A Proposal for Functionality Classes for Random Number Generators

Version 2.36 - Current intermediate document for the AIS 20/31 workshop*

Matthias Peter
Bundesamt für Sicherheit in der Informationstechnik (BSI)

Werner Schindler
Bundesamt für Sicherheit in der Informationstechnik (BSI)

June 2, 2023

*A large number of comments have already been taken into account in this intermediate document. This document is intended to support the AIS 20/31 workshop from June 5 – June 7, 2023. The focus is on the class requirements. The chapters are not aligned with each other. For these reasons, this document is not an official draft.

Abstract

This document proposes an evaluation methodology for true and deterministic random number generators. This document is updating the mathematical-technical reference of both, the AIS 20 (Funktionalitätsklassen und Evaluationsmethodologie für deterministische Zufallszahlengeneratoren. Version 3.0, May 15, 2013) and AIS 31 (BSI. Funktionalitätsklassen und Evaluationsmethodologie für physikalische Zufallszahlengeneratoren. Version 3, May 15, 2020), which define the evaluation methodology for true and deterministic random number generators in the German Common Criteria certification scheme.

Acknowledgments

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

Our special thanks to

Adrian Gabriel (Bundesamt für Sicherheit in der Informationstechnik (BSI)), Elaine Barker (National Institute of Standards and Technology (NIST)), Markus Dichtl, Jonas Fiege (Bundesamt für Sicherheit in der Informationstechnik (BSI)), Viktor Fischer (Hubert Curien Laboratory, Jean Monnet University Saint-Étienne), Aron Gohr, John Kelsey (National Institute of Standards and Technology (NIST)), Wolfgang Killmann, Kerry McKay (National Institute of Standards and Technology (NIST)), Johannes Mittmann (Bundesamt für Sicherheit in der Informationstechnik (BSI)), Stephan Müller (atsec), Jean-Renè Reinhard (Agence nationale de la sécurité des systèmes d'information (ANSSI)), Ernst Schulte-Geers (Bundesamt für Sicherheit in der Informationstechnik (BSI)), Torsten Schütze (Rohde & Schwarz), Meltem Turan (National Institute of Standards and Technology (NIST)).

List to be completed!

Contents

1 Introduction

1.1 Foreword

Random numbers are required by most cryptographic applications. Random numbers are used to generate session keys, signature parameters, nonces, challenges, blinding, and masking values (in order to prevent implementation attacks) to name just a few applications. 1

Weak Random number generators (RNGs) can decisively weaken cryptographic applications. This establishes the need for reliable and trustworthy security evaluations of RNGs. 2

In the German Common Criteria (CC) scheme for about two decades, the AIS 20 [AIS20] and AIS 31 [AIS31] have specified how RNGs shall be evaluated. Above all they define functionality classes for different types of RNGs. To be compliant to a particular functionality class, an RNG must fulfill all class-specific requirements. Furthermore, the AIS 20 [AIS20] and AIS 31 [AIS31] outline an evaluation methodology for Deterministic RNGs (DRNGs) and True RNGs (TRNGs). 3

This document is the mathematical-technical reference of both AIS 20 [AIS20] and AIS 31 [AIS31]. It is intended for developers, evaluators, and certifiers.
Note: This document itself is often loosely referenced as AIS 20 [AIS20] or AIS 31 [AIS31], respectively. Below, we follow this convention. 4

The first versions of this mathematical-technical reference were published in 1999 [AIS20An_99] and in 2001 [AIS31An_01] (mathematical-technical references to AIS 20 [AIS20] and AIS 31 [AIS31], respectively) when the CC was still new and no guidelines for the evaluation of RNGs existed. Its practical evaluation criteria have been field-tested and modernized ever since. In 2011 the mathematical-technical reference was updated [AIS2031An_11]. That is the predecessor of this document. 5

This document distinguishes between DRNGs, Physical true RNGs (PTRNGs), and Non-physical true RNGs (NPTRNGs). In Chapter 3 six functionality classes are defined (DRG.2, DRG.3, DRG.4, PTG.2, PTG.3, NTG.1). Each functionality class specifies requirements that an RNG has to fulfill to be compliant to that class. Most of these functionality classes are hierarchically ordered with regard to their requirements and thus with regard to their security strength. The overall strongest class is PTG.3. 6

Compared to its predecessor [AIS2031An_11] two functionality classes have been cancelled, namely DRG.1 and PTG.1. The definitions of the remaining functionality classes are similar to those in [AIS2031An_11] (which justifies maintaining their class names), but they are different in detail. An in-depth explanation of the differences to the specifications in [AIS2031An_11] can be found in the Subsections 3.3.1, 3.4.1, and 3.5.1. 7

1.2 Character of this Document

8

The specification of the functionality classes takes into account that consuming cryptographic applications have different security requirements and may run on devices with different resources and limitations. This document does not assign functionality classes to the cryptographic applications. This is part of the security evaluation of the devices that consume random numbers.

Part 1 of the technical guideline TR 02102 [TR-02102] recommends cryptographic mechanisms, 9
including appropriate functionality classes for the employed RNG. The same applies to the elliptic curve document of the AIS 46 [AIS46_ECC] but there the focus is on Elliptic-curve cryptography (ECC).

Apart from this document, the Bundesamt für Sicherheit in der Informationstechnik (BSI) also 10
provides other documents on the security of RNGs (e.g., [Linux_RNG_2016], [Linux_RNG_2020], [Linux_RNG_2022], and [RNG_virtual_env]). They can be found here.

This document does not contain a complete evaluation methodology (which would precisely 11
describe the tasks of the developer, the evaluator, and the certifier), but it specifies for each functionality class a list of deliverables (including security proofs) that an applicant of a certificate has to provide to the evaluator. The complete evaluation methodology is specified by separate documents to which the AIS 20 [AIS20] and AIS 31 [AIS31] refer. This approach has the advantage that this document can easily be applied to evaluation schemes other than the CC (Common Criteria).

This document does not make any statements about the patent situation of mechanisms described 12
here.

1.3 Structure of this Document

This document consists of five chapters. 13

Chapter 1 places this document in the overall context. 14

Section 2.1 explains the scope and the limits of this document. These explanations are relevant 15
for the overall evaluation of the device in which the RNG is implemented. Sections 2.2 and 2.3 give a brief introduction and motivation for readers who are not yet familiar with AIS 20 and AIS 31. Both sections may be skipped by experienced readers without loss of information. Finally, Section 2.4 briefly addresses other RNG standards.

Chapter 3 is the core of this document. Six functionality classes for DRNGs (DRG.2, DRG.3, 16
DRG.4), PTRNGs (PTG.2, PTG.3) and NPTRNGs (NTG.1) are defined and application notes explain how to apply the particular requirements. One subsection addresses cross-class aspects. Furthermore, in Chapter 3 background information is explained, definitions are introduced, and the specification of the functionality classes is motivated. Chapter 3 refers at various places to sections, subsections, paragraphs and concrete examples of Chapter 4 and Chapter 5. These references may be normative or informative.

17

Chapter 4 provides central mathematical concepts that are important or at least helpful for the evaluation of RNGs according to AIS 20 and AIS 31. Chapter 4 also serves as a reference for different questions that commonly arise during the evaluation of an RNG. In Section 4.5 the concept of a stochastic model and the purpose of the online test and of the total failure test are explained in detail and illustrated by simple examples. To be compliant to the functionality classes PTG.2 and PTG.3, PTRNGs need to apply effective online tests and total failure tests while the stochastic model is the core of an evaluation of a PTRNG. In Section 55 the statistical tests are specified that the evaluator has to apply to the raw random numbers of PTRNGs.

18 Chapter 5 illustrates the concepts of Chapter 4 by more complex examples. This may be useful for both the design and the evaluation of cryptographic post-processing and non-cryptographic post-processing algorithms, the evaluation of noise sources, and online tests. Exemplary verifications of the requirements of the functionality classes are intended to make developers and evaluators familiar with the subject matter. In Section 5.3 the conformity of the National Institute of Standards and Technology (NIST)-approved designs in [SP800-90A] to the functionality classes DRG.3 and DRG.4 is analyzed. Developers (applicants for certificates) may refer to Section 5.3, which disburdens them from having to produce security evidence themselves. Section 5.6 summarizes the results from a long-term study on the Linux RNGs, /dev/random and /dev/urandom, commissioned by the BSI. These results can be referenced by the developers.

19 A glossary, lists of acronyms, abbreviations from Common Criteria, and of symbols, and the references conclude the document.

2 AIS 20 and AIS 31 — Scope, Limits, RNG Classes, and Concepts

Chapter 2 is informative. Section 2.1 outlines the scope and the limits of AIS 20 and AIS 31, 20
while Sects. 2.2 and 2.3 explain the fundamental concepts of this document. In particular, the classification of RNGs is motivated. The Sections 2.2 and 2.3 are written for readers who are not yet or only slightly familiar with the AIS 20 and AIS 31. These sections are intended to facilitate the introduction to the subject area. We do not provide detailed definitions there as they are stated and explained in the subsequent chapters. Furthermore, the text is linked to the glossary. The Sections 2.2 and 2.3 may be skipped by experienced readers.

2.1 Scope and Limits of the AIS 20 and AIS 31

This document treats Deterministic RNGs (DRNGs), Physical true RNGs (PTRNGs), and Non- 21
physical true RNGs (NPTRNGs). In Chapter 3 six functionality classes are defined. Generic requirements are formulated that an RNG has to fulfill to be conformant to a particular functionality class. The requirements are technology neutral and thus leave room for new designs. This shall encourage research and new developments in this field. Whether a particular RNG actually fulfills these requirements has to be verified in a security evaluation.

Besides the requirements of the functionality class, further aspects and features exist, which are 22
also relevant for the security of random number generation, but that are largely outside the scope of this document. An RNG is usually not the Target of Evaluation (TOE) of a CC evaluation but a component of a larger device (e.g., of a smart card or software product) that employs the RNG. Depending on the threats, assumptions, and security objectives formulated in the Security Target (ST), there are additional requirements that have to be covered by the overall security evaluation of the product. Below we briefly address several aspects. We do not claim that this list is exhaustive.

As a rule, the vulnerability analysis of smart cards is performed according to the requirements of 23
the highest class, AVA_VAN.5. In particular, if an RNG is a component of a smart card (which is usually the case for PTRNGs), the RNG implementation shall be secured against implementation attacks (par. 26) and attacks on the memory and data channels (par. 27), all against high attack potential.

A developer of a cryptographic application has to select an RNG belonging to an appropriate 24
functionality class. This document contains advice and informative examples for what purpose RNGs from different functionality classes can be used. However, it does not assign functionality classes to cryptographic applications. Whether a particular functionality class is suitable for a cryptographic application is part of the security evaluation of this application. Furthermore, considerations regarding resilience or redundancy in order to satisfy safety requirements or provide additional security are out of scope.

The output values of RNGs treated in this document behave (in a certain sense and to a par- 25
ticular degree) similarly to values assumed by independent random variables that are uniformly

distributed on $\{0,1\}$ or on $\{0,1\}^k$ for some integer $k > 1$. For many applications (e.g., generating a key for a block cipher), the generation of bit strings of an appropriate length usually suffices. But other applications require random values with special properties (e.g., uniformly distributed values in $\{0,1,\ldots,q-1\}$ for Elliptic Curve Digital Signature Algorithm (ECDSA) or prime numbers of a certain size for Rivest–Shamir–Adleman cryptosystem (RSA)). The transformation from random bit strings to application specific random values is not part of a general-purpose RNG and thus not considered in this document. Instead, this is part of the evaluation of the respective cryptographic application. In the case of using a DRNG, this evaluation also needs to consider whether the security strength (of cryptographic mechanisms) provided by the random numbers is sufficient. The technical guideline [TR-02102] proposes appropriate transformations for the above-mentioned applications.

26 Devices intended for high security applications are usually required to be resistant to implementation attacks. In particular, this comprises hardware attacks, side channel attacks, and fault attacks. In this case, the noise source as well as the algorithmic components of an RNG need to be protected against attacks that might compromise or allow the manipulation of random numbers. Such attacks are not discussed in this document but have to be covered by the overall security evaluation of the product. While fault attacks are outside the scope of this document, accidental failures and unintended weaknesses of the noise source are considered in the functionality classes for PTRNGs (cf. online test and total failure test).

27 Memory and data channels containing sensitive data should be protected against unauthorized access and manipulation during operation and memory should be securely erased after use. This may comprise physical security measures, restrictions regarding logical access and attacks, or protection against cloning due to virtualization. Examples of sensitive memory areas include the internal state of a DRNG instance, a ring buffer of a PTRNG, or registers used for cryptographic post-processing. Furthermore, it might be necessary to establish replay-protected secure channels to guarantee authenticity, integrity, and confidentiality of messages between components of the RNG and applications requesting random numbers. These considerations should be taken into account in the overall system design and are outside the scope of this document.

28 The online tests and total failure tests treated in this document focus on asserting a proper working condition of a noise source. Other tests that are not directly used to assess the quality or the strength of the RNG are out of scope. For example, Known-answer tests (KATs) or other self tests in order to ensure the correct basic working of a device (e.g., algorithmic parts of an RNG) are not covered by this document. Whether the RNG applies such tests should be explained in the ST of the TOE.

2.2 RNG classification and functionality classes

29 The random number generators (RNGs) considered in this document output random bit strings, i.e., digital binary-valued random sequences. We point out that this constraint does not exclude constructions having intermediate analog values. In this section we illuminate and explain fundamental concepts behind the AIS 20 and AIS 31. The following explanations are intended to facilitate the reading of this document. They are informative and do neither replace or supersede the requirements of the particular functionality classes defined in Chapter 3 nor the

corresponding application notes.

