835
```

The chi-squared = 14.9783 and the p-value = 0.001835. The test rejects the Null hypothesis on independence because the p-value is less than 0.01.

Example 3: Contingency table 2x2

379 For testing the 1bit-to-1bit dependency (i.e., $x = y = 1$) by means of a 2×2 -contingency table, one may calculate the power of the test.

	Frequencies			Probabilities		
	0	1		0	1	
0	n_{00}	n_{01}	$n_{0\bullet}$	$p_{00} = P\{(B_{i+1}, B_i) = (00)\}$	$p_{01} = P\{(B_{i+1}, B_i) = (01)\}$	$p_{0\bullet} = P\{B_{i+1} = 0\}$
1	n_{10}	n_{11}	$n_{1\bullet}$	$p_{10} = P\{(B_{i+1}, B_i) = (10)\}$	$p_{11} = P\{(B_{i+1}, B_i) = (11)\}$	$p_{1\bullet} = P\{B_{i+1} = 1\}$
	$n_{\bullet 0}$	$n_{\bullet 1}$	n	$p_{\bullet 0} = P\{B_i = 0\}$	$p_{\bullet 1} = P\{B_i = 1\}$	

Since the process is assumed to be stationary, $p_{\bullet 0} = p_{0\bullet} = p_{00} + p_{01}$ and $p_{\bullet 1} = p_{1\bullet} = p_{10} + p_{11}$. Suppose that the consecutive bits are not independent, i.e., that $p_{01} \neq p_{11}$. The power of the test β is the probability that the test rejects the Null hypothesis if it is false. This depends on the bias, i.e., $p_0 := p_{01}$ and $p_1 := p_{11}$, the number of observations n , and the probability of error type 1, i.e., the level of significance α . The R function `power.prop.test` allows the calculation of any of the parameters n , p_0 , p_1 , α , and β if the others are given. Exactly one of the parameters n , p_0 , p_1 , α , and β must be passed as NULL, and this parameter is determined from the others.

Example 4: Calculation of the power of a 2×2 -contingency table test

380 We calculate the power of a 2×2 -contingency table test

```
power.prop.test(p1=0.495,p2=0.505,sig.level=0.05,n=70000)
Two-sample comparison of proportions power calculation
n = 70000
p1 = 0.495
p2 = 0.505
sig.level = 0.05
power = 0.9626076
alternative = two.sided
NOTE: n is number in *each* group
Calculation of the number of necessary observations
power.prop.test(p1=0.495,p2=0.505,sig.level=0.01,power=0.99)
```

```
Two-sample comparison of proportions power calculation
n = 120151
p1 = 0.495
p2 = 0.505
sig.level = 0.01
power = 0.01
```

```
alternative = two.sided
NOTE: n is number in *each* group.
```

381 The following script for R is an example showing how to analyse binary data for dependencies between consecutive bits. Note that this example is not optimized code.

```
# contingency table analysis of bit patterns in byte sequences

# bits are enumerated for 0 to 7, least significant bit is 0-bit

# pattern is sequence of bit numbers, e.g. 0, 2 and 3


# definition of helping function

byte2bit <- matrix(rep(0,8*256),ncol=256)

for (i in 1:8)
{
  b <- 2^(i-1)

  for (j in 1:256) byte2bit[i,j] <- ((j-1)%/b)%2
}

# function fbyte2bit(x,i) generates an array of bits, where
# fbyte2bit(x,i) ist bit i in byte x

fbyte2bit = function(x,i)
{
  if ((0 <= min(x)) & (max(x) <=255)) y <- byte2bit[i+1,x+1] else y <- NA

  return(y)
}


# definition of parameters

setwd("D:\\test") # put the working directory here

data_file_name <- "random.bin" # put the data file name here

data_length <- file.info(data_file_name)$size # length of data in bytes
```

```

first_pattern <- c(0,2,3)          # example of first bit pattern to test

second_pattern <- c(5,6,7)        # example of second bit pattern to test


# reading test byte stream

byte_sequence<-readBin(data_file_name, integer(), size=1, n=data_length,
signed=FALSE)

# calculate vector of byte values with selected pattern

first_pattern_vector              <-
(2^first_pattern)%*%fbyte2bit(byte_sequence,first_pattern)

second_pattern_vector            <-
(2^second_pattern)%*%fbyte2bit(byte_sequence,second_pattern)

(chisq.test(table(first_pattern_vector,second_pattern_vector)))

```

382 It produces an output like this

Pearson's Chi-squared test

```

data:  table(first_pattern_vector, second_pattern_vector)