The crucial question when evaluating random generators is: which properties constitute a secure 30
RNG? A 'natural' requirement would be the following: The RNG should output all admissible values with the same probability and independently from predecessors and successors. This characterizes an ideal RNG, which is easy to define in terms of random variables. However, it is a purely mathematical construct. In the real world it is virtually impossible to build an ideal RNG, at least in a strict mathematical sense. Furthermore, even if an ideal RNG existed, it would be infeasible to prove or verify ideal randomness.

Instead, the best one can do is to aim for RNGs that are 'close' to an ideal RNG in a certain 31
sense. The rationale is that IT security applications usually demand 'secure' random numbers, which can neither be predicted nor determined later; cf. Sections 4.1 and 4.3. For TRNGs this 'security' can be measured in terms of entropy, while for DRNGs a computational equivalent is needed; cf. Section 3.3. This document divides RNGs into three main classes.

The first class contains the Deterministic RNGs (DRNGs) a.k.a. Pseudorandom number gen- 32
erators (PRNGs). DRNGs 'extend' short random sequences (called seed or seed material) to (possibly) very long output sequences of random bits in a deterministic way. These output sequences look random, but the unpredictability is based on the secrecy of the seed, whose degree of uncertainty depends on the entropy available from the TRNG. Sometimes additional input is provided during the life cycle of the DRNG. Well-known examples of DRNGs are the NIST-approved designs in [SP800-90A].

The second class comprises the Physical true RNGs (PTRNGs). PTRNGs produce high-entropy 33
random bits from a physical noise source based on a randomness-exhibiting physical phenomenon. This phenomenon may be realized by a physical experiment or by an electronic circuit. Usually, this allows precise entropy estimates. Examples of PTRNGs include constructions based on ring oscillators whose random behavior may be traced back to thermal noise or constructions based on Zener diodes.

Finally, the third class consists of Non-physical true RNGs (NPTRNGs). NPTRNGs also de- 34
liver true random bits but gather their entropy from non-physical noise sources for which a precise entropy estimate is usually impossible because the NPTRNG may run on completely different platforms which are not under the control of the designer. A well-known example are the implementations of /dev/random in certain versions of the Linux kernel.

It is not always possible to sharply distinguish between these three classes because RNGs may 35
have design features from both DRNGs and TRNGs. For instance, DRNGs may get additional input during their life cycle, and TRNGs may apply a cryptographic post-processing.

For the evaluation we conceptually divide RNGs into a deterministic part and a non-deterministic 36
part.

A hybrid RNG is a RNG that has security properties of both DRNGs and TRNGs. We distinguish 37
between hybrid DRNGs and hybrid TRNGs.

38

This document aims at being as open as possible regarding RNG designs and keeping requirements to a minimum. Apart from mild assumptions on the format of the random numbers and on the minimum (average) entropy per bit (for TRNGs), there are almost no restrictions as to what technology or constructions may be used to build an RNG that can be evaluated according to this document. Instead of approved designs, the functionality classes in Chapter 3 formulate technology-independent requirements and specify evidence that the developer needs to provide. This does, however, exclude constructions for which the developer is not able to provide sufficient evidence, maybe due to lack of a clear design rationale or access to intermediate values.

39 This document defines six functionality classes (DRNGs: DRG.2, DRG.3, DRG.4, PTRNGs: PTG.2, PTG.3, NPTRNGs: NTG.1). The functionality classes DRG.4, PTG.3, and NTG.1 define hybrid RNGs. Precise definitions of these functionality classes are given in Chapter 3.

2.3 Stochastic model for PTRNGs

40 The key property of TRNGs is their ability to deliver random numbers which, prior to leaving the device, contain a certain amount of entropy. That means that any entity observing the TRNG from the outside, irrespective of its knowledge about the design of the TRNG, its computational power, or cryptanalytic abilities, is uncertain about the value of the next random number to the degree specified by the entropy claim. This advantage over DRNGs comes at the price of being, in general, more difficult to evaluate. A DRNG's computational security can be evaluated independently of its implementation, and approved standard DRNG mechanisms exist. This is not true for TRNGs, however, where the same design may behave completely different when using different hardware. Due to a lack of standards, TRNGs can take on many different forms and utilize various physical phenomena or properties of the underlying technology. A standardized evaluation approach for TRNGs must be able to cope with this diversity and provide a method for establishing, in each case, with a very high degree of assurance that the output does indeed contain the claimed amount of entropy.

41 A common approach to establish a baseline of assurance is subjecting a TRNG's noise source to a pre-defined series of statistical tests. The idea behind this approach is the following. Entropy is a property of the noise source itself and not the random data produced by it. In mathematical terms, random numbers are a realization of a sequence of random variables describing the behavior of the noise source. The entropy of the random numbers (prior to observing them) depends on the properties of the sequence of random variables. An empirical analysis of the random numbers may allow a determination of the properties of the random variables and thus, the entropy.

42 Such statistical tests analyze input data for certain properties or attempt to find regularities or patterns that allow a partial prediction of the input stream. Indeed, if a TRNG is consistently found to behave less random than expected, it can be safely assumed that this TRNG does not meet its claim. Unfortunately, the converse is not true. The property of a sequence of random variables having a certain entropy means that is it not completely determined by patterns. However, there are infinitely many ways to completely or partly prescribe a stream of output values. Statistically verifying an entropy claim would therefore require demonstrating the absence of any (characteristic) pattern. Then again, any finite collection of statistical tests can only check

for a finite number of pattern types.

Another problem with solely relying on statistical testing is that entropy depends on one's knowledge. Blackbox tests, in the sense that they are not tailored to the design of the noise source, will perceive all information in their input as random. But, while interference from a nearby data bus or power supply can provide additional entropy, this information may also be available to an outside observer. Furthermore, an adversary who is very acquainted with the design of the noise source, may have a much higher chance of predicting the random number than a generic algorithm. Therefore, a blackbox statistical evaluation of a noise source without considering its nature is prone to overestimate the quality of a TRNG. 43

This document describes a practical approach that is applicable to most PTRNGs (and also some NPTRNGs). The relevant portions of the possibly extremely complex behavior of the noise source are taken into account to construct a stochastic model that approximates the true behavior. Taking the stochastic model as a postulate, suitable statistical tests can be chosen to empirically determine relevant parameters and finally calculate the entropy. This approach has been field-tested and successfully applied to many PTRNGs. The working definition of a stochastic model is as follows. 44

A stochastic model provides a partial mathematical description (of the relevant properties) of a (physical) noise source using random variables. It allows the verification of a (lower) entropy bound for the output data (internal random numbers or intermediate random numbers). Formally, a stochastic model consists of a family of probability distributions that contains the true distribution of the noise source output (raw random number) or of suitably defined auxiliary random variables during the lifetime of the physical RNG, even if the quality of the digitized data goes down. The stochastic model is based on and justified by an understanding of the physical noise source. 45

Essentially, a stochastic model is a mathematical formulation of the idea from which the TRNG was designed and how it actually works. Much of the work necessary to construct a stochastic model should already have been done when the TRNG was conceptually designed. Formulating the stochastic model requires an understanding how the TRNG functions. It then enables the evaluator to also understand the idea behind it. A stochastic model of a TRNG can be likened to annotated pseudocode for a piece of software. It is therefore a very natural requirement for the evaluation of a TRNG. It allows the evaluation of different TRNGs using different kinds of noise sources to have the same level so that each submission can be treated in the same way. 46

Once the relevant properties of the raw random numbers have been identified, they can be analyzed and estimated with customized statistical tests. While blackbox statistical tests have to consider all possible patterns, the stochastic model reveals which pattern a test needs to look for. This means that using a stochastic model is not the opposite of statistical testing. The stochastic model is a catalyst to make statistical testing meaningful and practical. In most cases, a stochastic model states a class of mathematical distribution of an intermediate value, but not its precise parameters. Using the model, those parameters can be efficiently empirically determined for a certain device and for certain environmental conditions. 47

Knowing the range in which the true parameters for devices of certain type of TRNG lie allows the calculation of the effect of post-processing and a final determination of the entropy of the 48 internal random numbers. Analyzing the relationship between parameters and the entropy of the output also allows a classification of a desired range, a tolerable range, and a non-tolerable range for the parameters. Then a lean online test can be chosen to monitor the parameters and thus determine whether the entropy claim still holds while the RNG is in operation.

49 Section 4.5 explains the concept of a stochastic model in detail.

2.4 Other RNG standards

Par. 57 provides a list of several RNG standards. The list does not claim to be exhaustive. Here are some short remarks on these documents. 50

The ISO standard [ISO_18031] specifies properties under which RNGs are compliant to this ISO standard. 51

The ISO standard [ISO_20543] considers the evaluation of RNGs. Like the AIS 20 and AIS 31 this standard distinguishes between the evaluation of PTRNGs and NPTRNGs. The core of a PTRNG evaluation is a stochastic model. Furthermore, PTRNGs require efficient online tests (health tests) and total failure tests. 52

The NIST standard [SP800-22] provides a collection of statistical tests for RNGs. 53

The NIST standard [SP800-90A] contains three NIST-approved designs (DRNGs). In Section 5.3 of the present document the Hash_DRBG is analyzed. It is shown that (for specified hash algorithms) the Hash_DRBG is compliant to the functionality class DRG.3 or even to DRG.4, provided that the seeding procedure, reseeding procedure, and high-entropy additional input are appropriate. 54
Note: The document [SP800-90A] is under revision.

The NIST standard [SP800-90B] considers entropy sources. It requires that the developer justifies his entropy claim. Currently, a stochastic model is not mandatory, but can be used to support the justification of the entropy claim. 55

The NIST standard [SP800-90C] defines several RNGs constructions. 56

Here is a list of RNG standards from ISO and NIST. 57

- [ISO 18031] ISO / IEC 18031: Information technology – Security Techniques. Random Bit Generation. 2011 / Cor 1: 2014 / A1: 2017.
- [ISO 20543] ISO / IEC 20543: Information technology – Security Techniques. Test and Analysis Methods for Random Bit Generators within ISO / IEC 19790 and ISO / IEC 15408. 2019.
- [SP 800-22] NIST SP 800-22, Revision 1a: A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, S. Vo, (revision) L. Bassham : A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications, April 2010. https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-22r1a.pdf
- [SP 800-90A] NIST SP 800-90 A, Revision 1: E. Barker, J. Kelsey: Recommendation for Random Number Generators Using Deterministic Random Bit Generators. June 2015. http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf

- [SP 800-90B] NIST SP 800-90 B: M. Turan, E. Barker, J. Kelsey, K. McKay, M. Baish, M. Boyle: Recommendation for the Entropy Sources Used for Random Bit Generation. January 2018. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90B.pdf
- [SP 800-90C] NIST SP 800-90 C, Third Draft: E. Barker, J. Kelsey: Recommendation for Random Bit Generator (RBG) Constructions. September 2022. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90C.3pd.pdf

3 Functionality classes

In Sects. 3.3 (DRNG), 3.4 (PTRNG), and 3.5 (NPTRNG) six functionality classes are defined. 58
Within each subsection the functionality classes are hierarchically ordered: the classes with greater numbers provide more security capabilities.

To keep redundancies low, general explanations are placed before the subsections that define the 59
functionality classes. The reader is often referred to Chapter 4 and Chapter 5.

The definitions of the functionality classes use security functional requirements of the CC com- 60
ponent FCS_RNG.1. The definition of the functionality classes is accompanied by application notes explaining their security capabilities and quality metrics.

The requirements of the functionality classes do not depend on the targeted assurance level (EAL 61
level) of the CC certification process. This also applies to the depth of the evidence.

Section 3.6 considers cross-class aspects. 62

3.1 Evaluation of the RNG: General aspects

Implementation attacks (e.g., side-channel attacks or fault attacks) constitute serious threats 63
against cryptographic implementations. This also applies to RNGs.

As explained in Subsect. 2.1, in particular pars. 26 and 27, implementation attacks are not 64
covered by AIS 20 [AIS20] and AIS 31 [AIS31]. Consequently, implementation attacks must be part of the vulnerability analysis of the TOE to verify that successful implementation attacks on the RNG are impractical.

The most fundamental security requirements for RNGs are backward secrecy and forward secrecy. 65
They formally describe the property of an RNG to be unpredictable, i.e., that knowledge of a subsequence of random numbers does not enable an adversary to determine or to guess the successor or predecessor of this subsequence with non-negligibly greater probability than without the knowledge of this subsequence.

More secure RNGs also provide enhanced backward secrecy and enhanced forward secrecy. These 66
properties aim to lessen the impact of a compromise of the internal state. Enhanced backward secrecy means the following: Assurance that knowledge about previous output values cannot be derived with practical computational effort from the knowledge of the current internal state of an RNG. Analogously, enhanced forward secrecy provides assurance that knowledge about subsequent output values cannot be derived with practical computational effort from the knowledge of the current internal state of an RNG.

In the case of DRNGs, the backward secrecy, forward secrecy, and enhanced backward secrecy 67
requirements shall be ensured by algorithmic properties (i.e., aiming at computational security).

In the case of TRNGs, these requirements shall be ensured using fresh entropy (i.e., aiming at information-theoretic security). The enhanced forward secrecy requirement can only be achieved using fresh entropy.

68 The DRNG functionality classes defined in Section 3.3 use the above security requirements. Classes DRG.2 through DRG.4 gradually increase in security by including more of them. While this is done explicitly for DRNGs, the functionality classes for the TRNGs defined in Sections 3.4 and 3.5 shall satisfy the requirements implicitly as a result of the entropy requirements.

3.2 Overview of the functionality classes

69 Figures 1, 2, 3, and 4 illustrate schematic RNGs designs that are compliant to the particular functionality classes defined below. We point out that these designs are exemplary and that other technical realizations of the class requirements are possible.

70 Fig. 1 illustrates pure DRNG designs. For simplicity, in Figs. 1 and 2 we assume that requests are limited to a single random number (in particular $S_{req} = S$ and ϕ_0 equals the identity mapping, see pars. 139 and 143). Internal random numbers denote the final stage of the random numbers of an RNG that are ready to be output. We use the following notation:

- ϕ = state transition function
- ψ = output function
- A$\rightarrow$B = symbol for a one-way function

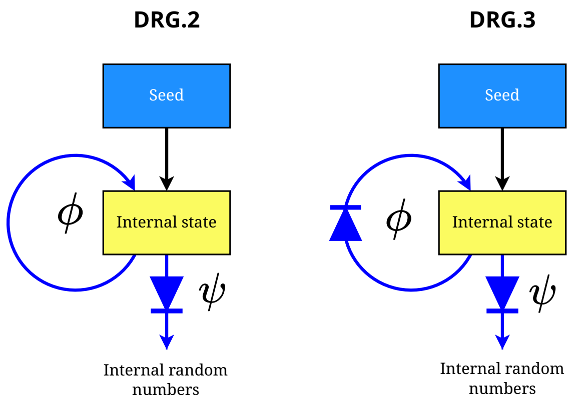

Figure 1: Functionality classes DRG.2 and DRG.3 (exemplary schematic designs)

71 Figure 3 illustrates PTRNG designs. Online tests are indicated with a red background and total failure tests with a pink background.

72

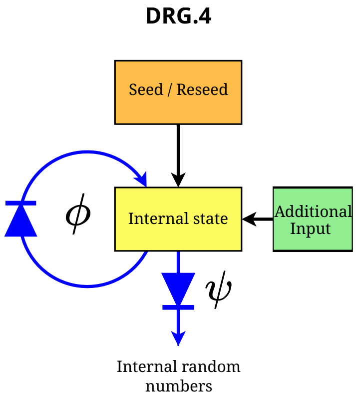

Figure 2: Functionality class DRG.4 (exemplary schematic design)

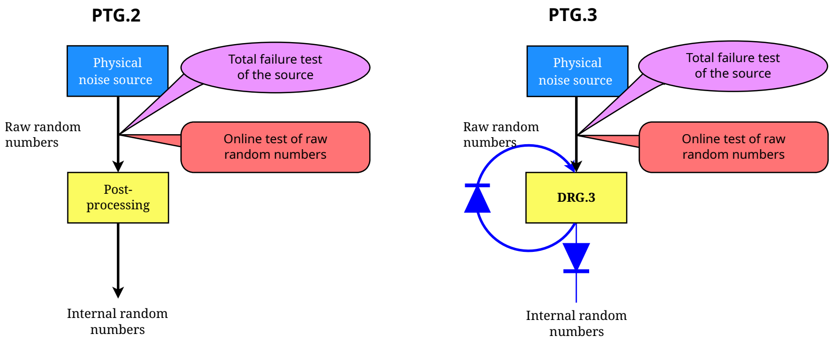

Figure 3: Functionality classes PTG.2 and PTG.3 (exemplary schematic designs)

Figure 4 illustrates a typical NPTRNG design.

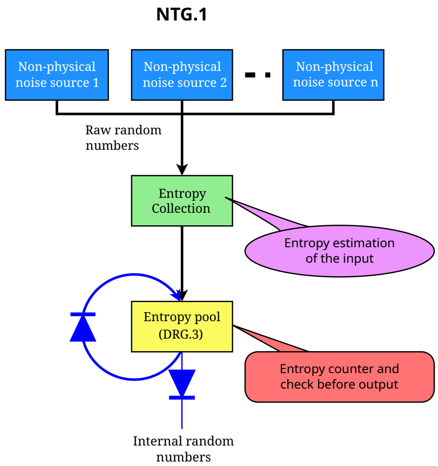

Figure 4: Functionality class NTG.1 (exemplary schematic design)

73 Fig. 5 illustrates the hierarchy between the functionality classes.

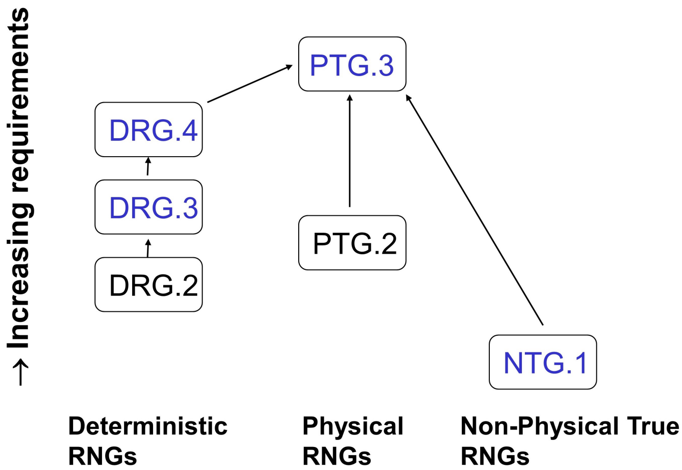

Figure 5: Hierarchy between the functionality classes

3.3 DRNG: Functionality classes

74 Subsections 3.3.3, 3.3.4, and 3.3.5 define the functionality classes DRG.2, DRG.3, and DRG.4. The differences in this version to the previous AIS 20 [AIS2031An_11] are explained in Subsect. 3.3.1. Subsect. 3.3.2 contains explanations that are relevant for all DRNG classes. We begin with general remarks.

75

A DRNG is called a pure DRNG if it does not receive any external input data except by the seeding procedure or possibly by an (externally triggered) reseeding procedure. A DRNG is called hybrid DRNG if it accepts additional input (regardless of its entropy) or if it is able to trigger a seeding procedure or a reseeding procedure. The second condition requires that the DRNG has access to a true RNG.

Originally, functionality classes DRG.2 and DRG.3 were designed for pure DRNGs, but this document also covers hybrid DRNG designs. 76

A DRNG that is compliant to functionality class DRG.2 provides backward secrecy and forward secrecy. 77

If the DRNG is compliant to functionality class DRG.3, it additionally provides enhanced backward secrecy; cf. requirements DRG.3.7 (and DRG.4.7). 78

Enhanced backward secrecy can be achieved by a state transition function ϕ that has the one-way property (i.e., a one-way function). For functionality classes DRG.3 and DRG.4, it is required that the state transition function is a one-way function. 79

For hybrid DRNGs the security requirements backward secrecy, forward secrecy, and enhanced backward secrecy presume that the adversary knows all additional input values. In other words: backward secrecy, forward secrecy, and enhanced backward secrecy shall be ensured by the algorithmic properties of the DRNG alone and without relying on any entropy in the additional input data. 80

A DRG.4 compliant DRNG is compliant to the functionality class DRG.3, too. Additionally, it provides enhanced forward secrecy. 81

The DRG.4-specific security requirement of being able to guarantee enhanced forward secrecy cannot be ensured by pure DRNGs, because without fresh entropy, introduced as additional input or by a reseeding procedure, knowledge of the internal state (and further additional input values) reveals all future random numbers. Enhanced forward secrecy requires a hybrid DRNG that is able to trigger seeding, reseeding, or the input of high-entropy additional input. 82

It should be noted that the definitions of functionality classes DRG.2, DRG.3, and DRG.4 as well as their objectives have been reworked in this version of the document. The definitions are similar to those in [AIS2031An_11] (which justifies maintaining the class names), but they are different in detail. An in-depth explanation of the differences to the previous definitions in [AIS2031An_11] can be found in Section 3.3.1. 83

3.3.1 DRNG: Main Differences from [AIS2031An_11]

The requirements on the seeding procedure, the reseeding procedure, and the size of the internal state (or rather, the effective internal state) have significantly increased (cf. par. 124). 84

85

The document [AIS2031An_11] defines a further functionality class DRG.1 that is weaker than DRG.2, because it only requires forward secrecy. Class DRG.1 has been removed, as it did not turn out to be relevant in certification practice.

86 This document introduces the concept of requests, see pars. 116 to 122.

87 The definition of a request is new in AIS 20. It was not specified in [AIS2031An_11]. Remark: Of course, the situation in [AIS2031An_11] can be interpreted that only requests are allowed for which the number of requested bits coincides with the bit length of a single internal random number.

88 The introduction of a request in AIS 20 is part of the harmonization of AIS 20 and AIS 31 with the NIST documents NIST SP 800-90[A,B,C].

89 Under suitable conditions, AIS 20 now allows the seeding or reseeding of a DRNG by another DRNG; cf. pars. 150 to 155.

90 In [AIS2031An_11], functionality classes DRG.2, DRG.3, and DRG.4 demand that the evaluator applies statistical tests (at least several specified blackbox tests) to the output of the DRNG (deprecated requirements in terminology of [AIS2031An_11]: DRG.2.5, DRG.3.5, DRG.4.7). This mainly had 'historical reasons' because [AIS20An_99] contained two functionality classes (K1 and K2) which allowed non-cryptographic DRNGs.

91 The requirements concerning the application of statistical test suites within the evaluation have been relaxed.

92 In the previous version [AIS2031An_11], compliance to functionality classes DRG.2 or DRG.3 requires that the state transition function and / or the output function shall be cryptographic, while DRG.4 demands that both the state transition function and the output function shall be cryptographic. In the new version of the AIS 20, both functionality classes DRG.3 and DRG.4 require that the state transition function and the output function are cryptographic.

93 The previous versions of AIS 20 ([AIS20An_99; AIS2031An_11]) contained a security requirement that says that within the life cycle of a DRNG instance, high-dimensional random bit vectors shall be mutually disjoint with very large probability; see the requirements DRG.1.3, DRG.2.4, DRG.3.4, DRG.4.6 in [AIS2031An_11]. These requirements are not included in this document.

94 The mutual disjointness requirement in [AIS20An_99; AIS2031An_11] was motivated by the fact that in the past, the internal states of the DRNG implementations on resource-limited devices like smart cards usually were smaller than today. Furthermore, [AIS20An_99] also permitted non-cryptographic DRNGs (compliant to functionality classes K1 or K2), which could be used for non-sensitive applications. The main reason for this requirement was to prevent too many random numbers being generated within a life cycle of the DRNG instance relative to the size of the internal state.

95 In this version of AIS 20, par. 124 formulates high requirements on the effective internal state

and on the entropy of its initial state. Thus, considering the new upper bound of random bits per life cycle of a DRNG instance, the requirement of mutual disjointness has been dropped.

Compared to [AIS2031An_11] (cf. Table 12) the minimal size of the effective internal state and of its entropy after the seeding procedure / reseeding procedure has become much larger (cf. pars. 124, 126). 96

The describing 6-tuple in [AIS2031An_11] (cf. par. 111 ff.) is replaced by a describing 9-tuple (par. 139). This is mainly due to the introduction of the concept of requests (cf. pars. 116 to 122). Furthermore, describing 5-tuples for the seeding procedure and reseeding procedure have been introduced (pars. 158 and 162). 97

The previous version, [AIS2031An_11], distinguishes between different attack potentials; cf. Table 1, Table 2, Table 12, Table 13, and the corresponding paragraphs. In this document the requirements do not depend on the attack potential. 98

3.3.2 DRG.[2,3,4]: Definitions, requirements, and justification

The seedlife of a DRNG instance begins with the seeding procedure or with the reseeding procedure, respectively. It ends with the next reseeding procedure or when the DRNG is uninstantiated. Uninstantiation results in the termination of a DRNG instance. In particular, the internal state and secret parameters are deleted. 99

Knowledge of the internal state of a DRNG allows the prediction of future outputs until sufficient fresh entropy is mixed into the internal state (from additional input or by applying the reseeding procedure). Pure DRNGs do not receive fresh entropy before uninstantiation or before the next reseeding procedure, respectively. 100

Consequently, it is a minimum requirement for the security of a DRNG that it must not be possible to guess the entire internal state with non-negligible probability. 101

Depending on the DRNG design, it may be possible that an adversary knows or is able to learn parts of the internal state. This, in particular, refers to publicly known parameters and values. In Subsection 5.2.1, for example, a DRNG that applies AES-256 in Output feedback (OFB) mode is discussed. A part of the internal state (128 bits) coincides with the next random number and may thus be known to an adversary. Those parts of the internal state do not contain entropy to defend against guessing attacks. Nevertheless, those parts may still have a positive impact on the security of the DRNG, e.g., against pre-computation attacks (unless these parts are always the same) and multi-target attacks; cf. par. 126. 102

We refer to the security-critical part of the internal state of a DRNG that an adversary does not know and which he cannot determine or guess (with probability that is significantly greater than indicated by its size; we always assume optimal encoding) from the knowledge of subsequences of internal random numbers as the *effective internal state*; cf. par. 105. Par. 113 provides an illustrating example. 103

104

The concept of the effective internal state applies to backward secrecy and forward secrecy. An important requirement of functionality classes DRG.2, DRG.3, and DRG.4 is that enough entropy is inserted into the DRNG by the seeding procedure, the reseeding procedure, or by high-entropy additional input. For the enhanced backward secrecy, the size of the effective internal state is not relevant (neglecting the fact that enough entropy has been inserted) because the adversary is assumed to know the whole internal state anyway.

105 The definition of the effective internal state in par. 103, of course, does not take an adversary with unlimited computational power into account, because an adversary with unlimited computational power could determine the complete internal state from a few random numbers. Instead, our definition aims at computational security. This is reasonable because an adversary with unlimited computational power would be able to break any DRNG.

106 The effective internal state and its size shall be determined under the assumption that the adversary knows a large number of internal random numbers (limited by the maximum number of random numbers between subsequent seeding procedures/reseeding procedures); cf. par. 190.

107 The uncertainty of the effective internal state from the view of an adversary shall be based on the seed material. The security of the effective internal state shall not be based on a personalization string, secret parameters, etc. (Kerckhoffs's principle) although, of course, these measures may support security. If it is possible to assign entropy to the secret parameters, e.g., because they were generated by a strong TRNG, they may be counted for the effective internal state.