X-squared = 45.2399, df = 49, p-value = 0.6263

```

5.3. Forward and backward secrecy

383 In this section we discuss forward secrecy, backward secrecy, and enhanced backward secrecy as security capabilities. We provide some elementary examples for illustration. These security capabilities are typically for DRNGs and the cryptographic post-processing algorithms for PTRNGs and NPTRNGs. Note that the following examples are not intended as advice for good DRNG design.

384 For simplicity, we interpret a DRNG as a Mealy machine; that is, a pure DRNG that runs without any external input after seeding,

$$\varphi: S \rightarrow S, s_{n+1} := \varphi(s_n) \text{ and } \psi: S \rightarrow R, r_n := \psi(s_n) \text{ for all } n \geq 1. \quad (59)$$

385 Forward secrecy is the assurance that subsequent (future) values cannot be determined from current or previous output values. Suppose that the current or previous output values $r_{i_1}, r_{i_2}, \dots, r_{i_k}$ at time i_1, i_2, \dots, i_k are known and that the attacker wants to calculate the output value r_n for some index $i_1, i_2, \dots, i_k < n$. The attacker might use the system of relations:

$$r_n \in \psi(\varphi^{n-i_j}(\psi^{-1}(r_{i_j}))) \text{ for } j = 1, 2, \dots, k. \quad (60)$$

(Note that the output function ψ is not invertible in general, and hence one must check all the elements in the pre-image $\psi^{-1}(r_{i_j})$.) If the internal state s is known, the state transition function φ and the output function ψ are easy to evaluate. Thus, it shall be difficult to calculate the pre-image, e.g., because it is too large or if ψ is a cryptographic one-way function.

- 386 Backward secrecy ensures that previous output values cannot be determined from current or future output values. Assume that the current or future output values $r_{i_1}, r_{i_2}, \dots, r_{i_k}$ at time instant i_1, i_2, \dots, i_k are known and that the attacker wants to calculate the output value r_n for some index $i_1, i_2, \dots, i_k < n$. The attacker might apply the system of relations:

$$r_n \in \psi(\varphi^{n-i_j}(\psi^{-1}(r_{i_j}))) \text{ for } j = 1, 2, \dots, k. \quad (61)$$

If the internal state s is known, the state transition function φ and the output function ψ are easy to evaluate. Thus, the composition $\varphi^{n-i_j} \circ \psi^{-1}$ shall be hard to compute (note that $n - i_j < 0$ for $j = 1, 2, \dots, k$). One may choose the state transition function φ and the output function ψ as appropriate cryptographic one-way functions.

- 387 Enhanced backward secrecy ensures that previous values cannot be determined from the current internal state, or from current or future output values. Because of lack of (unknown) input for a DRNG as Mealy machine, knowledge of the current internal state s_c is sufficient to calculate current and future output values $r_{i_1}, r_{i_2}, \dots, r_{i_k}$ at times i_1, i_2, \dots, i_k , $i_1, i_2, \dots, i_k > c$. Thus, the attacker knows the current internal state s_c and wants to calculate the output value r_n , $c > n$. The attacker might use the system of relations:

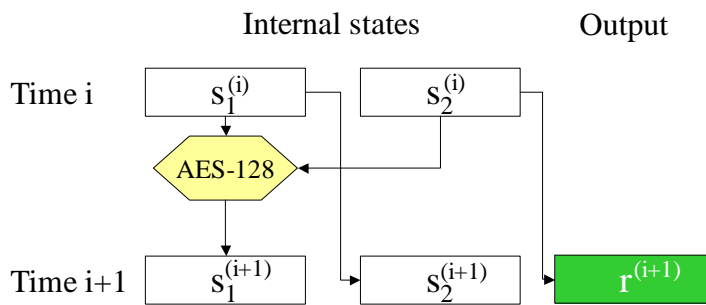
$$r_n \in \psi(\varphi^{n-c}(s_c)) \text{ for } j = 1, 2, \dots, k. \quad (62)$$

Because the output function ψ is easy to calculate, the state transition function φ and its negative power φ^j shall be cryptographic one-way functions.

- 388 In the following examples, suppose the DRNG has the internal state $s^{(i)} = (s_1^{(i)}, s_2^{(i)})$, $s_j^{(i)} = (s_{j1}^{(i)}, s_{j2}^{(i)}, \dots, s_{j128}^{(i)})$ for $j = 1, 2$; and $AES(k, x)$ denotes AES-128 [AES] for plaintext x and key k .

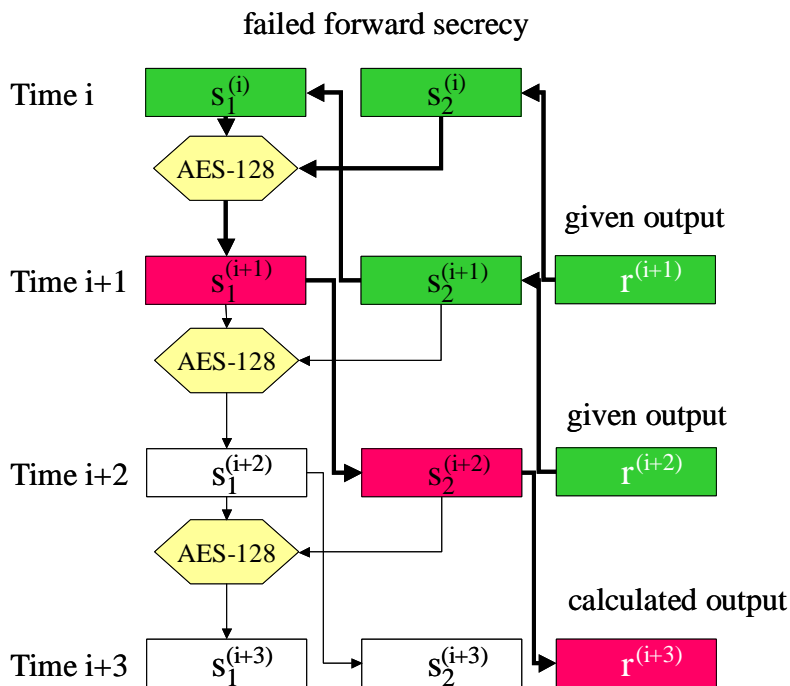
Example 5: (backward secrecy, no forward secrecy)

- 389 Suppose the DRNG uses the state transition function $s^{(i+1)} = \varphi(s^{(i)}) = (AES(s_2^{(i)}, s_1^{(i)}), s_1^{(i)})$ and the output function $r^{(i+1)} = \psi(s^{(i)}) = s_2^{(i)}$. Note that the plaintext $s_2^{(i)}$ becomes the key in the next step.

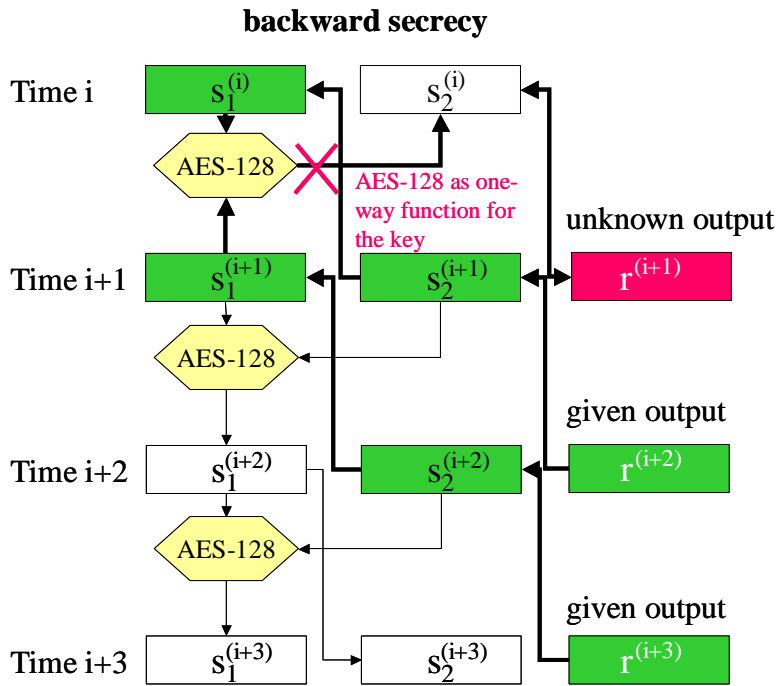


$$r^{(i+3)} = AES(\kappa^{(i+1)}, r^{(i+2)})$$

390 Breaking forward secrecy requires a simple AES-128 calculation .



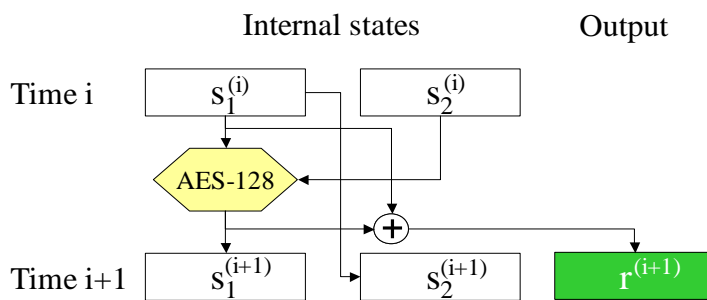
391 This DRNG ensures backward secrecy because $c = AES(k, p)$ as a one-way function for the key k prevents calculation of previous output, even if the (plaintext/ciphertext) pair (p, c) is known.



As the attacker knows the current output $r^{(i+2)}$ and partly knows the internal state $s_1^{(i+2)}$ or $s_2^{(i+2)}$ at time $i+2$, the attacker is able to calculate the previous internal state at time $i+1$, because $s_1^{(i+1)} := s_2^{(i+2)}$, $s_1^{(i+1)} := AES^{-1}(r^{(i+2)}, s_1^{(i+2)})$ and $s_2^{(i+1)} := r^{(i+2)}$, but cannot calculate $r^{(i+1)}$.

Example 6: (forward and backward secrecy)

392 Suppose the DRNG uses the state transition function $s^{(i+1)} = \varphi(s^{(i)}) = (AES(s_2^{(i)}, s_1^{(i)}), s_1^{(i)})$ (as in the previous example) and the output function $r^{(i+1)} = \psi(s^{(i)}) = AES(s_2^{(i)}, s_1^{(i)}) \oplus s_1^{(i)}$.

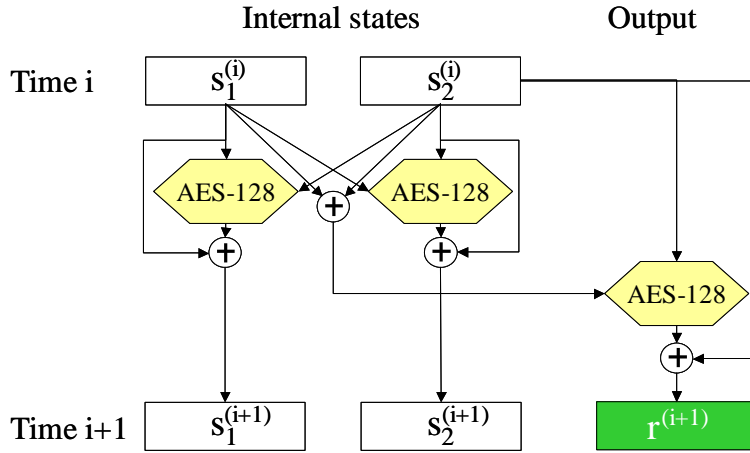


The attacker cannot calculate the internal state from the output values, because the output function is a one-way function for any fixed $s_2^{(i)}$ ⁴².

⁴² under the assumption that the AES-128 behaves like a randomly-selected function

Example 7: (enhanced backward secrecy)

393 Suppose the DRNG uses the state transition function $s^{(i+1)} = \varphi(s^{(i)}) = (AES(s_2^{(i)}, s_1^{(i)}) \oplus s_1^{(i)}, AES(s_1^{(i)}, s_2^{(i)}) \oplus s_2^{(i)})$ and the output function $r^{(i+1)} = \psi(s^{(i)}) = AES(s_1^{(i)} \oplus s_2^{(i)}, s_2^{(i)}) \oplus s_2^{(i)}$.



The state transition function and the output function are different one-way functions.

5.4. Examples of post-processing algorithms

394 This section considers examples of post-processing algorithms. For further exposition, we refer the interested reader to [Schi09b], section 2.5.

395 We use a description as generalized Mealy machine $(S, I, R, \varphi^*, \psi^*)$. Let S denote the set of internal states, I input alphabet, $\bar{i} = (i_1, i_2, \dots, i_n)$ the input sequence of strings I^* over I , R the output alphabet $\bar{r} = (r_1, r_2, \dots, r_m)$ the output sequence of strings R^* over R , s_0 the initial internal state, $\varphi^* : S \times I^* \rightarrow S$ the state function, $s_{k+1} := \varphi^*(s_k, i_k)$, and $\psi^* : S \times I^* \rightarrow R^*$ the output function, $r_{k+1} := \psi^*(s_k, i_k)$. Let ϵ denote the empty string.

5.4.1. Von Neumann unbiasing

396 Von Neumann unbiasing works asynchronously, i.e., it receives pairs of bits $i_k = (j_{2k}, j_{2k+1})$ as input, but does not generate any output for certain input pairs. Moreover, it has no internal state, i.e., S is the empty set. $I = \{0,1\}^2$, $R = \{0,1,\epsilon\}$ and

$$r'_k = \psi^*(s, i_k) = \begin{cases} 0 & \text{for } i_k = (01) \\ 1 & \text{for } i_k = (10) \\ \epsilon & \text{for } i_k = (00) \\ \epsilon & \text{for } i_k = (11) \end{cases}$$

The output sequence \bar{r} is the concatenation of all $r'_k \in \{0,1\}$, $k \in \overline{1,n}$.

- 397 **Lemma 3:** If $(j_1, j_2, \dots, j_{2n})$ is a sequence of independent but biased bits, von Neumann unbiasing generates a sequence $\bar{r} = (r_1, r_2, \dots, r_m)$ of independent and unbiased output bits with $m \leq n/2$.

5.4.2. Xoring of non-overlapping segments of independent bits

- 398 Assume that $\bar{X} = (X_1, X_2, \dots, X_n)$ with independent, identically-distributed biased binary-valued random variables as components, i.e., for X_j all $j \in \overline{1,n}$ we have $P\{X_j = 0\} = p_0$ and $P\{X_j = 1\} = p_1$ with $\mathcal{E} = P\{X_i = 0\} - P\{X_i = 1\}$. Denote

$$p_{0,n} := P\left\{\sum_{i=1}^n X_i \equiv 0 \pmod{2}\right\} = \sum_{k=0}^{\lfloor n/2 \rfloor} \binom{n}{2k} p_1^{n-2k} p_0^{2k},$$

$$p_{1,n} := P\left\{\sum_{i=1}^n X_i \equiv 1 \pmod{2}\right\} = \sum_{k=0}^{\lfloor (n-1)/2 \rfloor} \binom{n}{2k+1} p_1^{n-2k-1} p_0^{2k+1}, \text{ and}$$

$$\mathcal{E}_n := p_{0,n} - p_{1,n}.$$

We calculate

$$\begin{aligned} \mathcal{E}^n &= (p_0 - p_1)^n = \sum_{k=0}^n (-1)^k \binom{n}{k} p_0^{n-k} p_1^k = \sum_{k=0}^{\lfloor n/2 \rfloor} \binom{n}{2k} p_0^{n-2k} p_1^{2k} - \sum_{k=0}^{\lfloor (n-1)/2 \rfloor} \binom{n}{2k+1} p_0^{n-2k-1} p_1^{2k+1} = \\ &= p_{0,n} - p_{1,n}. \end{aligned}$$

- 399 The xor-sum of the bits smoothes exponentially the bias of the independent bits. These sums are independent, as well.

5.4.3. Two sources

- 400 Suppose that post-processing calculates the function $Z = f(X, Y)$ of two random variables X and Y with values in (z_1, z_2, \dots, z_n) , where X ranges over the set (x_1, x_2, \dots, x_n) with distribution $P\{x_i = X\} = p_i$, w.l.o.g. $p_1 \leq p_2 \leq \dots \leq p_n$, and Y over the set (y_1, y_2, \dots, y_n) with distribution $P\{y_i = Y\} = q_i$, w.l.o.g. $q_1 \leq q_2 \leq \dots \leq q_n$. The distribution of Z is given by $P\{z = f(X, Y)\} = \sum_{z=f(x,y)} P\{x = X, y = Y\}$. Suppose further that f is invertible in the second argument, or more precisely: For each i and j , there exists exactly one k with $z_i = f(x_j, y_k)$, $i, j, k \in \overline{1,n}$. Therefore, the function f generates for each i a permutation π_i over $\overline{1,n}$

according to $z_i = f(x_j, y_{\pi_i(j)})$. We can write $P\{z_i = f(X, Y)\} = \sum_i P\{x_i = X, y_{\pi_i(j)} = Y\}$. If the random variables X and Y are independent, one can write the distribution of Z like this:

$$P\{z_i = f(X, Y)\} = \sum_{j=1}^n p_j \cdot q_{\pi_i(j)}$$

For sums on the right side, we can apply the rearrangement inequality [HaLP34] like this:

$$\sum_{k=1}^n p_k \cdot q_{n-k+1} \leq P\{z_i = f(X, Y)\} = \sum_{j=1}^n p_j \cdot q_{\pi_i(j)} \leq \sum_{k=1}^n p_k q_k \quad (63)$$

401 This formula provides an estimate for the distribution of Z as the output of post-processing using the random variables X and Y . If the random variables X and Y have the same distribution, i. e., $P\{x_i = X\} = P\{x_i = Y\} = p_i$ for all $i \in \overline{1, n}$, this formula shows that the collision entropy of X (or Y) (cf. paragraph 145) is a lower bound for the min-entropy of Z (cf. paragraph 143):

$$H_{\min}(Z) = -\log_2 \max_{i \in \overline{1, n}} \{p(z_i)\} \geq -\log_2 \left(\sum_{x \in X} (P\{X = x\})^2 \right) = H_2(X).$$

402 Assume that the binary-valued random vectors X and Y have length l , and assume that their realisations are observed from the same stationary random source but at time instants that are sufficiently far away to exclude dependencies between X and Y . The post-processing algorithm f is defined as bitwise xor. Then the right-hand side of the inequality (63) equals the maximum probability of the “zero” vector $\bar{0} = (0, 0, \dots)$, and the left-hand side can be substituted by the maximum of all probabilities of post-processing output:

$$\max_i \{P\{z_i = f(X, Y)\}\} = \sum_{j=1}^n p_j^2 = P\{z = \bar{0}\}$$

and finally we get:

$$H_{\min}(Z) = -\log_2 \left(\sum_{i=1}^n p_i^2 \right) = -\log_2 P\{\bar{0} = Z\}$$

This formula reduces the estimation of the min-entropy to the estimation of the probability of the “zero” vector $\bar{0}$.

5.4.4. Uniformly distributed input data for random mappings

403 In this example we consider the effect of a random mapping on uniformly distributed input data.

404 More precisely, let $Z_n := \{0, 1, \dots, n-1\}$, $Z_m := \{0, 1, \dots, m-1\}$, $f := Z_n \rightarrow Z_m$, $V_{(f)s} := \{z \in Z_m : |f^{-1}(z)| = s\}$ and $v_{(f)s} := |V_{(f)s}|$. For a uniformly distributed random

variable X on Z_n , we obtain $\Pr(f(X) = z) = \frac{s}{n}$ for each $z \in Z_m$ with exactly s pre-images.

Hence $\Pr(f(X) \in V_{(f)s}) = \frac{sv_s}{n}$ and $\Pr(f(X) \in \bigcup_{s=r}^n V_{(f)s}) = \sum_{r=s}^n \frac{sv_s}{n}$, and the corresponding work

factor is given by $\sum_{s=r}^n v_s$. If the mapping f is selected randomly, i.e., if f may be viewed as a realisation of a uniformly distributed random mapping F , we obtain:

$$\begin{aligned} e_r &:= E_F(\Pr(F(X) \in \bigcup_{s=r}^n V_{(F)s})) = \sum_{r=s}^n \frac{sE_F(v_{(F)s})}{n} = \sum_{r=s}^n \frac{sE_F(\sum_{z \in Z_m} 1_{\{|F^{-1}(z)|\}})}{n} \\ &= \sum_{r=s}^n \frac{s \sum_{z \in Z_m} \Pr_F(|F^{-1}(z)| = s)}{n} = \sum_{s=r}^n \frac{s}{n} m \binom{n}{s} p^s (1-p)^{n-s} \\ &= \sum_{s=r}^n \binom{n-1}{s-1} p^{s-1} (1-p)^{n-s} = \Pr(Y \geq r-1) \end{aligned}$$

with $p = \frac{1}{m}$ while Y denotes a $B(n-1, p)$ -distributed random variable. Further, $E_F(\cdot)$ and

$\Pr_F(\cdot)$ denote the expectation and the probability with regard to the random mapping F , respectively. In particular, e_r quantifies the average probability that $F(X)$ has $\geq r$ pre-images. The corresponding work factor equals

$$w_{e_r}(F(X)) = E_F(|\bigcup_{s=r}^n V_{(F)s}|) = \sum_{s=r}^n m \binom{n}{s} p^s (1-p)^{n-s} = m \Pr(Y' \geq r)$$

where Y' denotes a $B(n, p)$ -distributed random variable.

405 In particular, $E(Y') = \frac{n}{m}$ and $E(Y) = \frac{n-1}{m}$. If n and m are large (the case we are interested

in) and if $\gamma := \frac{n}{m} \gg 1$, the Central Limit Theorem implies

$$e_r := 1 - \Phi\left(\frac{r-1-0.5-\gamma}{\sqrt{\gamma}}\right) = \Phi\left(\frac{\gamma-r+1.5}{\sqrt{\gamma}}\right) \quad \text{and} \quad w_{e_r}(F(X)) := m\Phi\left(\frac{\gamma-r+0.5}{\sqrt{\gamma}}\right).$$

(For $n \approx m$, the Poisson approximation should be more convenient.) Similarly, for a uniformly distributed random variable U on Z_m we obtain the work factor $w_{e_r}(U) = me_r$. For $r \geq 1$ the difference of work factors

$$w_{e_r}(U) - w_{e_r}(F(X)) = m\left(\Phi\left(\frac{\gamma-r+1.5}{\sqrt{\gamma}}\right) - \Phi\left(\frac{\gamma-r+0.5}{\sqrt{\gamma}}\right)\right)$$

quantifies the ‘distance’ between U and $F(X)$ on the elements of Z_m with pre-image size $\geq r$. Linear interpolation in r yields an approximation of the work factor $w_\alpha(F(X))$ and the corresponding work factor defect for each parameter $\alpha \in (0,1)$. More precisely,

$w_\alpha(F(X)) = m\Phi\left(\frac{\gamma - r_\alpha + 0.5}{\sqrt{\gamma}}\right)$ with $r_\alpha = r - 1 + \frac{\alpha - e_{r-1}}{e_r - e_{r-1}}$ for $e_r \leq \alpha \leq e_{r-1}$ while trivially $w_\alpha(U) = m\alpha$. The difference $w_\alpha(U) - w_\alpha(F(X))$ ('work factor defect' for parameter α) is clearly bounded by $m(\Phi(\frac{0.5}{\sqrt{\gamma}}) - \Phi(\frac{-0.5}{\sqrt{\gamma}}))$.

406 In the context of cryptographic post-processing algorithms we are mainly interested in parameters n and m that are powers of two (i.e., $n = 2^n$, $m = 2^m$). The table below provides exemplary numerical values.

Table 14: Work factor and work factor defect for uniform mappings with equidistributed input

γ	r	$w_{e_r}(U)$	$w_{e_r}(F(X))$	$w_{e_r}(U) - w_{e_r}(F(X))$
2^{10}	1026	$0.494m$	$0.481m$	$0.013m$
2^{10}	1025	$0.506m$	$0.494m$	$0.012m$
2^{10}	1100	$0.0099m$	$0.0092m$	$0.0008m$
2^8	258	$0.4875m$	$0.4627m$	$0.0248m$
2^8	257	$0.5125m$	$0.4875m$	$0.0250m$

γ	α	$w_\alpha(U)$	$w_\alpha(F(X))$	$w_\alpha(U) - w_\alpha(F(X))$	$\max_{\alpha' \in (0,1)} \{w_{\alpha'}(U) - w_{\alpha'}(F(X))\}$
2^{10}	0.5	$0.5m$	$0.4875m$	$0.0125m$	$0.0126m$
2^{10}	0.1	$0.1m$	$0.4947m$	$0.0053m$	$0.0126m$
2^8	0.5	$0.5m$	$0.4751m$	$0.0249m$	$0.025m$

5.5. Examples of online test, tot test, and start-up test

407 In all three examples in this section, we assume that the internal random numbers are stored in a 512-bit FIFO. The FIFO outputs internal random numbers upon external request. The FIFO is filled up with currently-generated consecutive internal random numbers r_1, r_2, \dots at an instant when there are only between 128 and 256 fresh random bits left. The PTRNG continuously generates internal random numbers. For the second and third example, we assume that the das-random numbers are binary-valued, i.e., one random bit is generated per time unit.

5.5.1. An online test of the internal random numbers

408 All internal random numbers that are used to fill up the FIFO have been tested. The internal random numbers are interpreted as bit strings and segmented into 4-bit words. χ^2 goodness-of-fit tests on 128 bits (4-bit words) are applied. The online test fails if the test value exceeds 65.0. According to ([Kanj95], pp. 69), the test variable is approximately χ^2 -distributed with 15 degrees of freedom, which gives rise to the significance level $3.8 \cdot 10^{-7}$.

409 If a test fails, i.e., if the null hypothesis (i.e. the internal random bits were generated by an ideal RNG) is rejected, the PTRNG is shut down and an error message is generated. The error

message is logged and the PTRNG must be restarted manually. Only two manual restarts are permitted within the life cycle of the PTRNG.

- 410 Internal random numbers that are not stored in the FIFO are neither saved nor tested unless they are needed to complete a sample for the online test. It may be expected that the PTRNG applies around 1000 online tests per year. For an ideal random number generator, the χ^2 -distribution function gives rise to a probability of around $3.8 \cdot 10^{-4}$ that there will be at least one noise alarm within a year (cf. to the next example).
- 411 This online test meets functional requirement PTG.1.3. Whether the test detects a total failure of the noise source clearly depends on the post-processing algorithm.

5.5.2. A straightforward online test

- 412 In the preceding example, the internal random numbers were tested, which were stored in the FIFO or were used to fill up a sample for the online test. In this example, we instead assume that an online test applies to those das-random numbers that are used to generate the stored internal random bits or are needed to complete a sample for the online test. Again, a χ^2 goodness-of-fit test on 128 bits (4-bit words) is applied, and a test value that exceeds 65.0 causes a noise alarm. The consequences of a noise alarm are the same as in the first example.
- 413 We analyse the proposed solution with regard to the requirements of class PTG.2 (also PTG.3), which are more restrictive than those of PTG.1.
- 414 As pointed out earlier, the online test should be selected with regard to the stochastic model of the noise source. Thus, whether the χ^2 goodness-of-fit test is appropriate depends on the concrete noise source. This aspect is outside the scope of this example.