108 The applicant for a certificate has to provide evidence that their DRNG design fulfills the class-specific requirements.

109 AIS 20 does not prescribe approved DRNG designs. However, it is strongly recommended to use widely recognized cryptographic primitives and techniques. Otherwise, the evaluation and the verification of the algorithmic properties of the DRNG (such as backward secrecy, forward secrecy, and enhanced backward secrecy) may become impractical.

110 Cryptographic primitives are considered to be widely recognized cryptographic primitives if they have undergone diversified scientific review from many researchers and if the cryptographic community has no serious doubts concerning their security strength (of cryptographic mechanisms) in relevant operational circumstances. Examples: the AES block cipher, and the SHA-2 and SHA-3 families of hash functions.
Note: AIS 20, in particular, views cryptographic algorithms as widely recognized that are recommended in the technical guideline [TR-02102]. Further cryptographic primitives are possible *if accepted by the certification body.*

111 If widely recognized cryptographic primitives are used, the applicant may claim the generally accepted properties of these primitives in the security proofs of the class-specific requirements.

112 Examples of the generally accepted security properties of cryptographic primitives are the following:
(i) The AES is not susceptible to chosen-plaintext attacks (cf. Subsect. 5.2.1, pars. 777 and 779).
(ii) The one-way function property, second pre-image resistance, and collision resistance of

SHA -256.

Example: AES-256 in OFB mode (cf. Subsection 5.2.1): The internal state comprises 384 bits (a 128-bit vector plus the long-term key). Since the first 128 bits of the internal state equal the next random number, the effective internal state only comprises 256 bits. In order to claim that an observer cannot practically determine bits of the long-term key from the output (random numbers), the applicant may present security proofs relying on well-established properties of AES; cf. Subsection 5.2.1. 113

We refer to state transition functions and output functions as cryptographic if they are composed of cryptographic primitives (e.g., block ciphers or hash functions). Incrementation by 1, simple XOR-additions, additions and multiplications in small moduli, Linear-feedback shift registers (LFSRs), and projections, for example, *are not viewed as cryptographic*. The composition of cryptographic primitives with a non-cryptographic operation usually remains cryptographic. 114
Example: $s \mapsto \text{SHA-256}(s) + 1 \bmod 2^{256}$.

It should be noted that cryptographic functions are not automatically suitable. The output function $s \mapsto \text{SHA-256}(s) + \text{SHA-256}(s) \bmod 2^{256}$, for instance, is cryptographic but obviously weak because the least significant bit is always 0. Furthermore, consider par. 128. 115

[Definition of a request] Many DRNG designs only produce a sequence of internal random numbers of a fixed bit size (e.g., 128 bits). Other DRNG designs, such as the NIST-approved Deterministic random bit generators (DRBGs), allow for generating a variable, larger amount of random bits (e.g., when the DRNG is called by an application which requires a certain amount of random bits). Upon receiving an external request, the DRNG outputs the desired number of random bits. Depending on the size of the internal random numbers in bits, the DRNG internally generates one or several internal random numbers to piece together the desired number of random bits. It may happen that the last internal random number is only partially output; cf. pars. 138 to 141 for a formal description. Internal random numbers bits that have not been output in the request shall not be stored but deleted. 116

In order to formulate verifiable requirements for backward secrecy, forward secrecy, and enhanced backward secrecy, this document uses the concept of a granularity level on which these security properties hold. 117
For example, a DRNG satisfies *forward secrecy on the granularity level of internal random numbers* if knowledge about subsequent internal random numbers cannot be derived with practical computational effort from the knowledge of current or previous internal random numbers.
Analogously, *enhanced backward secrecy on the granularity level of requests* means assurance that knowledge about previous requests cannot be derived with practical computational effort from the knowledge of the current internal state of an RNG.

The security requirements for DRNG in this document were formulated to be applicable to as many designs as possible and make a successful evaluation of 'secure' DRNG feasible. However, in a strict mathematical sense, they do not guarantee 'ideal security' in every conceivable way. For example, forward secrecy on the granularity level of internal random numbers is weaker than forward secrecy on the granularity level of single output bits or a similar security property where an adversary knows previous as well as some subsequent output values. 118
But forward secrecy on the granularity level of single output bits is substantially harder to prove,

if not impossible. For this reason, the DRNG functionality classes defined in this document 'only' require forward secrecy on the granularity level of internal random numbers paired with other requirements such as using widely recognized cryptographic primitives or statistical inconspicuousness. This gives the developer of a secure DRNG the practical means to convince an evaluator that the DRNG is indeed 'secure'. It is the task of the evaluator to identify weak DRNG designs and reject them on the basis of violating the requirements.

119 The requests of a DRNG satisfy the condition of conceptual atomicity if the DRNG finishes every request by the application of the state transition function before any of the requested bits are used.

120 [justification of the conceptual atomicity condition (cf. par. 119)] Enhanced backward secrecy on the granularity level of requests (i.e., 'between' requests) protects the internal random numbers generated for previous requests even if the current internal state is compromised. But it may happen that the internal state or the request state is compromised *during* a request (i.e., while the DRNG generates a internal random numbers for a request). In this case, enhanced backward secrecy on the granularity level of requests *does not* prevent an adversary to determine previous internal random numbers that have been generated for the current request. While conceptual atomicity does not effectively prevent this situation, it assures the following condition. If the DRNG has not been compromised until the point in time when the output is used, then the output remains unaffected from possible compromises of the DRNG in the future. A more effective requirement instead of conceptual atomicity would be that a request cannot be suspended or is executed within a short period of time (e.g., one second). This requirement, however, would mean that a DRNG algorithm can only be evaluated together with the entire environment in which it is executed. Furthermore, in many environments this requirement cannot be fulfilled (e.g., due to CPU scheduling, virtualization, or lack of access to a clock signal).

121 The conceptual atomicity condition can be satisfied by outputting all random numbers of a request only after the processing for the request has been terminated. Smart cards and alike usually write the internal random numbers to the target memory location while they are being generated. Here, the conceptual atomicity condition is fulfilled if the consuming application waits until the request has been terminated before processing any of the generated bits. This does, of course, limit the maximal size of a request to the size of the memory available to buffer generated data.

122 [conceptual atomicity condition (par. 119); exception] If the DRNG provides enhanced backward secrecy on the granularity level of internal random numbers (i.e., 'within' requests), the conceptual atomicity condition can be dropped. More precisely, it then suffices that the request state is overwritten after an internal random number has been generated. In order to actually achieve enhanced backward secrecy for the implementation of a DRNG (and not just its conceptual model), the guidance in par. 144 needs to be taken into account.
Note: The Hash_ DRBG (cf. Subsect. 5.3.1), for example, provides enhanced backward secrecy on the granularity level of requests but not within requests. In contrast, the HMAC_ DRBG (cf. Subsect. 5.3.2) provides enhanced backward secrecy on the granularity level of internal random numbers.

123 The next paragraph contains an informative summary of the requirements on the maximum number of random bits within a seedlife, the minimum size of the effective internal state, and

its minimal entropy after the seeding procedure (or reseeding procedure) for functionality classes DRG.2, DRG.3, or DRG.4. The normative requirements are contained in the definitions of the functionality classes below.

[DRNG: Minimal requirements]; cf. the requirements DRG.x.2, DRG.x.3, DRG.x.4 ($x \in \{2, 3, 4\}$) and DRG.4.10 124

- Within the lifetime of seed material (seedlife) of a DRNG instance at most 2^{48} requests may be output. Each request shall comprise at most 2^{19} bits.
- The min-entropy of the initial effective internal state after the seeding procedure, resp. after the reseeding procedure, shall be at least 240 bits. Alternatively, 250 bits of Shannon entropy suffice, provided that the raw random numbers of the true RNG are stationarily distributed.

The second bullet point implies that the size of the effective internal state is larger than 240 bits. A more precise formulation of the size condition would be the following. For some $m > 240$, the set $\{0, 1\}^m$ can be embedded into the set of all admissible effective internal states S_{eff}, i.e., there exists an injective mapping $\chi\colon \{0, 1\}^m \to S_{\text{eff}}$. Similarly, a more precise formulation of the entropy condition would be the following. Assume that the random variable X takes on values in S_{eff} and describes the seeding procedure, resp. the reseeding procedure. Then $H_{min}(X) \geq 240$ or $H(X) \geq 250$, respectively. 125

The lower bounds for the entropy after the seeding procedure / reseeding procedure and, implicitly, for the minimal size of the effective internal state defined in par. 124 shall repel, among other things, potential threats by quantum computers and by multi-target attacks. It should be noted that for the prevention of multi-target attacks, it would suffice to increase the size of the internal state accordingly (instead of the effective internal state), provided that all parts of the internal state affect the internal random numbers. 126

If the applicant claims Shannon entropy for the seeding procedure or the reseeding procedure with a PTRNG, an additional condition has to be met, namely the stationarity (time-local stationarity) of the raw random numbers; cf. DRG.2.4, DRG.3.4, DRG.4.4, and application note par. 191). The stationarity condition shall prevent well-known pathologies that may arise when probability distributions are (extremely) unbalanced; cf. par. 550, for example. 127

In the presence of quantum computers, DRNG designs whose security relies on the hardness of factoring or on the discrete log problem will likely become irrelevant (due to being susceptible to quantum attacks). Until then such designs may be used but, of course, parameter(s) shall be selected so that the instances resist all known factoring algorithms or algorithms that compute the discrete logarithm. This document discourages using DRNG designs that are based on the hardness of factoring or the discrete log problem. 128

The entropy of the initial effective internal state is an upper bound for the overall entropy of the generated random numbers or subsets thereof if no fresh entropy is mixed into the internal state. 129

As already explained in par. 105, DRNGs are ineffective against an adversary with unlimited 130 computational resources. Relative to an adversary with unlimited computational resources, the effective internal state would contain entropy only if the adversary did not know more than a few random bits (information-theoretic security) or had not otherwise observed information about the seed material to test their hypotheses. But, if the DRNG is computationally secure, then a resource-limited adversary observing random numbers will still be unable to determine the information that is contained in the (effective) internal state.

131 There are two generic attacks to guess an unknown k-bit output string x of a DRNG, namely blind guessing of x and blind guessing of the initial internal state (or seed material, whichever has less entropy). If an adversary guesses the initial internal state correctly, this adversary can compute all random numbers for the current seedlife (assuming they also know any additional input data that was provided). Requirements DRG.2.4, DRG.3.4, DRG.4.4, respectively, ensure that the second attack is (at least approximately) a 240-bit problem. For an 'ideal' DRNG, guessing x (first attack) would be a k-bit problem if $k < 240$ (first attack) and (at least) a 240-bit problem, otherwise.

132 In the context of forward secrecy, backward secrecy, and enhanced backward secrecy, for k-bit output strings the security strength (of cryptographic mechanisms) shall not be significantly lower than for the ideal case (cf. par. 130), i.e., 240 bits. Furthermore, the security strength (of cryptographic mechanisms) of the DRNG shall not significantly decrease over time.

133 The entropy of the internal state can decrease during the life cycle of a DRNG instance if the state transition function ϕ is not bijective. To be compliant to the functionality class DRG.3, ϕ must be a one-way function. The effective internal state shall contain sufficient entropy within the whole life cycle of the instance to prevent successful guessing attacks.

134 Ideally, after the seeding procedure/reseeding procedure, the probabilities of all possible values of the effective internal state would be the same. From an adversary's point of view (trying to guess the effective internal state), this represents the worst-case, as every value would be equally likely. Since the state transition function ϕ is usually not bijective, the probability distributions of the internal states (or rather, the effective internal states) might become more and more imbalanced over time, thereby reducing the entropy of the internal state. If an adversary had precise knowledge of these distributions, they could be leveraged to speed up guessing attacks by selecting the most probable internal states after n iterations of ϕ.

135 However, such a deep understanding of the iterated application of ϕ would likely also allow analytical attacks on the forward secrecy property. If theoretical statements on the distributions of future internal states are possible, this shall be considered in the security proofs of the algorithmic properties of the DRNG.
Note: Sect. 4.4 treats random mappings.

136 An adversary without such deep understanding (of the distributions of the internal state / of the effective internal state after the iterated application of ϕ) could only try to mount a generic guessing attack to exploit a growing imbalance of the distribution of the (effective) internal states. For example, to obtain a guess of the internal state s after the n^{th} iteration of ϕ, the adversary could randomly select an element of S and apply the state transition function ϕ n times. In this case, each single guess is much more costly than a single 'blind' guess of the internal state after n iterations of ϕ. For a hash function that is a widely recognized cryptographic primitive (e.g., for

SHA-256), it is assumed that even with precomputations, the entropy loss due to the iterated application is not practically exploitable.