- 415 A noise alarm occurs if a single test gives a value greater than 65.0. This is a very rare event, at least under the null hypothesis (ideal noise source), which implies independent and uniformly distributed 4-bit words. However, this approach has two disadvantages that will be described below.
- 416 On the one hand, even under the null hypothesis, the test variable is only asymptotically χ^2 -distributed with 15 degrees of freedom. More precisely, this is the limit distribution of the test variable when the sample size tends towards infinity. If the sample size is ‘small’ especially for large rejection bounds, i.e., for small failure probabilities, the relative error $|p_{exact} - p_{approx}| / p_{approx}$ can be large. Here, p_{exact} denotes the exact rejection probability, whereas p_{approx} is the approximate rejection probability derived from the χ^2 distribution. For example, for the sample size 320 bit (= 80 four-bit words) for the rejection bound 65.0 the relative error is 10.1 ([Schi01], Sect. 4). We put p_{approx} and not p_{exact} into the denominator, since the designer of a PTRNG grounds his further considerations on the approximate probability. For 128 (4-bit words), this ratio should be smaller, but the number of noise alarms should be considerably greater than is to be expected on the basis of the asymptotic limit distribution. This might affect functionality aspects, but it is not a security issue. For other statistical tests, this effect may be converse, resulting in considerably fewer failures than expected.

- 417 A second drawback of the proposed online test is that it is hardly possible to estimate the true rejection probability if the distribution of the das-random numbers deviates from the output of an ideal RNG (e.g., because of a bias).
- 418 Whether this online test fulfils the functional requirements PTG.2.3 and PTG.2.4 (equivalently, PTG.3.3 and PTG.3.4) first depends on whether the χ^2 goodness-of-fit test is appropriate for the given noise source. Moreover, it demands analysis of the rejection probabilities for tolerable and non-tolerable weaknesses of the das-random numbers. As pointed out above, this is difficult, especially for rare events (large rejection boundaries).
- 419 These problems are the motivation to propose a more sophisticated approach that is discussed in the next example.

5.5.3. A more sophisticated online test procedure

- 420 In this section we discuss a generic approach that covers the online test, tot test, and start-up test. As in the previous example, the scope of testing is the das-random numbers that are used to generate the internal random numbers stored in the FIFO or needed to complete a sample for the online test. For details, the interested reader is referred to [Schi01]. This online test procedure also is discussed in [AIS31An], Example E.7.
- 421 The start-up test is performed (a single χ^2 test over 128 bits (4-bit words)) when the PTRNG is started. The PTRNG passes the start-up test if the test variable is ≤ 65.0 . This evaluation rule detects a total breakdown of the noise source and very obvious statistical weaknesses immediately when the PTRNG is started. The start-up test thus fulfils functional requirement PTG.2.1 and the first part of functional requirement PTG.2.3 (also PTG.3.1 and PTG.3.3).
- 422 As explained below, the online test procedure also covers the tot test functionality while the PTRNG is in operation.
- 423 First, the type of the so-called “basic test” must be selected. The stochastic model of the noise source clearly should be taken into account, because an unsuitable basic test may reduce the effectiveness of the online test procedure considerably; however, this aspect is not in the scope of this example. For simplicity, we decided on a χ^2 goodness-of-fit test over 128 bits (4-bit words) as in the previous examples. We point out that the proposed online test procedure is generic and transfers almost literally to other basic tests.
- 424 A test suite is made up of a maximum of $N = 512$ basic tests (here: 512 χ^2 tests). In the following steps, we will use the notation C_1, C_2, \dots to refer to the test variables, i.e., random variables that correspond to the test values of basic tests. In particular, $H_0 := E(C_1)$ (i.e. the expected value of the basic test variable under the null hypothesis) and $H_j := (1 - \beta)H_{j-1} + \beta C_j$ for $j = 1, 2, \dots$ with $\beta = 2^{-6}$, whereby the test variables C_j and H_j are each rounded to 6 binary digits. This allows the calculation of the “history variables” H_1, H_2, \dots using integer arithmetic. The observed test values and the computed history variables are denoted with small letters (c_1, c_2, \dots and h_1, h_2, \dots , respectively). Three evaluation rules apply for each step $1 \leq j \leq N$:

(i) if $c_{j-2}, c_{j-1}, c_j > 26.75$, then there is a preliminary noise alarm,

(ii) if $c_j \notin [13.0, 17.0]$, then there is a preliminary noise alarm,

(iii) if $c_j > 269.5$, then there is a noise alarm.

- 425 If no preliminary noise alarm occurs within a test suite, a new test suite is started. Each preliminary noise alarm causes the current test suite to be cancelled and the FIFO to be deleted. Each preliminary noise alarm is logged. If three consecutive test suites are stopped due to a preliminary noise alarm, a noise alarm occurs, the PTRNG is shut down, and a corresponding error message is generated. Note that evaluation rule (iii) covers the tot test functionality as is discussed below.
- 426 For demonstration purposes, we assume for simplicity that the digitised noise signals are realisations of binary-valued independent and identically distributed random variables. The probability $P(1)$ that a das-random bit assumes the value 1 might depend on the individual device and might change in the course of time due to aging effects.
- 427 In a real-world evaluation, an assumption like this (independence assumption) should be the result of a thorough analysis of the stochastic model and investigation of prototypes. For instance, the analysis of the stochastic model (cf. e.g., to [KiSc08]) might imply that the random variables are stationary with rapidly decreasing dependencies, and that the das-random numbers pass test suite B; moreover, no one-step dependencies were detected.
- 428 We assume that with regard to the algorithmic post-processing algorithm (which is beyond the scope of this example) that it is sufficient if $P(1) \in [0.49, 0.51]$. If this probability lies outside of the interval $[0.475, 0.525]$, the online tests should soon recognise this fact and trigger a noise alarm.
- 429 Table 15 shows the probability of a preliminary noise alarm within a test suite and the average number of noise alarms per year. Here it has been assumed that 1584 basic tests are performed each day (of which 144 are based on the event of filling up the FIFO).

Table 15: Probability for a noise alarm within a test suite and the expected number of noise alarms per year for different distributions of the das-random numbers

P(1)	Probability for a noise pre-alarm within one test suite	Average number of noise alarms per year
0.500	0.0162	0.0047
0.495 or 0.505	0.0187	0.0072
0.490 or 0.510	0.0292	0.027
0.485 or 0.515	0.0794	0.52
0.480 or 0.520	0.2954	21.1
0.475 or 0.525	0.7670	
0.470 or 0.530	0.9912	

- 430 Compared to the online test procedure proposed in the preceding example, the situation is more favourable.
- 431 Under the null hypothesis (ideal RNG), $P(C_j > 26.75) \approx 0.03$. Here, the χ^2 distribution still has "mass" and the relative error is low.
- 432 Decision rule (ii), too, does not depend on the occurrence of a single, very rare event but on a several events that, taken individually, are by no means rare. The small weight factor β ensures this.
- 433 If the distribution of the digitised noise sequence deviates from the null hypothesis, the distribution function of the test variable can be estimated by means of a stochastic simulation. Here, a pseudo-random number generator is used to generate standard random numbers, i.e., pseudo-random numbers that are uniformly distributed on the interval $[0,1)$. Typically, one uses a linear congruential generator or a linear feedback shift register, since they are very fast and have good statistical properties, and unpredictability of the pseudo-random numbers is irrelevant here ([Schi09a], subsection 2.4.3). From the standard random numbers, one computes a large number of sequences (e.g., 10^6 of $4 \cdot 128$ pseudo-random bits) according to the desired distribution. To each sequence, the χ^2 test is applied. For the distributions taken into account in Table 17, stochastic simulations delivered the following probabilities that the test variable is > 26.75 : 0.0299 (null hypothesis), 0.0303, 0.0331, 0.0371, 0.0416, 0.0526 and 0.0656 (order as in Table 17).
- 434 Under the above assumptions, the basic test variables C_1, C_2, \dots can be interpreted as the realisation of independent random variables. Decision rules (i) and (ii) define a homogeneous Markov chain on the finite state space