If the DRNG design allows a computation of the n-fold composition of ϕ that is significantly faster than a step-by-step evaluation of ϕ or other ways to speed up guessing, this shall be considered in the evaluation. 137

DRNGs usually have a core function that generates blocks of internal random numbers of a size prescribed by the chosen cryptographic primitive (e.g., designs based on AES usually generate blocks of 128 bits). For simple DRNG designs this core function coincides with the output function (which means that the DRNG can only generate random numbers of a fixed size). But in practice the logic to generate random numbers of a requested length is often already integrated into the DRNG. In order to formulate security requirements, we use the following formal model. 138

The algorithmic structure of a DRNG can be described by a 9-tuple 139

$$(S, S_{req}, R, A, I, \phi, \phi_{req}, \phi_0, \psi) \qquad \text{(describing 9-tuple of the DRNG)} \tag{3.1}$$

The components of (3.1) have the following meaning:

- S = set of admissible internal states (typically, $S = \{0,1\}^n$)
- S_{req} = set of admissible (temporary) internal request states
- R = set of admissible output values (internal random numbers), $R = \{0,1\}^k$ for some $k \in \mathbb{N}$.
- A = set of admissible additional input (typically, $A = \{0,1\}^*$, where o denotes the empty string (= no additional input))
- I = set of admissible request lengths, counted in bits
- $\phi\colon S \times A \times I \to S$ (state transition function, logically computed at the end of a request)
- $\phi_{req}\colon S \times A \to S_{req}$ (generates the internal request state)
- $\phi_0\colon S_{req} \to S_{req}$ (request state transition function)
- $\psi\colon S_{req} \to R$ (output function)

If public parameters and / or secret parameters affect any of the mappings ϕ, ϕ_{req}, ϕ_0, or ψ, then these data items are part of S. 140

If the request requires the generation of $p \in I$ bits, then $m := \lceil \frac{p}{k} \rceil$ internal random numbers are generated (where k is the dimension of R, i.e., the bit length of the internal random numbers as 141 explained in par. 138). In pseudocode, the DRNG works as follows:

$$
\begin{aligned}
& s_{req} := \phi_{req}(s, a) \\
& \text{for } j := 1 \text{ to } m \text{ do } \{ \\
& \qquad r_j := \psi(s_{req}); \\
& \qquad s_{req} := \phi_0(s_{req}, a); \\
& \} \\
& s := \phi(s, a, p)
\end{aligned} \tag{3.2}
$$

Depending on p (number of requested random bits), the m^{th} internal random number may be truncated when output. More precisely, the right-most $k\lceil \frac{p}{k} \rceil - p$ bits of the last internal random number are not output.
Note: Often, $S_{req} \subseteq S$. In this case, the internal request state s_{req} 'vanishes' automatically right after the request has ended. Prominent examples are the Hash_ DRBG and the HMAC_ DRBG [SP800-90A]. Otherwise, if $S_{req} \not\subseteq S$, the internal request state s_{req} shall be overwritten or erased when the request has been terminated. Usually, for $j = m$, the item $s_{req} := \phi_0(s_{req})$ of the loop need not be executed.

142 If the DRNG does not allow additional input, then $A = \{o\}$ and ϕ, ϕ_{req}, and ϕ_0 actually do not depend on A.

143 The 9-tuple describes the *conceptual structure* of the DRNG. As already mentioned in the note of par. 141, within a request S_{req} often coincides with a subset of S.

144 For the analysis of enhanced backward secrecy the evaluator shall not only consider the 9-tuple (i.e., the conceptual structure of the DRNG) but also the DRNG itself and (if applicable) its implementation. The conceptual structure can be used to verify that certain information that practically allow computing previous output values need not remain in the internal state. The evaluator shall verify that this is indeed the case. For example, if a DRNG derives a request state s_{req} from a state s (cf. par. 141) and claims enhanced backward secrecy on the on the granularity level of internal random numbers (i.e., within requests), then s needs to be deleted or overwritten. Analogously, after a request has been completed, all information on the request state s_{req} need to be deleted from memory in order to achieve enhanced backward secrecy. Note that by definition, the request state is part of the internal state of the DRNG. Vice versa, if during a request the variable s is still physically present in memory, then it belongs to the internal state of the DRNG.

145 Consider a pure DRNG that only allows requests of $\leq k$ bits. Since only one internal random number is generated per request, the space S_{req} is not really needed. This scenario can be modeled as follows: $S_{req} = S$, and $\phi_{req} \colon S \to S_{req}$ denotes the identity mapping.
Note: Here it is permitted to define the state transition function as $\phi \colon S \to S$ instead of $\phi \colon S \times A \times I \to S$.

146 Example: see Subsection 5.2.1 (pure DRNG, request length $\leq k$).

147 A DRNG derives its initial internal state from a randomly selected seed value (seed material). The 'seed entropy', that is, the entropy contained in the bit string used for the seeding procedure, the reseeding procedure (if applicable), and a description of how the seed material was generated

must be covered by the deliverables from the applicant; cf. pars. 181, 202, and 217.

[TRNG seeding DRNG] If the seed material (for the seeding procedure or reseeding procedure) is generated by a PTRNG (compliant to PTG.2 or PTG.3) or by an NPTRNG (compliant to NTG.1), the requirements PTG.2.3, PTG.3.4 or NTG.1.5 guarantee lower entropy bounds per internal random number bits. The requirements PTG.2.3 and PTG.3.4 allow claims in both Shannon entropy and min-entropy while NTG.1.5 prescribes min-entropy. 148

[TRNG seeding DRNG] The entropy requirements for seeding, reseeding, and inserting additional high-entropy input are quantified in min-entropy. Under suitable conditions, namely if the seed material is generated by a PTRNG and if its raw random numbers are time-locally stationarily distributed (fulfilled for functionality classes PTG.2 and PTG.3), the min-entropy claim can be substituted by a Shannon entropy claim; cf. requirements DRG.2.4, DRG.3.4, DRG.4.4, DRG.4.10 for details. 149

[DRNG seeding DRNG] There are scenarios in which no TRNG is available to seed a DRNG. Typically, this affects software implementations on PCs, servers, etc. for which no TRNG exists that might be called by the applications. One example is the following: The DRNG of the operating system has been seeded by an NPTRNG, and the applications call this DRNG for seed material to seed their own DRNG. 150

[DRNG seeding DRNG] For these reasons, a DRNG may optionally also be seeded / reseeded by another DRNG, provided that certain conditions are fulfilled (specified in par. 150). These conditions shall mitigate additional security threats and risks (exemplarily addressed in par. 152) that are caused by the fact that the seeding procedure/reseeding procedure does not use a TRNG. In particular, this requires that the origin and entropy of the first DRNG's seed material as well as its security properties are known to an evaluator in order to verify the security requirements for the second DRNG. 151

[DRNG seeding DRNG; additional risks and security measures] Compared to the usual way of (re-)seeding a DRNG with a TRNG, (re-)seeding with a DRNG bears additional security risks. When (re-)seeding with a TRNG, it suffices that the TRNG works properly at that time. Possible entropy defects or successful attacks in the past (or in the future) are not relevant. When (re-)seeding uses a DRNG, an (undetected) compromise of the internal state of the (re-)seeding DRNG would affect *all* DRNGs that are seeded after this event. For (re-)seeding chains of more than two DRNGs, the whole chain has to be considered. In particular, 'seed cycles' (i.e., cycles in the seed tree) shall be prevented because then a DRNG would transitively (re-)seed itself. Due to the sketched security problems, we strictly recommend the use of a TRNG for (re-)seeding if it is available. 152

[DRNG seeding DRNG] DRNG B is called a *direct seed successor* of DRNG A if DRNG B has been seeded by DRNG A. Vice versa, DRNG A is called a *direct seed predecessor* of DRNG B. DRNG C is a *seed successor* of DRNG A, if a chain of direct successors exists from DRNG A to DRNG C. Then, DRNG A is a *seed predecessor* of DRNG C. 153

[DRNG seeding DRNG, seed tree structure] The seed tree is a graph with a distinguished node, called the *root DRNG*. The root DRNG receives seed material directly from a TRNG. If DRNG B is a direct seed successor of DRNG A (i.e., if DRNG B has been seeded by DRNG A) it is a 154 child node of DRNG A in the seed tree. In particular, apart from the root DRNG, each DRNG in the seed tree has been seeded by a DRNG. As usual, the *height* of a seed tree is the length of the longest path from the root DRNG to a leaf DRNG. In particular, a seed tree that consists only of a root DRNG has height 0.

155 [DRNG seeding DRNG, compliant seed tree] This paragraph summarizes requirements for seeding procedures and reseeding procedures in a compliant seed tree. One of the main goals is to prevent seed cycles. In particular, requirement (iv) implies a tree structure.

(i) The root DRNG shall be compliant to functionality class DRG.3 or DRG.4. The root DRNG shall exclusively use a TRNG to generate the seed material for the seeding procedure and reseeding procedure.

(ii) Apart from root and the leaves (i.e., the nodes corresponding to DRNG with no direct seed successor), all nodes of the seed tree shall algorithmically be compliant to class DRG.3. This means that these DRNGs shall fulfill requirements DRG.3.1 to DRG.3.3 and DRG.3.5 to DRG.3.10. For a leaf node, it suffices that the corresponding DRNG is algorithmically compliant to functionality class DRG.2. That means fulfilling requirements DRG.2.1 to DRG.2.3 and DRG.2.5 to DRG.2.9. In particular, DRG.2-compliant DRNGs shall not seed other DRNGs.

(iii) Each DRNG in the seed tree (except for the root DRNG) shall be seeded with at least 256 bits of seed material. Then Requirement DRG.3.4 (resp. DRG.2.4) is considered to be fulfilled if the following condition holds: If the seed material had been generated by a TRNG and if its min-entropy were ≥ 250 bits, then after the seeding procedure or reseeding procedure the initial effective internal state has min-entropy ≥ 240 bits.

(iv) Apart from the root, each DRNG in the seed tree shall only be reseeded by its direct seed predecessor in the seed tree. In particular, if its direct seed predecessor is not available (e.g., uninstantiated), it shall not receive seed material from another DRNG.

(v) A (re-)seeded DRNG shall use received seed material only for the seeding procedure or reseeding procedure.

(vi) The DRNGs in a compliant seed tree can accept additional input but it shall not be credited as seeding or reseeding. In particular, it cannot not substitute a reseeding procedure. Additional input (cf. DRG.2.7, DRG.3.8, DRG.4.8) should not come from seed successors.

(vii) All DRNGs within the seed tree shall be inside the same security boundary. This need not apply to the TRNG that (re-)seeds the root DRNG. The seed material shall not leave the security boundary of the DRNG.

Note 1: It is possible to evaluate and certify only a subtree that contains the root. This could be, for example, a node in the tree consisting of all its predecessors up to the root and all successors. The criteria apply accordingly to the subtree.

Note 2: Requirement (iv) means that the seed tree is static in the sense that no nodes or edges shall be removed during operation. Formally, uninstantiated DRNGs remain in the seed tree (they can be viewed as ‘greyed out’). Requirement (iv) prevents seed cycles, because it implies a tree structure. This eliminates the need to maintain a dynamic seed tree structure during the operation of the DRNGs. It suffices that each DRNG 'remembers' its direct seed predecessor.

Note 3: For the usual case where the seed material for a DRNG is generated by a TRNG, requirement DRG.3.4 (resp. DRG.2.4) is an information-theoretical condition. This condition cannot be met when seeding a DRNG with another DRNG. To at least quantify the information contained in the internal state of the seeded DRNG, the evaluator would have to consider all state transition functions, which were applied to all seed predecessors in the seed tree before the DRNG under consideration is (re-)seeded, and, additionally, the internal random numbers that were used as seed material for the predecessors. This would be practically infeasible. As a substitute, Requirement (iii) is suitable to at least ensure the plausibility that sufficient entropy is 'transported' from the seed material of the root DRNG to the effective internal state of the seeded DRNG.

Note 4: Although it is not credited (cf. requirement (vi)), it may be advisable to use additional input that, e.g., stems from non-physical noise sources, even if the guaranteed amount of entropy is low.

Note 5: The security of the scenario 'DRNG seeding DRNG' may depend on further aspects, which are out of scope for this document. This, in particular, refers to the secure transport of seed material from one DRNG to another DRNG (e.g. integrity, authenticity, confidentiality of the seed material as well as its secure deletion after the seeding procedure or the reseeding procedure has been performed). These security aspects shall be covered by the overall security evaluation of the product.

[Personalization string] An optional input parameter to the seeding procedure is a personalization string. This denotes a freely chosen value, often derived from information related to a specific DRNG instance. Unique personalization strings can inhibit side channel analysis and may prevent an adversary in control of the seed material from identifying the internal state of the DRNG. Furthermore, if the personalization string is kept secret, it may be the last resort if the seed material (cf. pars. 158 to 159) is compromised. 156

The 9-tuple (3.1) describes the algorithmic structure of the DRNG when it is in operation. Similarly, the seeding procedure and the reseeding procedure can be formally described. 157

The following 4-tuple describes the algorithmic aspects of the seeding procedure: 158

$$(SM, PS, S, \phi_{seed}) \qquad \text{(describing 4-tuple of the seeding procedure)} \tag{3.3}$$

The components of (3.3) have the following meaning:

- SM = set of admissible values of the seed material (typically, $SM = \{0,1\}^s$)
- PS = set of personalization strings (may contain public and secret parts)
- S = set of admissible internal states
- $\phi_{seed}\colon SM \times PS \to S$ (seeding procedure)

159 The security of the seeding procedure shall not be based on the entropy of the personalization string (even if it contains secret parameters). The seed material itself shall contain enough entropy to meet the requirements (DRG.2.4, DRG.3.4, DRG.4.4).

160 Simple seeding procedures are:
(i) $\phi_{seed}(sm, o) := sm$. The seed is copied into the internal state.
(ii) $\phi_{seed}(sm, o) := \phi(sm, o)$. The seed material is copied into the internal state and the state transition function ϕ is applied once.

161 Other, more complex seeding procedures exist, cf. the NIST-approved DRBG designs in [SP800-90A], for example.

162 The following 4-tuple describes the algorithmic aspects of the reseeding procedure:

$$(SM, PS, S, \phi_{reseed}) \qquad \text{(describing 4-tuple of the reseeding procedure)} \tag{3.4}$$

The components of (3.4) have the following meaning:

- SM = set of admissible seed material values (typically, $SM = \{0, 1\}^s$)
- PS = set of personalization strings (may contain public and secret parts)
- S = set of admissible internal states
- $\phi_{reseed}\colon S \times SM \times PS \to S$ (reseeding procedure)

163 The security of the reseeding procedure shall not be based on the entropy of the current internal state or on the personalization string (even if it contains secret parameters). The seed material itself shall contain enough entropy to meet the requirements (DRG.2.4, DRG.3.4, DRG.4.4). However, using the current internal state as an additional parameter is recommended.