- $$\Omega = \{(2^{-6}k, i) \mid k \in N, 2^{-6}k \in [13.0, 17.0], 0 \leq i \leq 2\} \cup \{\omega\},$$

where ω is an absorbing state. The state (v, i) is reached if the history variable assumes the value v and the last $i \leq 2$ test variables were greater than 26.75. The absorbing state ω corresponds to a preliminary noise alarm being triggered (see also [Schi01]).

- 435 Depending on the application and on the consequence of a noise alarm the expected number of noise alarms for the range of permitted probabilities $P(1)$ might be too large. We point out that by selecting other parameter sets the online test procedure can be made more or less restrictive (see also [Schi01]).
- 436 Under the assumption that the χ^2 test is appropriate for the given noise source, functional requirement PTG.2.3 (also PTG.3.3) is fulfilled, and also PTG.2.4 and PTG.2.5 (also PTG.3.4 and PTG.3.5).
- 437 Suppose that a total failure of the noise source results in constant output sequences. If the last 220 bits of a sample are constantly 0 or constantly 1, this implies $c_j \geq 269.5$, which triggers a noise alarm due to decision rule (iii). Note that the current basic test may not necessarily detect a total failure if it occurs later. However, a noise alarm is triggered at the latest by the subsequent basic test. At this point in time, however, no internal random number has left the FIFO, which has been used to fill up the FIFO after the total failure occurred. Thus, functional requirement PTG.2.2 (also PTG.3.2) also is fulfilled.

5.6. Examples of RNG designs

5.6.1. PTRNG with two noisy diodes

Example Basic RNG Design with noisy diodes

- 438 In this section we discuss an example of a PTRNG (cf. [KiSc08] for details).
- 439 The random source of the RNG consists of two equal noisy diodes. For example, Zener diodes have a reverse avalanche effect (depending on diode type 3–4 Volts or about 10 V) and produce more than 1mV noisy voltage on about 10 MHz. The Flicker Noise in Schottky diodes is associated with static current flow in both resistive and depletion regions (caused by traps due to crystal defects and contaminants, which randomly capture and release carriers).
- 440 Both diodes provide symmetric input to an operational amplifier to amplify the difference of noise voltages. The output of the operational amplifier is provided to a Schmitt trigger, where the mean voltage of the amplifier meets the threshold of the Schmitt trigger. The output signal of the Schmitt trigger consists of zero and one signal of random length. This signal is latched to the digitised random signal with a clock, which should be at least 20 times slower than the output signal of the Schmitt trigger.
- 441 The tot test separately checks the generation of noisy voltage of each diode. The online test shall check the quality of the digitised noise signal by suitable statistical tests.
- 442 Figure 9 illustrates the basic design.

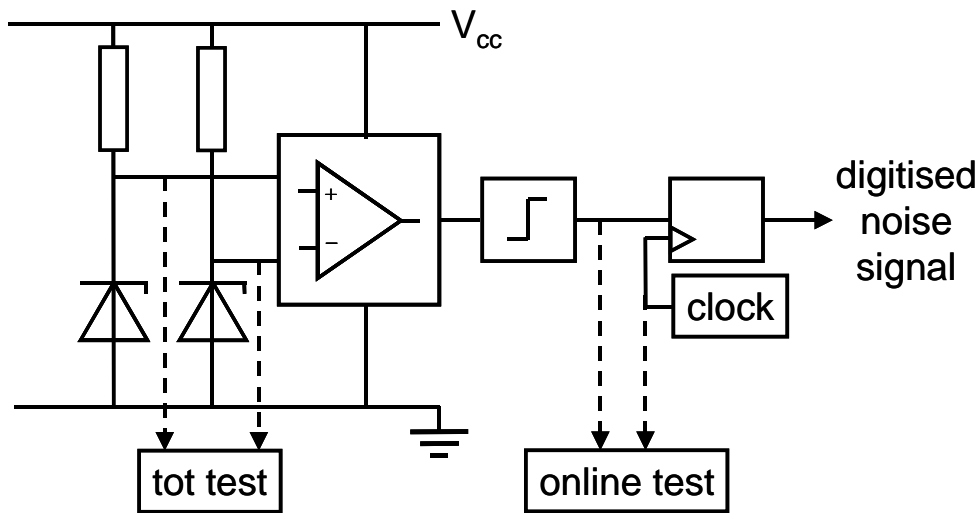


Figure 9: Basic design of RNG with noisy diodes

443 The circuit for AC coupling, the negative feedback to the operational amplifier, the stabilization of the power supply and the temperature compensating effects are not shown in this figure. A drift of the noisy voltages or the operational amplifier output results in impulses that are too long or too small, causing a biased digitised noise signals. Therefore, the digitised random signal shall be passed to a Neumann/Peres unbiasing control. Clearly, long-term aging effects may be neglected here.

Variant of RNG Design with noise diodes

444 The advanced variant of the basic design outputs the number of Schmitt trigger impulses (caused by 0-1-crossings) modulo 2 as the digitised noise signal.

445 Figure 10 illustrates the advanced design.

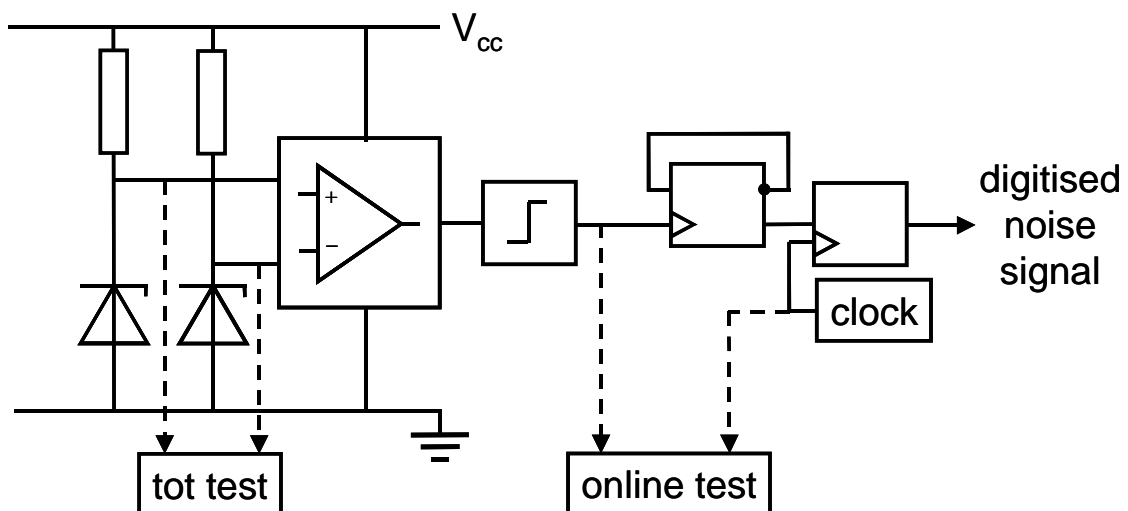
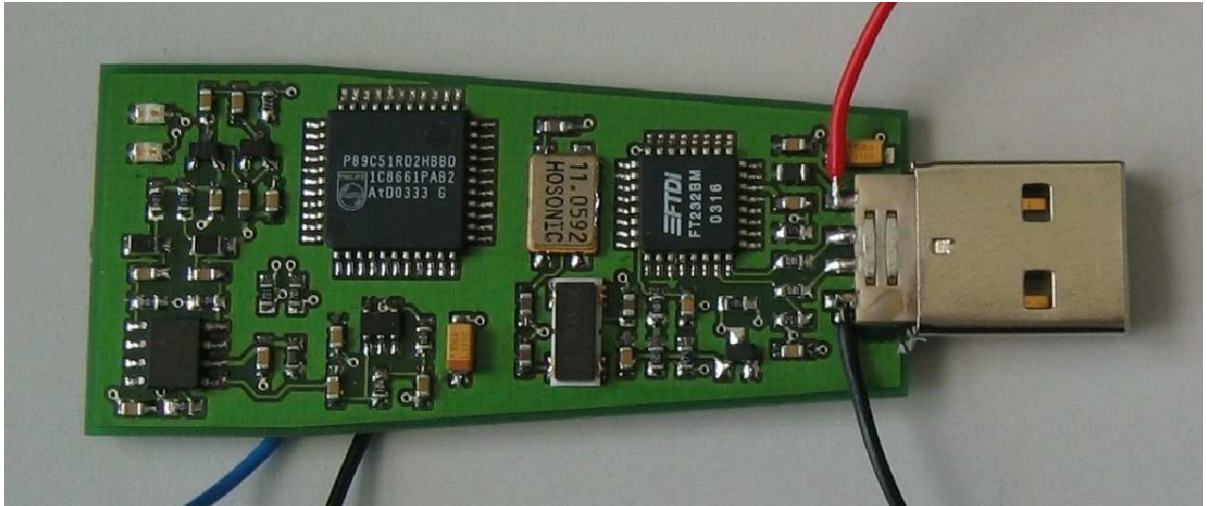


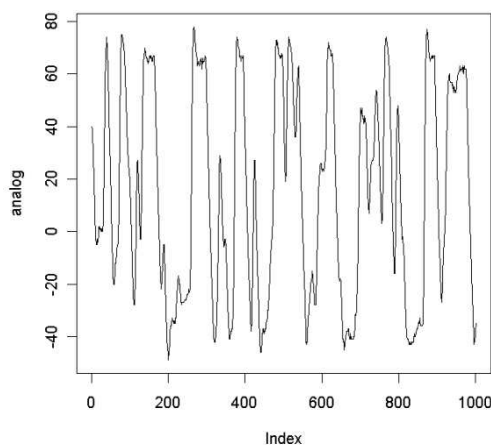
Figure 10: Variant of the basic design of RNG with noisy diodes

- 446 Each impulse of the Schmitt trigger inverts the signal that is latched by the clock. The quality of the digitised noise signal depends on the randomness of the numbers of these impulses. Unlike the basic design described above, it is not relevant whether the intervals between 0-1 and 1-0-crossings, or between 1-0 and 0-1-crossings, are identically distributed.

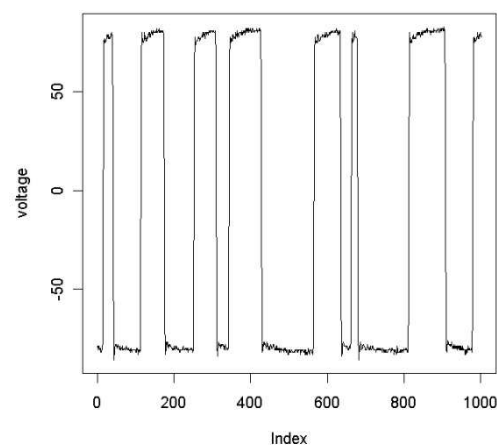
We provide some exemplary measurements of a similar design (cf. [KiSc08]).



- 447 The output of the operational amplifier within time intervals of 1ns gives diagrams like the following (resolution: 8 bits).

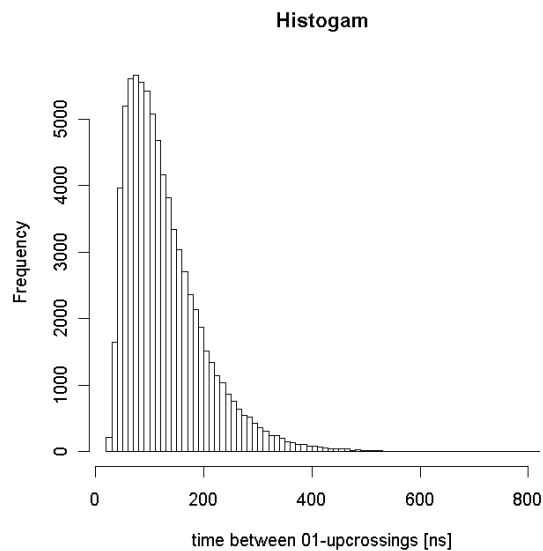


Difference of noisy voltages of the operational amplifier (low amplification)

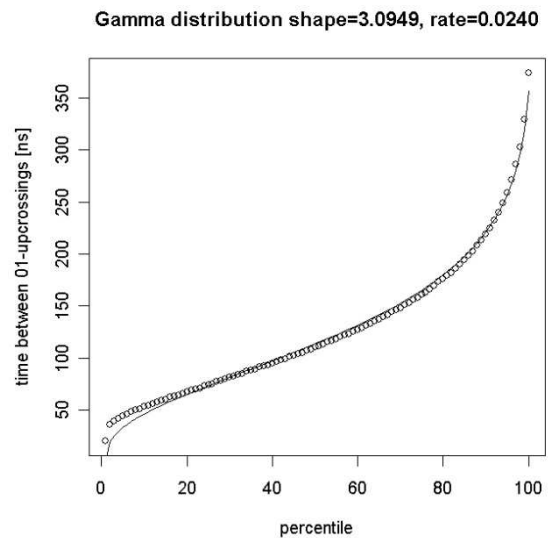


Output signal of the operational amplifier (maximum amplification)

- 448 The distribution of the time intervals between successive 01-upcrossings may be illustrated by a histogram or by percentiles of a distribution. (Note that these diagrams belong to different measurements).

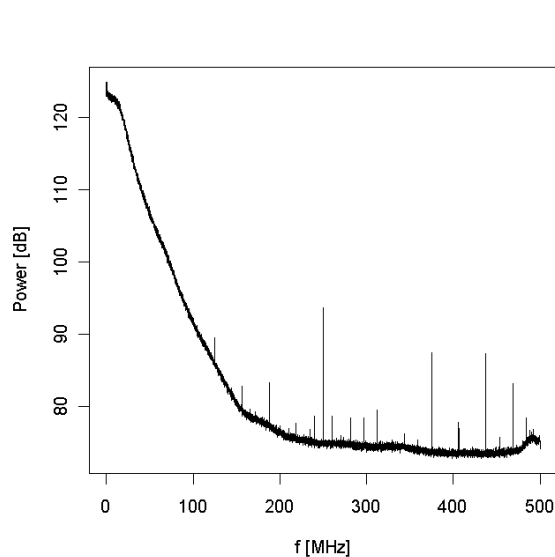


This histogram shows the empirical distribution of the time intervals between successive 01-upcrossings (in ns).

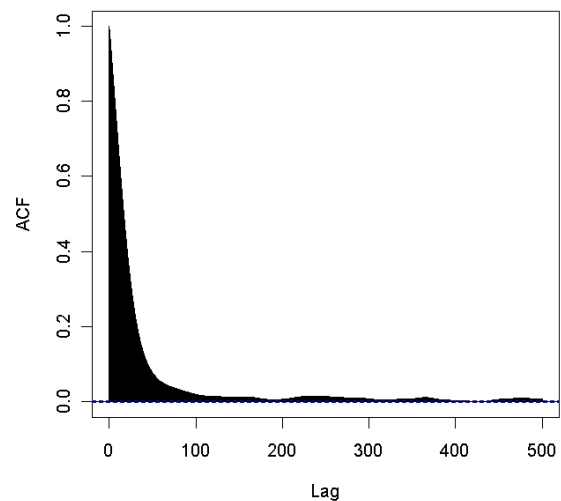


This graphic shows the percentiles of the gamma distribution versus the observed percentiles of the time intervals between successive 01-upcrossings (in ns).

- 449 These measurements also allow calculation of the power spectrum and calculation of the auto-correlation of the signals.



The mean power spectrum of the output of the amplifier (low amplification)



Autocorrelation of the difference of noise voltages (maximum amplification)

- 450 [KiSc08] develops and analyses a stochastic model that fits to this PTRNG. This stochastic model allows the estimation of a lower bound for the average entropy per internal random number. One key observation is that under mild assumptions, the internal random numbers may be viewed as realisations of a stationary process. In particular, autocovariance and autocorrelation can be calculated. Interestingly, the same type of stochastic model fits several other RNG designs, too (the stochastic processes having different distributions, of course). The interested reader is referred to [KiSc08] for details.
- 451 The security architecture of this PTRNG should describe and implement protection against effects on the power consumption and self-tests (cf. self-protection).

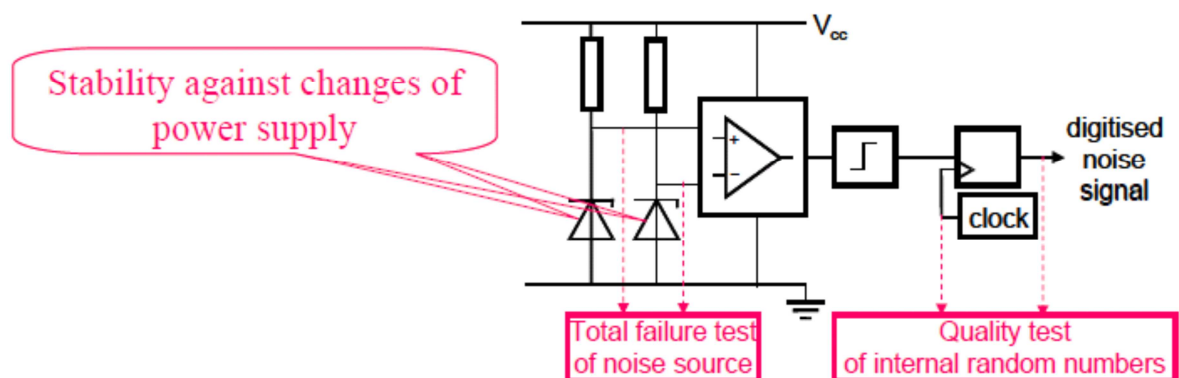


Figure 11: Examples of self-protection in PTRNG based on noise diodes

- 452 Effective online tests should be tailored to the stochastic model of the noise source. Reference [KiSc08] analyses properties that effective online tests should have.

5.6.2. Examples of DRNGs

Example 39: NIST SP 800-90 DRBG Mechanisms based on Hash Functions

- 453 NIST Special Publication 800-90 “Recommendation for Random Number Generation Using Deterministic Random Bit Generators” [NIST800-90] specifies mechanisms for the generation of random bits using deterministic methods. The methods discussed are either based on hash functions, block ciphers, or problems from number theory. We give a brief description of NIST recommended DRNGs and hybrid DRNGs that are based on NIST-approved hash functions and HMAC. The reader may refer to [NIST800-90] for details.
- 454 [NIST800-90] comprises a detailed description of the interfaces and the functions of the DRNG. Figure 12, which is from [NIST800-90], chapter 7, illustrates the generic design of the RBG. The entropy input and the seed shall be kept secret. The secrecy of this information provides the basis for the security of the random bit generator (RBG). The entropy input shall at least provide the amount of entropy requested by the Deterministic Random Bit Generator (DRBG) mechanism given by the parameter *security_strength*. Other data fed into the DRNG as personalisation string, nonce, and additional input may or may not be required to be kept secret

by a consuming application; however, the security of the RBG itself does not rely on the secrecy of this information.

455 [NIST800-90], chapter 10.1, defines two mechanisms that are based on the hash functions SHA-1, SHA-224, SHA-256, SHA-384 and SHA-512⁴³:

- Hash_DRBG using one of these hash functions,
- HMAC_DRBG using one of these hash functions and the HMAC based on these hash functions.

The hash functions are used directly and in the form of the functions *Hash_df* (cf. [NIST800-90], section 10.4.1) and *Hashgen* (cf. [NIST800-90], section 10.1.1.4). We denote the output length of the hash function as *hash_outlen*, which is 160 for SHA-1 and *x* for SHA-*x* for the other hash functions.

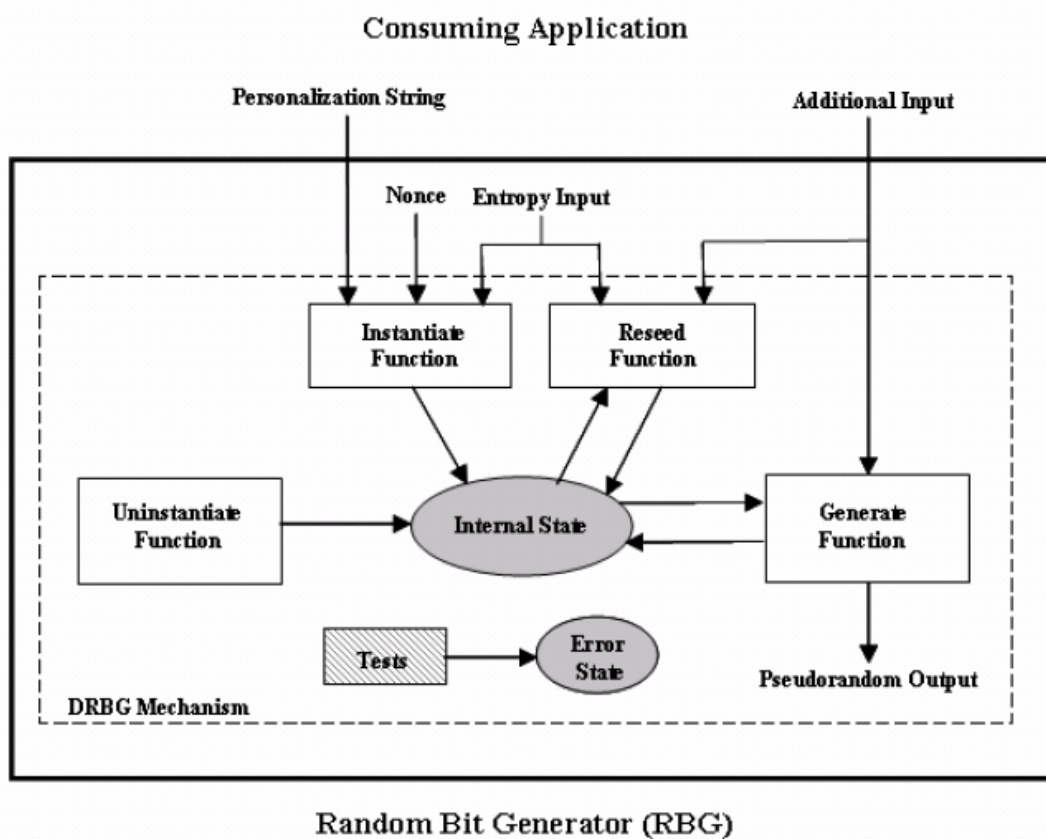


Figure 12: RGB Functional model defined in [NIST800-90]

456 [NIST800-90] describes these DRNGs with parameters depending on the hash function used (cf. [NIST800-90], Table 2 in section 10.1 for the limitation of these parameters). The parameter *seedlen* equals 440 for the hash functions SHA-1, SHA-224, SHA-256 and equals 888 for SHA-

⁴³ Note that since SHA-224 is based on SHA-256, and SHA-384 is based on SHA-512, there is no efficiency benefit for using SHA-224 or SHA-384.

384 and SHA-512. Note the length of the entropy input, the personalisation string and the additional input shall be $\leq 2^{35}$. The number of requests since last reseeding⁴⁴ is counted by *reseed_counter*⁴⁵. The *reseed_counter* is bounded by the parameter *reseed_interval*, which shall be $\leq 2^{48}$. The length of the binary output sequences per request shall be $\leq 2^{19}$.

457 [NIST800-90] describes the instantiation function, the reseed function and the generate function of the DRNG. The reseed function and the generate function have the same internal state but different input lengths, and the reseed function does not generate output. Therefore, it is easier to use two algorithmic descriptions of the DRNG introduced in section 2.2.3 on page 25:

- the 6-tuple $(S, I, R, \varphi, \psi, p_A)$ for the “normal operation” using the generate function, and
- the 4-tuple $(S, \hat{I}, \hat{\varphi}, p_A)$ for the reseeding operation.

A complete model for both operations is also possible by combining the state transition functions φ and φ^\wedge in one more complex function and an empty string output in case of reseeding. p_A denotes the distribution of the internal state after instantiation or reseeding of the DRNG.

458 The internal state of the Hash_DRBG consists of a value V that is updated during each call of the DRNG, a constant C that depends on the seed, and a counter *reseed_counter* that indicates the number of requests for output since new entropy input was obtained during instantiation or reseeding. Both algorithmic descriptions use the same internal state S defined as

$$S = \{0,1\}^{\text{seedlen}} \times \{0,1\}^{\text{seedlen}} \times \{0,1\}^{48}$$

the set of internal states, where we write $s = (v, c, z)$, v stands for V , c for C and z for *reseed_counter* in [NIST800-90].

459 The **instantiation function** generates the initial internal state $s_0 = (v_0, s_0, r_0)$ by setting the initial value r_0 of the counter *reseed_counter* to 1 and calculates the values v_0 and c_0 using an entropy input string *entropy_input* obtained from the source of entropy input, a value *nonce* as a time-varying value that has at most a negligible chance of repeating, and an optional *personalization_string* (cf. [NIST800-90], section 10.1.1.2). The derivation function *Hash_df* calculates from an input bit string *input_string* an output bit string *requested_bits* of length *no_of_bits_to_return* by repeated application of the hash function until *temp* contains at least *no_of_bits_to_return* bits, where the initial *temp* string is empty, as follows:

temp := \emptyset , *counter* := 1

⁴⁴ According to our definition, it is indeed a seed-update as will be shown later. In this example, we use the notation from [NIST800-90].

⁴⁵ More precisely, this counter is set to 1 by the instantiation function and by the reseeding function.

$$len := \left\lceil \frac{no_of_bits_to_return}{hash_outlen} \right\rceil$$

For $i=1$ to len do

$$temp := temp \parallel Hash(counter \parallel no_of_bits_to_return \parallel input_string)$$

$$counter := counter + 1$$

$requested_bits :=$ Leftmost $(no_of_bits_to_return)$ bits of $temp$

$$Hash_df(input_string, no_of_bits_to_return) := requested_bits$$

One denotes the instantiation function using $Hash_df$ like this:

$$v_0 = Hash_df(entropy_input \parallel nonce \parallel personalization_string, seedlen)$$

$$c_0 = Hash_df(0x00 \parallel v_0, seedlen)$$

$$r_0 = 1$$

- 460 The instantiation function generates internal states with an initial distribution p_0 depending on the input. The minimum entropy and minimum length of entropy input sequence provided for instantiation are denoted as parameter *security_strength* in [NIST800-90].
- 461 The **generate function** updates the internal state and generates the output of a requested length. In terms of the algorithmic description it may be described by the 6-tuple $(S, I, R, \phi, \psi, p_A)$ with

- $I, I = \bigcup_{i=0}^{2^{35}} \{0,1\}^i \times \overline{1, 2^{19}}$, is the input alphabet, where $i = (e, l)$, e is the input sequence used for the state transition function and the output function (denoted as *additional_input* in [NIST800-90]) and l the length of the requested output (denoted as *outlen* in [NIST800-90]),

- $R, R = \bigcup_{i=0}^{2^{48}} \{0,1\}^i$, the output alphabet is the set of binary sequences of length $\leq 2^{48}$, and

- p_A is the distribution of the internal state after instantiation or reseeding.

- 462 The DRNG may or may not support additional input for the generate function, the state transition function and the output function. The additional input may be publicly known or an entropy input for a hybrid DRNG (section 2.2.3). If the DRNG allows additional input for the generate function, the state transition function, and the output function, the DRNG uses pre-computation of an intermediate value v' of the first part of internal state, as follows:

$$v'_n = (v_n + Hash(0x00 \parallel v_n \parallel e_n)) \bmod 2^{seedlen}$$

If the DRNG does not support additional input, we set $v'_n = v_n$ for simplicity of description.

463 Pre-computation prevents malicious control of the internal state and the output through the additional input if the value v is unknown and makes this control very hard even if the value v is known, because the hash function is a one-way function.

464 The **state transition function** $\varphi : S \times I \rightarrow S$ calculates the next initial state $s_{n+1} = (v_{n+1}, c_{n+1}, z_{n+1}) = \varphi(v'_n, c_n, z_n)$ as follows:

$$v_{n+1} = (v'_n + \text{Hash}(0x02 \parallel v'_n) + c_n + z_n) \bmod 2^{\text{seedlen}}$$

$$s_{n+1} = s_n$$

$$z_{n+1} = z_n + 1$$

465 The **output function** $\text{Hashgen } \psi : S \times I \rightarrow R$ calculates output of length l from input $i = (e, l)$ from the intermediate value v' as follows:

$$r_{n+1} = \psi(v'_n, l) = \text{Hashgen}(l, v'_n)$$

The function *Hashgen* generates the binary string *output* from the value *input* for a given length *requested_no_of_bits* by repeated application of the hash function until *temp* contains at least as follows:

$temp := 0, data := input$

$$len = \left\lceil \frac{\text{requested_no_of_bits}}{\text{outlen}} \right\rceil$$

For $i=1$ to len do

$temp := temp \parallel \text{Hash}(data)$

$data := (data + 1) \bmod 2^{\text{seedlen}}$

$output := \text{Leftmost}(temp) \text{ bits of } temp$

$\text{Hashgen}(\text{requested_no_of_bits}, input) := output$

466 Note that both the part of the state transition function that completes v_{n+1} and the output function are based on a hash function and behave as a random mapping. This provides forward secrecy as assurance that subsequent (future) values cannot be determined from current or previous output values.

467 The output function $\psi(v'_n, l) = \text{Hashgen}(l, v'_n)$ is a cryptographic one-way function with respect to v'_n (observing the output l is obviously known). The value v'_n will contain sufficient entropy to prevent successful guessing. The attacker cannot determine v'_n . Because v'_k is prerequisite for calculation of the output, the internal state s_n cannot be determined from current or future output values. The DRNG provides backward secrecy.

468 The function $v_{n+1} = \pi_{1, seedlen}(\varphi(v'_n, c_n, z_n))$ is a cryptographic one-way function, as well. For $k < n$, an attacker can calculate s_k and z_k , but he cannot calculate previous values v'_k or v_k if the internal state s_n and the possible additional inputs are known. The DRNG provides enhanced backward secrecy.

469 The **reseeding function** generates a new internal state $s_{n+1} = (v_{n+1}, c_{n+1}, r_{n+1})$ by setting the counter *reseed_counter* r_{n+1} to 1 and calculating the new values v_{n+1} and c_{n+1} using the current value v_n , an entropy input string *entropy_input* obtained from the entropy source, and an optional *additional_input* (cf. [NIST800-90], section 10.1.1.3). The minimum length of the entropy input string *entropy_input* is given by the parameter *security_strength*.

$$v_{n+1} = Hash_df(0x01 \| v_n \| entropy_input \| additional_input, seedlen)$$

$$c_{n+1} = Hash_df(0x00 \| v_{n+1}, seedlen)$$

$$r_{n+1} = 1$$

470 The call of the reseeding function is enforced by

- the consuming application setting the *prediction_resistance_flag*, and
- the reseed counter *reseed_counter* when reaching the maximum number of requests between reseeding *reseed_interval*.

The reseeding function provides the requested amount of entropy if the entropy input is independent of the current internal state. If the internal state is compromised, the secrecy of the internal state is re-established. Therefore, the DRNG ensures enhanced forward secrecy, i.e., the assurance that subsequent (future) values of a DRNG cannot be determined from the current internal state, current or previous output values, on demand of the consuming application, or automatically as configured through the parameter *reseed_interval*.