164 Calculating the precise probability distribution on the set of all admissible effective internal states after the seeding procedure and the reseeding procedure allows the determination the resulting entropy. However, except for very simple seeding procedures (e.g., when using bijections, as in par. 160), this will in general be practically infeasible. The same is true for the distribution on S, the set of all admissible internal states. Thus, unlike in the 6-tuple in [AIS2031An_11], the distribition on S after (re-)seeding is not a component of the describing 9-tuple. Instead, the requirements DRG.2.4, DRG.3.4, and DRG.4.4 aim at the entropy induced on the set of all admissible effective internal state.

165 In order to verify requirements concerning the entropy of the initial effective internal state (DRG.2.4, DRG.3.4, DRG.4.4), specifying the precise probability distribution is not demanded. Instead, it suffices to specify a lower entropy bound. The developer shall present a justified estimate, e.g., by modeling widely recognized cryptographic primitives such as hash functions or block ciphers as random mappings or random bijections. The evaluator decides whether simplifications made in this model are acceptable and whether the properties of random mappings and random bijections can be used as assumptions. Section 4.4 collects the properties of random mappings and random bijections.

The characteristics of random mappings usually cannot be applied if the definition range is too small. To give an extreme example: It is not permitted to model a randomly selected mapping $\{0,1\}^{16} \rightarrow \{0,1\}^{16}$ as a random mapping and to draw conclusions from Section 4.4. 166

In Subsect. 5.2.2, pars. 797 to 799, it is shown that if the state transition function of a hybrid DRNG is too simple, an adversary who is able to manipulate additional input data might be able to control the evolution of the internal state. This may weaken a DRNG completely. 167

Subsect. 5.2.2, par. 801 provides an example for disastrous interaction of the state transition function and the output function. A related example that accepts additional input data is $\phi(s,a) = \text{SHA-256}(s) + 1 + a \bmod 2^{256}, \psi(s) = \text{SHA-256}(s)$. This feature weakens the DRNG completely. 168

If additional input is permitted, it shall not weaken the algorithmic strength of the DRNG. The hybrid DRNG version shall not be less secure than the pure DRNG version of the DRNG that does not allow additional input (DRG.2.7, DRG.3.8, DRG.4.8). The hybrid DRNG mentioned in par: 168 does not meet these requirements. 169

A reasonable design strategy to prevent (chosen) additional input from weakening the DRNG certainly is to select a state transition function whose 'core' is a hash function and use it to mix the additional input into the internal state. However, depending on the DRNG design, simpler state transition functions may also be appropriate. Pitfalls where additional input weakens the DRNG should primarily be an issue for DRG.2-compliant DRNGs. 170

To ensure enhanced forward secrecy, additional input with sufficient entropy or a reseeding procedure is needed (cf. requirement DRG.4.10). One-time high-entropy input cannot be compensated by many low-entropy additional inputs within several requests. 171
Example: Assume that a random byte is mixed into the internal state of the DRNG in each iteration, i.e., whenever an internal random number is generated. Assume further that an adversary knows the current internal state and that the DRNG generates 128-bit internal random numbers which the application uses as AES keys. Provided that he knows a plaintext / ciphertext pair for each AES key an adversary could successively guess, e.g., the next 2^{10} AES keys with $\leq 2^{10} \cdot 2^8 = 2^{18}$ guesses.

It shall not be possible to distinguish sequences of internal random numbers that are generated by a DRNG that is compliant to the DRG.2, DRG.3, or DRG.4 functionality class from output sequences of ideal RNGs by statistical tests. Of course, 'unfair' statistical tests that exploit knowledge of the internal state of the DRNG are excluded. 172

Widely recognized cryptographic primitives should not show any statistical weaknesses. 173

Example: AES, CBC mode: ciphertext blocks (interpreted as a bit sequence) of arbitrary plaintexts should not show any statistical weaknesses. 174

Usually, evidence that the DRNG fulfills requirement DRG.2.9, DRG.3.10, or DRG.4.11, can be given by theoretical considerations about the cryptographic primitives. Then, the application of 175 statistical (blackbox) tests to the output of the DRNG is not necessary.

176 Note that in the presence of implementation or design flaws, the use of widely recognized cryptographic primitives alone does not preclude the existence of statistical weaknesses. Extreme examples: The output of a hash function is (accidently) concatenated with itself or bit strings are filled up with zeroes.

177 If the evaluator suspects that a given DRNG design does not meet the condition statistical inconspicuousness, targeted statistical tests should be applied to test for these possibilities.

3.3.3 Functionality Class DRG.2

178 Functionality class DRG.2 defines requirements for deterministic RNGs (DRNG).

179 DRG.2-compliant DRNGs are primarily recommended for cryptographic applications for which the disclosure of previous random numbers due to a compromise of the internal state is not an issue (e.g., for challenges in challenge-response protocols).
Note: A compromise of the internal state constitutes a serious security incident that is supposed to be prevented by the overall TOE.

180 The TOE Security Functionality (TSF) has to protect the internal state of the RNG from being compromised.

181 **DRG.2-specific deliverables by the applicant** The security architecture description and developer evidence shall contain at least

- a formal description of the algorithmic behavior of the DRNG by a 9-tuple (3.1),
- a formal description of the seeding procedure (3.3) and (if applicable) the reseeding procedure (3.4),
- a description of how the seed material for the seeding procedure and (if applicable) reseeding procedure is generated (DRG.2.1),
- proofs that the DRNG design and the seeding TRNG fulfill requirements DRG.2.1 to DRG.2.8.
- evidence that DRG.2.9 is fulfilled.

182 **DRG.2: Security functional requirements**

Security functional requirements of class DRG.2 are defined by component FCS_RNG.1 with specific operations as given below.

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

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

(DRG.2.1) The seed material (for the seeding procedure and reseeding procedure) shall be generated by a TRNG or DRNG. If a TRNG is used, the TRNG shall [selection: be a PTRNG of class PTG.2, be a PTRNG of class PTG.3, be an NPTRNG of class NTG.1, generate random bits with an average [selection: min-entropy, Shannon entropy] of [assignment: amount of entropy] per bit]. If a DRNG is used, it shall be part of a compliant seed tree.

(DRG.2.2) With each seedlife, at most 2^{48} requests shall be output. The length of a single request shall comprise at most 2^{19} bits.

(DRG.2.3) The effective internal state shall comprise more than 240 bits.

(DRG.2.4) The initial effective internal state (after the seeding procedure or the reseeding procedure) shall have [selection: min-entropy ≥ 240 bits, Shannon entropy ≥ 250 bits]. If Shannon entropy is claimed, the seed material-generating TRNG shall be a PTRNG with stationarily distributed raw random numbers.

(DRG.2.5) The DRNG shall provide forward secrecy on the granularity level of internal random numbers.

(DRG.2.6) The DRNG shall provide backward secrecy on the granularity level of internal random numbers.

(DRG.2.7) If applicable: additional input shall not weaken the strength of the DRNG even if an adversary is able to control the additional input.

(DRG.2.8) The state transition function ϕ, the output function ψ, or both shall be cryptographic.

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

(DRG.2.9) The internal random numbers shall have statistical inconspicuousness. This conclusion shall be based on [selection: theoretical considerations, theoretical considerations supported by statistical tests, statistical tests with justification of the choice].

Application notes

[DRG.2.1] Usually, the seeding procedure and the reseeding procedure use a PTRNG that is compliant to PTG.2 or PTG.3, or an NPTRNG that is compliant to NTG.1. 184

[DRG.2.1] It is permitted to use a TRNG that is not compliant to functionality classes PTG.2, PTG.3, or NTG.1 to generate the seed material. Of course, the verification of requirement DRG.2.1 then usually requires significantly greater effort. In particular, the applicant has to give evidence that the seed material (for the seeding procedure and reseeding procedure) indeed contains the claimed amount of entropy. This includes a reliable verification that the TRNG is working properly at the time when the seed material was generated; cf. par. 186 for an extreme example. The entropy claim has to be explained and justified. For functionality classes 185

DRG.2 and DRG.3, a formal stochastic model for the seeding PTRNG is recommended but not mandatory. Without a stochastic model more conservative (pessimistic) entropy estimates shall be applied. Blackbox tests are not sufficient. By default, min-entropy has to be claimed. Claiming Shannon entropy requires a stochastic model, and the raw random numbers need to be (time-locally) stationarily distributed.
Note: We refrain from making a stochastic model mandatory, because for NPTRNGs a stochastic model is not mandatory, either.

186 [DRG.2.1] Extreme example: Assume that a seeding PTRNG requires, for instance, a delay of 100.000 internal random number bits to reliably detect non-tolerable entropy defects or total failures of the noise source. In this case, it would be acceptable to generate and to store the seed material, apply an appropriate test on the next 100.000 random bits, and to output the seed material in the case of a positive test result. Of course, this PTRNG would not be compliant to functionality class PTG.2, because the above-mentioned online test would be too slow for general applications.

187 [DRG.2.1, DRNG seeding DRNG] A DRG.2-compliant DRNG can only be a leaf in the compliant seed tree whereas the seeding DRNG shall be compliant to class DRG.3; cf. par. 155.

188 [DRG.2.2] For further explanations regarding requests, see pars. 116 to 124. Unlike for functionality class DRG.3, for functionality class DRG.2, no conceptual atomicity condition (cf. requirement DRG.3.7) exists, because class DRG.2 only ensures backward secrecy and forward secrecy on the level of the internal random numbers (cf. the requirements DRG.2.5 and DRG.2.6). The conceptual atomicity of a request is only relevant in the context of enhanced backward secrecy.

189 [DRG.2.2] Requirement DRG.2.2 makes counters for the number of requests and internal random numbers necessary.

190 [DRG.2.3] The effective internal state and its size shall be determined under the assumption that the adversary knows a large number of internal random numbers. In particular, the following (conceivable) reasoning *will not be accepted* for the verification of requirement DRG.2.3: Since immediately after the seeding procedure / reseeding procedure an adversary did not have a chance to collect any information about the DRNG, any part of the internal state (apart from publicly known input) is unknown and thus effective.

191 [DRG.2.4] If a certified TRNG compliant to functionality class PTG.2, PTG.3, or NTG.1 is used for the seeding procedure or reseeding procedure, then usually the verification of requirement DRG.2.4 is an easy task. For class PTG.2, requirement PTG.2.3 ensures that the Shannon entropy per bit exceeds 0.9998, and if a min-entropy claim has been certified, the min-entropy per bit exceeds 0.98. For class PTG.3, the applicant has several options to claim PTRNG-specific entropy bounds, either in terms of Shannon entropy, min-entropy, or both (cf. requirement PTG.3.4). If the seeding procedure or the reseeding procedure applies a PTG.3-compliant PTRNG without a specific entropy claim (corresponding to the first three selections in requirement PTG.3.4), the entropy claims of the intermediate random numbers are used instead (i.e., ≥ 0.9998 bit Shannon entropy and / or ≥ 0.98 bit min-entropy), although the cryptographic post-processing may increase the entropy defect per bit. For class NTG.1, the min-entropy is applied to quantify a lower min-entropy bound per internal random number bit; cf. requirement NTG.1.5. If TRNGs are used for the seeding procedure or the reseeding procedure that are not

compliant to PTG.2, PTG.3, or NTG.1, the entropy per seed material bit can be lower. It is crucial to verify that the seeding procedure and reseeding procedure (under consideration of the seed material-generating TRNG) fulfills requirement DRG.2.4. If the TRNG is not certified, this requires (at least a partial) evaluation of the TRNG to derive a reliable lower entropy bound per bit. For the reseeding procedure the least favorable case shall be assumed where an adversary knows the previous internal state.

[DRG.2.4] For PTRNGs that belong to functionality classes PTG.2 or PTG.3, the raw random numbers satisfy the time-local stationarity condition; cf. requirements PTG.2.1 and PTG.3.1. 192

[DRG.2.[5,6,7]] The requirements DRG.2.5, DRG.2.6, and DRG.2.7 shall be guaranteed by the algorithmic properties of the DRNG, i.e., by the interaction of the state transition function and the output function. A lack of algorithmic properties cannot be compensated for by other measures, e.g., by high-entropy additional input. 193

[DRG.2.[5,6]] The DRNG may support generating output of variable length (by concatenating internal random numbers, cf. par. 141). Irrespective of that, DRG.2.5 and DRG.2.6 only require forward secrecy and backward secrecy on the granularity level of the internal random numbers. This means the following: Assume that an adversary knows a sequence of internal random numbers that have been generated within one or several requests. The sequence need not start nor terminate a request. The task of the adversary is to compute or to guess the internal random number that follows or precedes this sequence. The security strength (of cryptographic mechanisms) should be considered as explained in pars. 131 and 132. 194

[DRG.2.[5,6]] The focus on internal random numbers in DRG.2.5 and DRG.2.6 (instead of considering strings of internal random numbers bits of arbitrary length) will simplify the evaluation. The focus on internal random numbers is motivated by the fact that the internal random numbers are the basic building blocks of a request. Thus, forward secrecy and backward secrecy should extend from the granularity level of internal random numbers to the granularity level of strings of internal random number bits of arbitrary length with corresponding computational security strength. It is part of the evaluation to identify 'pathological' DRNG designs for which this is not the case. 195

[DRG.2.7] The term 'if applicable' in requirement DRG.2.7 refers to DRNGs that accept additional input. 196

[DRG.2.8] In case of doubt, the certification body decides whether a function is considered cryptographic. 197

[DRG.2.9] Regarding DRG.2.9 we refer to pars. 172 to 177. 198

3.3.4 Functionality Class DRG.3

Functionality class DRG.3 defines requirements for deterministic RNGs. The differences to class DRG.2 are explained in par. 207. 199

200

DRG.3-compliant DRNGs are suitable for all cryptographic applications except for those that require guaranteed fresh entropy.