471 Compared with the generate function, the reseeding function uses a longer input sequence (without any length of output), does not use c_n , and no output is generated. Therefore, the algorithmic description of the reseeding function may be described in the form of the 4-tuple $(S, \hat{I}, \hat{\phi}, p_A)$:

- the input alphabet is $\hat{I} = \{0,1\}^{35} \times \{0,1\}^{35}$,
 $i = (entropy_input, additional_input)$,
- the state transition $\hat{\phi}: \{0,1\}^{seedlen} \times \{0,1\}^{48} \rightarrow \{0,1\}^{seedlen} \times \{0,1\}^{seedlen} \times \{0,1\}^{48}$
 $s_{n+1} = (v_{n+1}, c_{n+1}, r_{n+1}) = \hat{\phi}(v_n, i_n)$

- 472 The distribution p_A depends on the entropy of the current internal state value v_n , the entropy input, and on the reseeding function itself. The *additional_input* might contain some entropy if it is kept secret, but the security of the DRNG does not depend on this.

Example 40: ANSI X9.17 DRNG

- 473 The ANSI X9.17 DRNG is defined as follows (cf. [Schn06]): Let $TDES(x, y)$ denote the triple DES encryption function with key x and plain text y , $TDES^{-1}(x, z)$ the decryption function for the key x and cipher text z , and k a secret triple-DES key (112 effective bits), which is generated when the DRNG is instantiated. Further, t_i denotes time stamps when random numbers are requested, $i = 1, 2, \dots$, and z_i denotes the secret internal state at time t_i (before the requested random numbers are generated), $z_i \in \{0, 1\}^{64}$, $i = 1, 2, \dots$. Finally, s_1 is the initial state, and r_i the output at time t_i , $i = 1, 2, \dots$, $R = \{0, 1\}^{64}$. Each time output is requested, the following steps are executed:

$$T_i = TDES(k, t_i)$$

$$r_i = TDES(k, T_i \oplus z_i)$$

$$z_{i+1} = TDES(k, T_i \oplus r_i)$$

- 474 The 6-tuple $(S, I, R, \varphi, \psi, p_A)$ is defined as:

- set of internal states: $S = K \times Z$, $K = \{0, 1\}^{112}$, $Z = \{0, 1\}^{64}$,
- publicly known input: $I = \{0, 1\}^{64}$, $|I|$ depends on the time base, e.g. for PC typically $|I| = 2^{32}$,
- output alphabet: $R = \{0, 1\}^{64}$
- state transition function: $\varphi : S \times I \rightarrow S$,
 $(k_{i+1}, z_{i+1}) = \varphi(k_i, z_i) =$
 $= (k, TDES(k, TDES(k, t_i) \oplus TDES(k, TDES(k, t_i) \oplus z_i)))$ (65)
- output function: $\psi : S \times I \rightarrow R$,
 $r_i = \psi(k, z_i) = TDES(k, TDES(k, t_i) \oplus z_i)$ (66)
- distribution of the initial internal state: p_A uniform distribution over S .

- 475 The distribution p_A is not explicitly described in [Schn06], but k and s_1 shall be secret. The uniform distribution provides secrets with maximum entropy.

476 The internal state contains 176 bits. The internal state of this DRNG meets the necessary entropy condition according to Table 13 to resist high attack potential, but does not meet the recommended length of 200 bits. Obviously, for any given k , the state transition function holds the internal entropy because it is a permutation; i.e., for $T_i = TDES(k, t_i)$ one gets:

$$k_{i+1} = k_i = k, \quad z_i = TDES^{-1}(k, TDES^{-1}(k, z_{i+1}) \oplus T_i) \oplus T_i \quad (67)$$

477 Assume that t_i are publicly known inputs or at least easy to guess by observing the DRNG operation. The strength of forward and backward secrecy appears to require cryptanalysis of the triple-DES. This DRNG neither ensures enhanced backward secrecy nor enhanced forward secrecy.

5.6.3. NPTRNG

Example 41: Linux NPTRNG [GuPR06]

478 The Linux operating system includes two RNGs as part of the kernel:

- the non-physical true random number generator `/dev/random/`
- the non-physical hybrid deterministic random number generator `/dev/urandom/`

479 Externally-visible (user-space) interfaces (devices) `/dev/random/` and `/dev/urandom/` are marked grey, while interfaces that are only accessible inside the kernel are displayed green. Both devices use a common DRNG `input_pool` and additionally implement separate DRNGs: `/dev/random/`, the DRNG `blocking_pool`, and `/dev/urandom/` the DRNG `nonblocking_pool`. While there must be sufficient entropy in the `input_pool` and the `blocking_pool` before one can use `/dev/random/`, `/dev/urandom/` also generates output when the estimated entropy ϵ' of the `input_pool` (and therefore `nonblocking_pool`) has reached zero. In other words, `/dev/random/` needs the `input_pool` to be seeded continuously and it does not output more bytes than the entropy that has (probably) been harvested by the sources of randomness. *This property of `/dev/random/` is defined to be the characteristic of an NPTRNG. In contrast to this feature, the non-blocking output of random numbers (i.e., regardless of an entropy estimate of the seeding input) makes the `/dev/urandom/` a hybrid DRNG.*

480 There are three different registers holding random values: the `input_pool`, `blocking_pool` and `nonblocking_pool`. Then, entropy is extracted from one of the pools by using `extract_buf`. `Extract_buf` involves a one-way function (SHA-1) that provides the desired output and simultaneously updates the buffer from which the bytes were read. This “cryptographically active” function is coloured red.

481 The functional design of the Linux random number generator (Linux-RNG) is shown in the following figure:

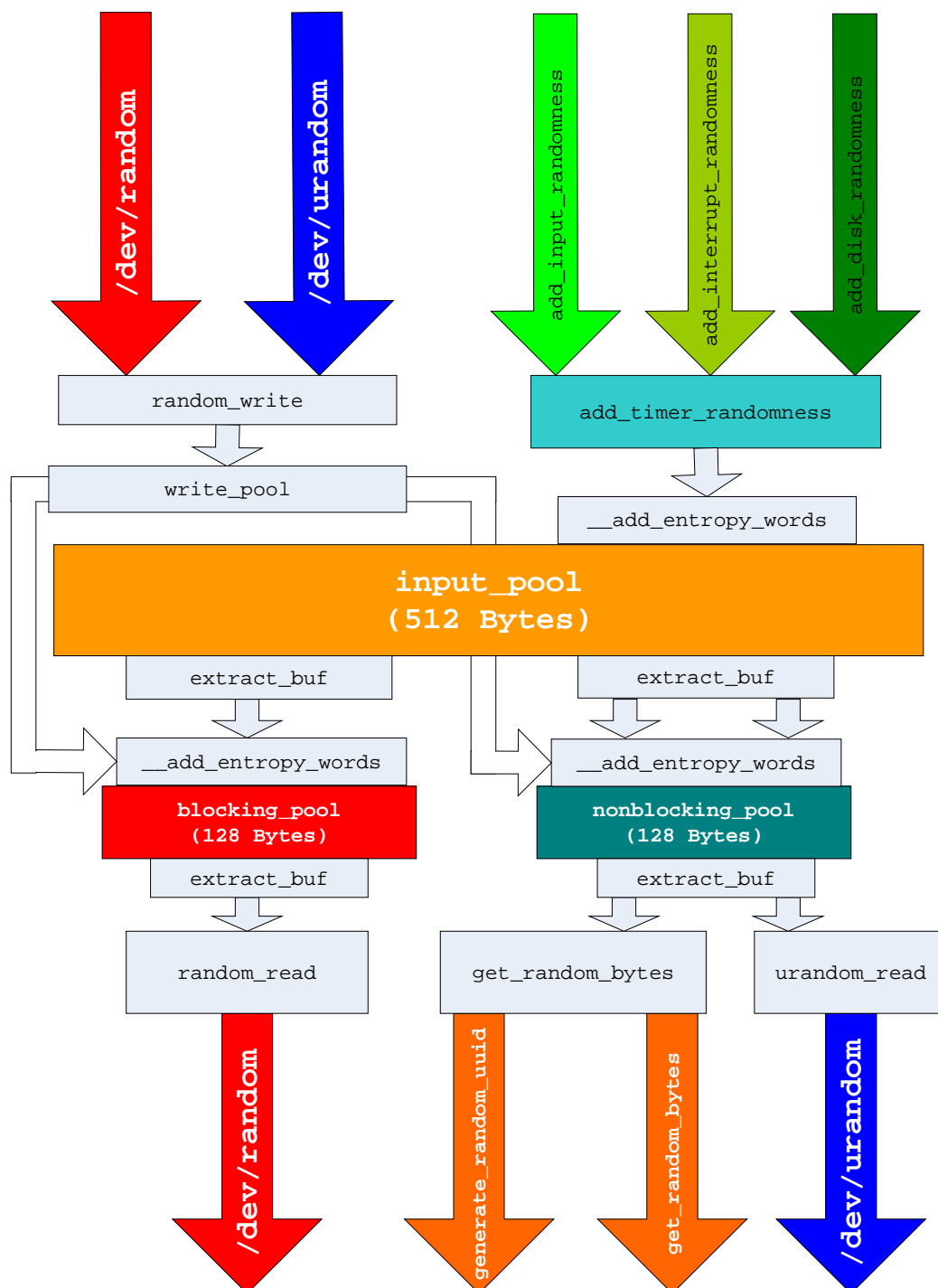


Figure 13: Functional design of the Linux NPTRNG

- 482 The Linux-RNG uses different non-physical sources of entropy (green), that are either dependent on user actions or internal system tasks. Every single event (e.g., keyboard and mouse actions; hard disk, CD or USB access) is mapped to a specific value that is coded in 32 bits. This “event” is then passed to `add_timer_randomness` where it is “tagged” with a (96-bit) timestamp composed of the 64-bit running number of processor cycles and a (32-bit) counter that is increased every 4 ms. The Linux term for this counter is “jiffies”. The value of this

counter is then used to estimate the entropy that the event carries: `add_timer_randomness`, therefore, compares (subtracts) the “jiffies” (i.e., a combination of the number of processor cycles encoded as 64-bit number and the number of timer interrupts since power-up of the operating systems) of two consecutive events ($t_n - t_{n-1}$). Moreover a “history” of the 1st, 2nd and 3rd order deltas (where the 2nd order delta is defined to be the previous 1st order delta subtracted from the current 1st order delta etc.) is saved:

$$\min\{(d_n = t_n - t_{n-1}), (d_n^2 = d_n - d_{n-1}), (d_n^3 = d_n^2 - d_{n-1}^2)\}$$

- 483 Using this method, it is possible to detect 1st, 2nd and 3rd order dependencies in the timing of the events. The smallest delta-value (as an additional security feature, limited to 11) is then set to be the estimated entropy ϵ' of the actual word. Note that the data written into the primary entropy pool is the complete Timestamp concatenated with the event. The four words are: `[cycles LSW] · [cycles MSW] · [jiffies] · [event]`.
- 484 One now can show that the entropy is estimated in a very conservative way: first, a maximum of 11 out of 128 bits must carry entropy; and second, the entropy estimate ϵ' is based on a single part of the data written to the pool. Its value changes much more slowly than the word that is supposed to carry the most entropy, namely `[cycles LSW]`. For example, while “jiffies” is increased by 1, e.g., a processor clocked at 1.8 GHz performs more than 7.2 million cycles. This leads to 22 “undefined” bits in the sampled counter.
- 485 A deep analysis of the Linux-RNG showed that each chunk of data written to the pool (event plus time stamp) has Min-entropy larger than 9 bit. At the same time, the average of the entropy estimates ϵ' by the kernel was below 1 bit.
- 486 The Linux-RNG of Linux kernel 2.6.21.5 is assessed as an appropriate entropy source [TR-02102].

6. Literature

- [AIS20] BSI: Application Notes and Interpretation of the Scheme (AIS) 20 – Functionality classes and evaluation methodology for deterministic random number generators, Version 1 (02.12.1999), English translation.
https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Zertifizierung/Interpretationen/ais20e_pdf.pdf?__blob=publicationFile
- [AIS20An] W. Schindler: AIS 20: Functionality classes and evaluation methodology for deterministic random number generators, Version 2.0 (02.12.1999), Mathematical-technical reference of [AIS20], English translation.
https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Zertifizierung/Interpretationen/ais20e_pdf.pdf?__blob=publicationFile
- [AIS31] BSI: Application Notes and Interpretation of the Scheme (AIS) 31 – Functionality Classes and Evaluation Methodology for Physical Random Number Generators, Version 1 (25.09.2001), English translation.
https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Zertifizierung/Interpretationen/ais31e_pdf.pdf?__blob=publicationFile
- [AIS31An] W. Killmann, W. Schindler: A Proposal for: Functionality Classes and Evaluation Methodology for True (Physical) Random Number Generators, Version 3.1 (25.09.2001), Mathematical-technical reference of [AIS31], English translation.
https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Zertifizierung/Interpretationen/trngk31e_pdf.pdf?__blob=publicationFile
- [AIS2031Stat] Implementation of test procedure A and test procedure B of this document;
https://www.bsi.bund.de/cae/servlet/contentblob/478136/publicationFile/30232/testsuite_zip.zip
- [AIS34] BSI: Application Notes and Interpretation of the Scheme (AIS) 34 – Evaluation Methodology for CC Assurance Classes for EAL5+, Version 1.4 Draft, 14.08.2008.
- [BuLu08] M. Bucci, R. Luzzi: Fully Digital Random Bit Generators for Cryptographic Applications, IEEE Transactions on Circuits and Systems I: Regular Papers, Vol. 5, Issue 3 (2008), pp. 861-875.
- [Calu02] C. S. Calude: Information and Randomness, An Algorithmic Perspective, 2nd ed., Springer, 2002.
- [CFPZ09] C. Chevalier, P.-A. Fouque, D. Pointcheval, S. Zimmer: Optimal Randomness Extraction from a Diffie-Hellmann Element. In A. Joux (ed.): Advances in Cryptology – Eurocrypt 2009, Springer, LNC 5479, 2009, pp. 572-589.
- [CCV31_1] Common Criteria for Information Technology Security Evaluation, Part 1: Introduction and General Model, Version 3.1, Revision 3 Final, July 2009, CCMB-2009-07-001.
- [CCV31_2] Common Criteria for Information Technology Security Evaluation, Part 2: Security Functional Requirements. Version 3.1, Revision 3 Final, July 2009, CCMB-07-002.
- [CCV31_3] Common Criteria for Information Technology Security Evaluation, Part 3: Security Assurance Requirements. Version 3.1, Revision 3 Final, July 2009, CCMB-07-003.

- [CEM] Common Methodology for Information Technology Security Evaluation (CEM): Evaluation Methodology, Version 3.1, Revision 3 Final, July 2009, CCMB-2009-07-004.
- [CCSDIC] Common Criteria Supporting Document, Mandatory Technical Document, The Application of CC to Integrated Circuits, Version 3.0, Revision 1, March 2009, CCDB-2009-03-002.
- [CoNa98] J.S. Coron and D. Naccache: An Accurate Evaluation of Maurer's Universal Test, in: S. Tavares and H. Meijer (eds.): Selected Areas in Cryptography '98, SAC '98, Springer, Lecture Notes in Computer Science, Vol. 1556, Berlin, 1999, pp. 57-71.
- [Coro99] J.S. Coron: On the Security of Random Sources, Gemplus Corporate Product R&D Division, Technical Report IT02-1998; also in: H. Imai and Y. Zheng (eds.): Public Key Cryptography, Second International Workshop on Practice and Theory in Public Key Cryptography, PKC'99, Springer, Lecture Notes in Computer Science, Vol. 1560, Berlin, 1999, pp. 29-42.
- [Craw05] M.J. Crawley: Statistics: An Introduction using R, John Wiley & Sons Inc., 2005.
- [Devr86] L. Devroye: Non-Uniform Random Variate Generation, Springer, Berlin, 1986.
- [DoGP07] L. Dorrendorf, Z. Guttermann, B. Pinkas: Cryptanalysis of the Random Number Generator of the Windows Operating System, November 4, 2007, eprint server of IACR, <http://eprint.iacr.org/2007/419.pdf>.
- [DaRo87] W. B. Davenport, Jr., W. L. Root: An Introduction to the Theory of Random Signals and Noise, IEEE Press, 1987.
- [FI140-1] NIST: FIPS PUB 140-1 (January 11, 1994), Security Requirements for Cryptographic Modules.
- [FI140-2] NIST: FIPS PUB 140-2 (1999), Security Requirements for Cryptographic Modules.
- [FI186] NIST: FIPS PUB 186-2 (October 2001), Specifications for the Digital Signature Standard (DSS), with Change Notice 1.
- [FI186-3] NIST: FIPS PUB 186-3 (June 2009/March 2006 Draft), Specifications for the Digital Signature Standard (DSS).
- [Golo64] S.W. Golomb: Random permutations, Bulletin of the American Mathematical Society, 1964, Vol. 70, No. 6.
- [FI0d89] P. Flajolet, A.M. Odlyzko: Random Mapping Statistics. In: J.-J. Quisquater, J. Vandevale (eds.): Advances in Cryptology, EUROCRYPT'89, LNCS, Vol. 434, Berlin 1990, pp. 329-354.
- [GuPR06] Z. Guttermann, B. Pinkas, T. Reinman: Analysis of the Linux Random Number Generator, The Hebrew University of Jerusalem, March 6, 2006, eprint server of IACR, <http://eprint.iacr.org/2006/086.pdf>.
- [HDCM00] Handbook of discrete and combinatorial mathematics, editor-in-chief Kenneth H. Rosen, CRC Press, 2000.
- [HaLP34] G.H. Hardy, J.E. Littlewood, G. Pólya: Inequalities, Cambridge, 1934.

- [Prus06] H. Pruscha: Statistisches Methodenbuch, Verfahren, Fallstudien, Programmcodes, Springer-Verlag, Berlin, Heidelberg, 2006.
- [Intel] B. Jun, P. Kocher: The Intel® Random Number Generator, Cryptography Research, Inc., White paper prepared for Intel Corporation, April 22, 1999.
- [ISO18031] ISO/IEC 18031: Random Bit Generation, November 2005.
- [ITSEC] Information Technology Security Evaluation Criteria (ITSEC), Provisional Harmonised Criteria, Version 1.2, June 1991.
- [ITSEM] Information Technology Security Evaluation Manual (ITSEM), Provisional Harmonised Methodology, Version 1.0, September 1993.
- [JIL] Information Technology Security Evaluation Criteria Joint Interpretation Library (ITSEC JIL), Version 2.0, November 1998.
- [JeWa69] G. M. Jenkins, D. G. Watts: Spectral Analysis and its Applications, Holden-Day, San Francisco, Cambridge, London, Amsterdam, 1969.
- [Kanj95] G. K. Kanji: 100 Statistical Tests, Sage Publications, London, 1995.
- [Kill06] W. Killmann: Applying the CC V3 ADV class to hardware, presentation at 7th ICCV, 2006.
- [KiSc04] W. Killmann, W. Schindler: Evaluation Criteria for Physical Random Number Generators, presentation at 5th ICCV, 2004.
- [KiSc08] W. Killmann, W. Schindler: A Design for a Physical RNG with Robust Entropy Estimators, in: E. Oswald, P. Rohatgi (eds): Cryptographic Hardware and Embedded Systems – CHES 2008, Springer, LNCS 5154, 2008, pp. 146-163.
- [KSWH98] J. Kelsey, B. Schneier, D. Wagner, C. Hall.: Cryptanalytic Attacks on Pseudorandom Number Generators. In: S. Vaudenay (ed.): Fast Software Encryption – FSE 1998, Springer 1998, LNCS, Vol. 1372, Berlin 1998, 168-188. .
- [Maur92] U. Maurer: A Universal Statistical Test for Random Bit Generators, Journal of Cryptology, Vol. 5, No. 2, 1992, pp. 89-105.
- [MeOV97] A.J. Menezes, P.C. van Oorschot, S.A. Vanstone: Handbook of applied cryptography, CRC Press Inc., 1997.
- [MSCE06] Microsoft Windows CE Enhanced Cryptographic Provider 5.01.01603, FIPS 140-2 Documentation: Security Policy, Microsoft 6/20/2006, FIPS 140-2, Certificate No. 460.
- [Neue04] D. Neuenschwander: Probabilistic and Statistical Methods in Cryptology, An Introduction to Selected Topics, Springer LNCS 3028, Berlin, 2004.
- [PGP] PGP 8.0 for Windows User's Guide, PGP Corporation, February 2003.
- [Plia99] J. O. Pliam: The Disparity between Work and Entropy in Cryptology, February 1, 1999, eprint server of IACR, <http://eprint.iacr.org/1998/024.ps>.
- [RFC4086] D. Eastlake, S. Crocker, J. Schiller: RFC 4086 Randomness Requirements for

- Security, June 2005, [//tools.ietf.org/html/rfc4086](http://tools.ietf.org/html/rfc4086).
- [RNGVS] National Institute of Standards and Technology, Information Technology Laboratory, Computer Security Division: The Random Number Generator Validation System (RNGVS), January 31, 2005.
- [RSA] PKCS#1: RSA Encryption Standard, An RSA Laboratories Technical Note, Version 1.5, November 1, 1993.
- [Ruk2000a] A. L. Rukhin: Testing Randomness: A Suite of Statistical Procedures, Department of Mathematics and Statistics UMBS, Baltimore.
- [Ruk2000b] A. L. Rukhin: Approximate Entropy for Testing Randomness, Journal of Applied Probability, Vol. 37, No. 1 (2000) pp. 88-100.
- [SaHe06] L. Sachs, J. Hedderich: Angewandte Statistik: Methodensammlung mit R, Springer-Verlag, Berlin Heidelberg, 2006.
- [Schn96] B. Schneier: Applied Cryptography: Protocols, algorithms and source code in C, 2nd edition, John Wiley & Sons, Inc., 1996.
- [Schi01] W. Schindler: Efficient Online Tests for True Random Number Generators, in: C.K. Koc, D. Naccache, C. Paar (eds.): Cryptographic Hardware and Embedded Systems – CHES 2001, Springer, LNCS, Vol. 2162, Berlin, 2001, pp. 103-117.
- [Schi03] W. Schindler: A Stochastic Model and Its Analysis for a Physical Random Number Generator Presented at CHES 2002, in: K.G. Paterson (ed.): Cryptography and Coding – IMA 2003, Springer, LNCS, Vol. 2898, Berlin, 2003, pp. 276-289.
- [Schi09a] W. Schindler: Random Number Generators for Cryptographic Applications, in: C.K. Koc (ed.): Cryptographic Engineering, Springer, Berlin, 2009, pp. 5-23.
- [Schi09b] W. Schindler: Evaluation Criteria for Physical Random Number Generators, in: C.K. Koc (ed.): Cryptographic Engineering, Springer, Berlin, 2009, pp. 25-54.
- [ScKi02] W. Schindler, W. Killmann: Evaluation Criteria for True (Physical) Random Number Generators Used in Cryptographic Applications, in: B.S. Kaliski Jr., C.K. Koç, C. Paar (eds.): Cryptographic Hardware and Embedded Systems – CHES 2002, Springer, LNCS 2523, Berlin, 2003, pp. 431-449.
- [SP800-22] A. Rukhin et al.: A statistical test suite for random and pseudorandom number generators for cryptographic applications, NIST Special Publication 800-22 (with rev 1a, 2010revisions dated May 15, 2001).
- [SP800-90] National Institute of Standards and Technology, Information Technology Laboratory, Computer Security Division: The NIST SP 800-90 Deterministic Random Bit Generator Validation System (DRBGVS), October 30, 2007.
- [TR-02102] BSI – Technische Richtlinie kryptographische Verfahren: Empfehlungen und Schlüssellängen, TR-02102, <http://www.bsi.de/literat/tr/tr02102/BSI-TR-02102.pdf>.