201 The TSF has to protect the internal state of the RNG from being compromised.

202 **DRG.3-specific deliverables by the applicant** The security architecture description shall contain at least

- a formal description of the algorithmic behavior of the DRNG by a 9-tuple (3.1),
- a formal description of the seeding procedure (3.3) and (if applicable) the reseeding procedure (3.4),
- a description of how the seed material for the seeding procedure and (if applicable) reseeding procedure is generated (DRG.3.1),
- proofs that the DRNG design and the seeding TRNG fulfill requirements DRG.3.1 to DRG.3.9.
- evidence that DRG.3.10 is fulfilled.

203 **DRG.3: Security functional requirements**

Security functional requirements of class DRG.3 are defined by component FCS_RNG.1 with specific operations as given below.

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

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

(DRG.3.1) The seed material (for the seeding procedure and reseeding procedure) shall be generated by a TRNG or DRNG. If a TRNG is used, the TRNG shall [selection: be a PTRNG of class PTG.2, be a PTRNG of class PTG.3, be an NPTRNG of class NTG.1, generate random bits with an average [selection: min-entropy, Shannon entropy] of [assignment: amount of entropy] per bit]. If a DRNG is used, it shall be part of a compliant seed tree.

(DRG.3.2) With each seedlife, at most 2^{48} *requests shall be output. The length of a single request shall comprise at most* 2^{19} *bits.*

(DRG.3.3) The effective internal state shall comprise more than 240 *bits.*

(DRG.3.4) The initial effective internal state (after the seeding procedure or the reseeding procedure) shall have [selection: min-entropy ≥ 240 *bits, Shannon entropy* ≥ 250 *bits]. If Shannon entropy is claimed, the raw random numbers of the seed material-generating PTRNG shall be stationarily distributed.*

(DRG.3.5) The DRNG shall provide forward secrecy on the granularity level of internal random numbers.

(DRG.3.6) *The DRNG shall provide backward secrecy on the granularity level of internal random numbers.*

(DRG.3.7) *The DRNG shall provide [selection: enhanced backward secrecy on the granularity level of internal random numbers, enhanced backward secrecy on the granularity level of requests and the requests shall satisfy the condition of conceptual atomicity].*

(DRG.3.8) *If applicable: additional input shall not weaken the strength of the DRNG even if an adversary is able to control the additional input.*

(DRG.3.9) *The state transition function ϕ and the output function ψ shall be cryptographic. The state transition function shall be a one-way function.*

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

(DRG.3.10) *The internal random numbers shall have statistical inconspicuousness. This conclusion shall be based on [selection: theoretical considerations, theoretical considerations supported by statistical tests, statistical tests with justification of the choice].*

Application notes

[DRG.3 vs. DRG.2] Class DRG.3 includes the requirements of class DRG.2. The requirements DRG.2.1 and DRG.3.1, DRG.2.2 and DRG.3.2, DRG.2.3 and DRG.3.3, DRG.2.4 and DRG.3.4, DRG.2.5 and DRG.3.5, DRG.2.6 and DRG.3.6, DRG.2.7 and DRG.3.8, DRG.2.9 and DRG.3.10 coincide. 205

[DRG.3 vs. DRG.2] Therefore, the corresponding application notes 184, 185, 186, 189, 190, 191, 192, 193, 194, 195, 196, and 198 are valid for DRG.3.y instead of DRG.2.x as well, where x and y correspond as described in par. 205. 206

[DRG.3 vs. DRG.2] In addition to the DRG.2 requirements, functionality class DRG.3 requires enhanced backward secrecy (DRG.3.7), and DRG.3.9 extends DRG.2.8. 207

[DRG.3.[5,6,7,8]] Same as in par. 193, the requirement DRG.3.5 to DRG.3.8 shall be guaranteed by the algorithmic properties of the DRNG, i.e., by the interaction of the state transition function and the output function. Missing algorithmic properties cannot be compensated by other measures, e.g., by high-entropy additional input 208

[DRG.3.7] Enhanced backward secrecy (DRG.3.7) is an algorithmic property. It cannot be compensated for or supported by technical security measures that (are claimed to) prevent the internal state from being compromised or modified. Clause DRG.3.7 essentially requires a state transition function that is a one-way function with respect to an adversary who knows the internal state and (if relevant) the last additional input. The security strength (of cryptographic mechanisms) should be as explained in pars. 131 and 132. 209

[DRG.3.7] The DRNG may support output of variable length (by concatenating internal random numbers, cf. par. 141). Clause DRG.3.7 requires enhanced backward secrecy on the granularity 210 level of requests. (If the DRNG outputs requests that do not comprise more than one internal random number, then DRG.3.7 trivially guarantees enhanced backward secrecy on the granularity level of internal random numbers.) This means the following: Assume that an adversary gains access to the current internal state and (if applicable) to the additional input during the previous request. Then requirement DRG.3.7 prevents an adversary from computing or guessing internal random numbers from previous requests. However, requirement DRG.3.7 does not prevent an adversary who has learned the internal state before the state transition function has been applied to compute or to guess all internal random numbers of this request. The impact of this attack is mitigated by the atomicity condition (primarily) and by the length restriction of a request. If the DRNG provides enhanced backward secrecy on the granularity level of internal random numbers, the atomicity condition is dropped.

211 [DRG.3.7] Size and entropy requirements for the effective internal state shall prevent successful attacks against forward secrecy and backward secrecy. Concerning enhanced backward secrecy, by assumption the adversary knows the current internal state. However, attacking the enhanced backward secrecy shall be about as difficult as attacks against forward secrecy and backward secrecy. This means that computing or guessing n bits of previously generated internal random number bits is at least about a $\min\{240, n\}$ bit problem (non-quantum attacker). For an adversary with access to a (sufficiently powerful) quantum computer the effort can be smaller (e.g., by Grover's algorithm) but a successful attack shall still be infeasible.

212 [DRG.3.9] For class DRG.3 both the state transition function ϕ and the output function ψ shall be cryptographic. Additionally, ϕ shall be a one-way function. We refer the reader to pars. 109 to 115. Note that the final decision whether a function is considered cryptographic rests with the certification body.

3.3.5 Functionality Class DRG.4

213 Functionality class DRG.4 defines requirements for deterministic DRNG. These requirements can only be fulfilled by hybrid DRNGs. The differences to class DRG.3 are explained in pars. 221 and 223.

214 DRG.4-compliant DRNGs are suitable for all cryptographic applications except for those that require a TRNG.

215 DRG.4-compliant DRNGs have access to a PTRNG during the seeding procedure, the reseeding procedure, and maybe to obtain high-entropy additional input. Furthermore, the additional input may also include data from sources without an entropy claim. These sources neither need entropy claims nor provide additional security guarantees. However, DRG.4.8 requires that these additional input data shall not weaken the security of the DRNG.

216 The TSF has to protect the internal state of the RNG from being compromised.

217 **DRG.4-specific deliverables by the applicant** The security architecture description shall contain at least

- a formal description of the algorithmic behavior of the DRNG by a 9-tuple (3.1),
- a formal description of the seeding procedure (3.3) and (if applicable) the reseeding procedure (3.4),
- a specification of the internal PTRNG and the mechanisms to trigger a seeding procedure and / or a reseeding procedure, and / or to obtain high-entropy additional input,
- a description of how the seed material for the seeding procedure and (if applicable) reseeding procedure is generated (DRG.4.1),
- proofs that the DRNG design and the internal PTRNG fulfill requirements DRG.4.1 to DRG.4.10
- evidence that DRG.4.11 is fulfilled.

DRG.4: Security functional requirements 218

Security functional requirements of 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) 219

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

(DRG.4.1) The seed material (for the seeding procedure and reseeding procedure) and (if applicable) the high-entropy additional input shall be generated by a PTRNG. The PTRNG shall [selection: be a PTRNG of class PTG.2, be a PTRNG of class PTG.3, generate random bits with an average [selection: min-entropy, Shannon entropy] of [assignment: amount of entropy] per bit].

(DRG.4.2) With each seedlife, at most 2^{48} requests shall be output. The length of a single request shall comprise at most 2^{19} bits.

(DRG.4.3) The effective internal state shall comprise more than 240 bits.

(DRG.4.4) The initial effective internal state (after the seeding procedure or the reseeding procedure) shall have [selection: min-entropy $\geq$ 240 bits, Shannon entropy $\geq$ 250 bits]. If Shannon entropy is claimed, the raw random numbers of the seed material-generating PTRNG shall be stationarily distributed.

(DRG.4.5) The DRNG shall provide forward secrecy on the granularity level of internal random numbers.

(DRG.4.6) The DRNG shall provide backward secrecy on the granularity level of internal random numbers.

(DRG.4.7) The DRNG shall provide [selection: enhanced backward secrecy on the granularity level of internal random numbers, enhanced backward secrecy on the granularity level of requests and the requests satisfy the condition of conceptual atomicity].

(DRG.4.8) *If applicable: additional input shall not not weaken the strength of the DRNG even if an adversary is able to control the additional input.*

(DRG.4.9) *The state transition function ϕ and the output function ψ shall be cryptographic. The state transition function shall be a one-way function.*

(DRG.4.10) *The DRNG shall provide enhanced forward secrecy [selection: on demand, on condition [assignment: condition], after [assignment: time]]. This is achieved by the seeding procedure (cf. DRG.4.4), the reseeding procedure (cf. DRG.4.4) or by high-entropy additional input generated by a PTRNG such that the effective internal state has [selection: min-entropy ≥ 240 bits, Shannon entropy ≥ 250 bits]. If Shannon entropy is claimed, the raw random numbers of the seed material-generating PTRNG shall be stationarily distributed.*
The DRNG may apply several of the above-mentioned methods. Minimum requirement: Until the next seeding procedure, reseeding procedure, or the next high-entropy additional input at most 2^{17} internal random number bits shall be generated.

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

(DRG.4.11) *The internal random numbers shall have statistical inconspicuousness. This conclusion shall be based on [selection: theoretical considerations, theoretical considerations supported by statistical tests, statistical tests with justification of the choice].*

Application notes

220 [DRG.4 vs. DRG.3 vs. DRG.2] Class DRG.4 includes the requirements of class DRG.3 and thus also of DRG.2. The requirements DRG.2.2 and DRG.4.2, DRG.2.3 and DRG.4.3, DRG.2.4 and DRG.4.4, DRG.2.5 and DRG.4.5, DRG.2.6 and DRG.4.6, DRG.2.7 and DRG.4.8, DRG.2.9 and DRG.4.11 coincide. Furthermore, DRG.3.7 and DRG.4.7, DRG.3.9 and DRG.4.9 coincide.

221 [DRG.4 vs. DRG.3 vs. DRG.2] Requirement DRG.4.1 limits the selection of a TRNG in DRG.2.1 and the selection of a TRNG or DRNG in DRG.3.1 to PTRNGs for class DRG.4.

222 [DRG.4 vs. DRG.3 vs. DRG.2] Thus, the application notes 188, 190, 191, 192, 193, 194, 195, and 198 remain valid if we replace DRG.2.x by DRG.4.y with regard to the correspondences from par. 220. Moreover, the application notes 207, 208, 209, 210, and 212 remain valid if one substitutes DRG.3.x by DRG.4.x.

223 [DRG.4 vs. DRG.3] Class DRG.4 includes the requirements of class DRG.3. Additionally, DRG.4 requires that the DRNG has the capability to ensure enhanced forward secrecy (DRG.4.10).

224 [DRG.4.1] Functionality class DRG.4 requires a PTRNG for the seeding procedure, the reseeding procedure (DRG.4.4), and for high-entropy additional input (DRG.4.10). Unlike for functionality classes DRG.2 and DRG.3 NPTRNGs and DRNGs are not allowed. This is justified by the fact that for NPTRNGs the environment, the platform, etc. are not under the control of the designer or evaluator. Moreover, the devices on which NPTRNGs run (PCs, server, mobile devices, etc.)

are usually more vulnerable to implementation attacks than, e.g., smart cards; cf. Subsect. 3.5.2. For these reasons, in general our trust in NPTRNGs is lower than our trust in PTRNGs.

[DRG.4.1] Usually, the seed material (for the seeding procedure or reseeding procedure) and high-entropy additional input is generated with a PTRNG that is compliant to classes PTG.2 or PTG.3. 225

[DRG.4.1] It is permitted to use a PTRNG that is not compliant to PTG.2 or PTG.3. In this case, the verification of requirement DRG.4.1 usually requires significantly greater effort than if a certified PTRNG is used that is compliant to class PTG.2 or PTG.3. In particular, the applicant has to give evidence that the seed material (for the seeding procedure and reseeding procedure) contains the claimed amount of entropy. This includes evidence that the PTRNG is working properly at the time of the seeding procedure/reseeding procedure. Unlike for functionality classes DRG.2 and DRG.3, a stochastic model of the PTRNG is mandatory. Claiming Shannon entropy requires that the raw random numbers are (time-locally) stationarily distributed. 226

[DRG.4.2, DRG.4.10] Three methods exist how fresh entropy can be mixed into the internal state: seeding, reseeding, or inserting high-entropy additional input. The requirements are essentially identical for all methods; cf. DRG.4.2, DRG.4.10. Unless fresh entropy is inserted by high-entropy additional input but instead by a seeding procedure or reseeding procedure, requirement DRG.4.10 ensures that the maximum number of requests per seedlife is limited by $\lceil 2^{17}/k \rceil$, which is much smaller than the limit 2^{48} from requirement DRG.4.2. Here, k denotes the size of the internal random numbers in bits. 227

[DRG.4.10] The entropy claim in requirement DRG.4.10 refers to the least favorable case where an adversary knows the previous internal state; cf. par. 191. 228

[DRG.4.10] Enhanced forward secrecy can only be achieved for internal random numbers that are generated after the next seeding procedure, after the next reseeding procedure, or after high-entropy additional input data have been mixed into the internal state by the state transition function. If high-entropy additional input is used, in general the internal random numbers of the current request do not provide enhanced forward secrecy. Exceptions are possible, if ϕ_{req} (the function that generates S_{req}), the request state transition function ϕ_0 and the output function ψ 'mix' the high-entropy additional input suitably into the generation of internal random numbers. Such examples are the Hash_ DRBG and the HMAC_ DRBG; see (5.37), (5.38), and (5.84), (5.84), and (5.39). If this is not the case, the DRNG shall apply the state transition function ϕ first (corresponds to an 'empty request') 229

[DRG.4.10, high-entropy additional input] If requirement enhanced forward secrecy is intended to be achieved by high-entropy additional input, the applicant shall describe the used PTRNG (as is required for the seeding procedure or reseeding procedure). In particular, requirement DRG.4.1 applies. This is unlike the case of using 'arbitrary' additional input, which is covered by DRG.4.8. 230

[DRG.4.10, high-entropy additional input] When achieving enhanced forward secrecy by high-entropy additional input, the requirements are rather similar to that for the reseeding procedure although, of course, the state transition function is applied in place of the (re-)seeding procedure. If a CC certificate confirms the compliance of the PTRNG to PTG.2 or PTG.3, it suffices that the 231 developer refers to this fact. Additional input strings derived from both reliable RNGs (e.g., from a PTG.2-compliant PTRNG) and ‘arbitrary’ noise sources (e.g., time stamps) are permitted. Of course, additional input from ‘arbitrary’ noise sources may also contribute some entropy but this cannot be taken into account for the evaluation of requirement DRG.4.10.

232 [DRG.4.10] Like the seed material in the seeding procedure and reseeding procedure, the high-entropy additional input data (DRG.4.10) must also be protected to ensure secrecy, integrity, and authenticity; cf. application note 340. The verification of these properties is part of the overall evaluation of the TOE.

233 [DRG.4.10] Enhanced forward secrecy ‘on demand’ is triggered by the requesting application. ‘On condition’ may be a specified maximum number of generated internal random numbers (comprising less than or equal to 2^{17} bits) after the previous high-entropy additional input, the previous seeding procedure, or the previous reseeding procedure, while ‘after time’ requires that a reliable time measurement (real time) is available.
Note: 2^{17} internal random number bits correspond to 512 internal random numbers that consist of 256 bits each or to 1024 internal random numbers that consist of 128 bits each, for example.

234 [DRG.4.10] It is not necessary to interrupt an ongoing request when 2^{17} internal random number bits have been generated since the previous high-entropy additional input or since the previous seeding procedure/reseeding procedure. The current request can be completed, but the DRNG shall receive fresh high-entropy additional input before it generates further output.

3.4 PTRNGs: Functionality classes

235 Subsects. 3.4.3 and 3.4.4 define functionality classes PTG.2 and PTG.3, respectively. The differences from the previous versions of the AIS 31 [AIS2031An_11] are pointed out in Subsect. 3.4.1. Subsect. 3.4.2 contains explanations that are relevant for both PTG.2 and PTG.3. We begin with general remarks.

236 We distinguish between pure PTRNGs and hybrid PTRNGs. Roughly speaking, the security of a pure PTRNG is essentially based on the entropy of the raw random numbers (to which an appropriate post-processing algorithm needs to be applied (the identity mapping is principally possible)), whereas hybrid PTRNGs have two security anchors, namely entropy from the noise source and computational security, the latter provided by a cryptographic post-processing algorithm that by itself is a DRNG.

237 This classification is not sharp (and not relevant for the evaluation). Usually, pure PTRNGs apply non-cryptographic post-processing (e.g., algorithmic post-processing to increase the entropy per data bit), but cryptographic post-processing is also allowed. PTRNGs that use cryptographic constructions for their post-processing algorithm but not with memory (i.e., those constructions are not DRNGs) are generally considered to be pure PTRNGs because they become practically insecure if the noise source becomes weak or breaks down completely. Hybrid PTRNGs apply cryptographic post-processing, which according to the definition in this document, always means with memory. By data compression, at the cost of performance, cryptographic post-processing

may also serve to increase the entropy per bit, but usually its main purpose is to add an additional security layer that is based on computational security.

A PTRNG that is compliant to functionality class PTG.2 is basically a well-understood physical noise source that exploits physical phenomena that provide a quantified amount of entropy with very high assurance. Together with a total failure test and an online test, this allows the generation of internal random numbers with an entropy per bit that is very close to 1. Functionality class PTG.3 is basically a PTG.2-compliant PTRNG combined with DRG.3-compliant cryptographic post-processing. A pure PTRNG can be compliant to functionality class PTG.2 but cannot be compliant to class PTG.3 because of the requirement for the cryptographic post-processing. 238

For both functionality classes PTG.2 and PTG.3, high assurance shall be established by a stochastic model of the raw random numbers. The stochastic model describes the stochastic behavior of the raw random numbers and traces the behavior back to true physical randomness. The stochastic model enables statistical analysis and the quantification of the entropy of the raw random numbers. Furthermore, the model allows the verification of the effectiveness of the algorithmic post-processing with regard to the entropy per internal random number bit. 239

[combining several physical noise sources] In the following the document speaks of *the* physical noise source. However, functionality classes PTG.2 and PTG.3 also allow the use of several physical noise sources, e.g., several ring oscillators. 240

[combining several physical noise sources] If the PTRNG uses several physical noise sources the physical noise sources may influence each other. Thus, *all* physical noise sources have to be analyzed jointly, which justifies to view them as *one* (single) physical noise source. Separate analysis of the used physical noise sources is permitted only if it can be shown that the physical noise sources do not influence each other. 241

In order to maintain this high assurance over the entire life cycle, a PTRNG compliant to PTG.2 or PTG.3 is required to have total failure tests that detect total failures so quickly that no internal random numbers are output that were generated after the total failure has occurred. The concrete PTRNG design may allow relaxations (e.g., due to buffering, in particular for PTG.3-compliant PTRNGs); cf. Subsect. 4.5.4. Non-tolerable deviations from the desired behavior shall be detected sufficiently soon by online tests. 242

A PTRNG compliant to functionality class PTG.2 or PTG.3 delivers output with entropy per data bit very close to 1 with a high level of assurance; cf. par. 279 for justification. 243

A PTRNG compliant to functionality class PTG.3 additionally has the security properties of DRG.3. 244

It should be noted that the definitions of functionality classes PTG.2 and PTG.3 (as well as DRG.3) have been reworked in this version of the document. The definitions of functionality classes PTG.2 and PTG.3 and their objectives are similar to that in [AIS2031An_11] (which justifies maintaining the class names) although they are different in detail. An in-depth explanation of the differences to the previous definitions in [AIS2031An_11] can be found in Section 3.4.1 (for DRG.3: see Section 3.3.1). 245

3.4.1 PTRNG: Main Differences from [AIS2031An_11]

246 The previous document [AIS2031An_11] defines an additional functionality class, namely PTG.1. Functionality class PTG.1 claims only statistical properties but not any minimum entropy bound. Class PTG.1 has been withdrawn due to the lack of interest by the applicants.

247 In [AIS2031An_11] functionality class PTG.3 requires that the evaluator applies statistical tests (at least several specified blackbox tests) to the output of the cryptographic post-processing component. This requirement (as well as the corresponding requirement in functionality class DRG.3) has been relaxed.

248 Compared to [AIS2031An_11] for PTG.2-compliant PTRNGs, the tolerated entropy defect has become significantly smaller. This change was motivated by the fact that the certified PTRNGs show significantly smaller entropy defects than allowed in [AIS2031An_11]; cf. also par. 284. Furthermore, it should be noted that in this document the entropy claim refers to the internal random numbers whereas in [AIS2031An_11] the entropy claim refers to the raw random numbers and test suite A limits (e.g.) the bias of the internal random numbers. Moreover, this document also allows min-entropy claims in addition to or instead of Shannon entropy claims.

249 Unlike in [AIS2031An_11] class PTG.3 allows individual entropy claims (with small entropy defects) in both Shannon entropy and min-entropy.

250 The modifications to functionality class DRG.3 (compared to [AIS2031An_11]) also affect functionality class PTG.3. See Section 3.3.1.

3.4.2 PTG.[2,3]: Definitions, requirements, and justification

251 PTRNGs use physical noise sources (whereas NPTRNGs use non-physical noise sources).

252 The 'core' of a PTRNG is its physical noise source. The physical noise source extracts randomness from a physical phenomenon (or several phenomena). The digitization mechanism associated with the noise source generates raw random numbers from the (typically) analog signals derived from the physical phenomenon (or several phenomena). The digitization mechanism is considered to be a part of the physical noise source.

253 Physical noise sources exploit physical phenomena (thermal noise, shot noise, jitter, metastability, radioactive decay, etc.) from dedicated hardware designs (using diodes, ring oscillators, etc.) or physical experiments to produce digitized random data. The dedicated hardware designs can use general-purpose components (like diodes, logic gates, etc.) if the designer is able to understand, describe, and quantify the characteristics of the design that are relevant for the generation of random numbers.

254 Usually, the physical noise source of a PTRNG is part of an electronic circuit or is realized as a physical experiment. When integrated into an electronic circuit (e.g., a microchip), the physical noise source consists of dedicated hardware that has been designed for this purpose.

Examples: The physical noise source may employ Zener diodes, noisy oscillators, or ring oscillators. Or, it may exploit chaotic behavior, radioactive decay, or other quantum effects. This list of possible phenomena and design principles is not complete. We refer the reader to Sect. 5.4 for a more detailed treatment. 255

The central task of both PTRNG evaluations and NPTRNG evaluations is the verification that the (average) entropy per internal random number bit exceeds a specified lower bound. 256

While the physical noise source of a PTRNG is 'under the control' of the RNG designer, the non-physical noise source of a NPTRNG usually cannot be controlled by the RNG designer (cf. Subsec. 3.5.2). This is an important difference between PTRNGs and NPTRNGs, which has an impact on the depth of the evaluation. 257

The fact that the physical noise source is based on a dedicated hardware design allows (at least in principle) precise modeling because one may assume that the same types of physical noise sources in different devices behave similarly. However, the physical noise sources generally do not behave identically in a strict sense because even digital physical noise sources usually consist of analog components. Differences may, for example, be caused by component variance (inside certain tolerance levels), aging effects, or different environmental conditions; cf. pars. 272 to 273. 258

The analog values produced by the physical noise source are digitized at some point, providing the raw random numbers (a.k.a. das random numbers where 'das' stands for 'digitized analog signal'). The digitization mechanism can involve simple transformations (e.g., dropping bits) and the raw random numbers may undergo several separated post-processing operations. For this reason, there may be some ambiguity as to what intermediate product should be referred to as *the raw random numbers*. 259

The developer / applicant decides which data are to be called *the raw random numbers*. Both PTG.2 and PTG.3 require a verifiable stochastic model for the raw random numbers that traces their stochastic behavior back to a physical phenomenon / several physical phenomena). It is therefore strongly recommended to choose the earliest possible stage. The evaluator accepts or rejects the stochastic model and the corresponding rationale. 260

Examples 261

- The physical noise source samples noisy voltage at a high frequency and these values are digitized to 8 bits. In order to reduce statistical dependencies, the developer decides to discard every other byte. The developer then declares the remaining bytes to be the raw random numbers and provides a stochastic model describing their bias and statistical dependency as well as how they relate to the physical phenomenon causing the noise. This approach is principally permitted.
  Note: This operation lowers the output rate by factor 2. Alternatively, the unmodified sequence can be chosen as the raw random numbers, and discarding every second byte may be viewed as algorithmic post-processing.
- A battery of ring oscillators is sampled and their output fed into a (cryptographic) hash function with a large compression factor. The developer declares the resulting hashed bits

to be the raw random numbers and claims stochastic independence and uniform distribution as a stochastic model. This approach is *not permitted* because the stochastic properties of the raw random numbers cannot be traced back to a physical phenomenon.

262 Viewed as a mathematical function, algorithmic post-processing usually has a small domain and a small range.

263 Examples (of algorithmic post-processing algorithms): XORing bits or binary vectors, modular addition, linear feedback shift registers, and identity mapping.

264 The raw random numbers may or may not undergo algorithmic post-processing and / or cryptographic post-processing (finally resulting in the internal random numbers, i.e., the data ready for output). If the raw random numbers already 'contain' sufficient entropy per data bit to meet the PTG.2 requirements, then the designer may choose to omit a post-processing algorithm. In this case, the raw random numbers coincide with the internal random numbers. Formally, a nonexistent post-processing algorithm can be interpreted as the identity mapping. Examples of mathematical post-processing algorithms are discussed in Sect. 5.1; cf. also par. 263.

265 [PTG.3] PTG.3 is the strongest functionality class in AIS 20 and AIS 31.

266 [PTG.3] The usual technical realization of a PTG.3-compliant PTRNG is to use a PTG.2-compliant PTRNG whose internal random numbers are fed into a DRG.3-compliant cryptographic post-processing algorithm. The PTG.2-compliant PTRNG then is the central component of the PTG.3-compliant PTRNG. However, it is not mandatory to have a clear-cut 'PTG.2-boundary' within the PTRNG. Of course, the lack of a clear PTG.2-boundary does not waive or relax any requirements on the raw random numbers and on the entropy verification of the internal random numbers.

267 [PTG.3] The data that are input to the cryptographic post-processing algorithm are called intermediate random numbers. If the PTRNG has a PTG.2-compliant core (the usual design, cf. par. 266), the intermediate random numbers of the PTG.3 design are the internal random numbers of the PTG.2-compliant PTRNG.

268 [PTG.3] The cryptographic post-processing algorithm shall not 'expand' its input data (i.e., the intermediate random numbers). This means that the average output rate in bits of cryptographic post-processing algorithm shall not be larger than its input rate (in bits). That is, the ratio between the number of intermediate bits (required for one internal random number) and the bit length of an internal random number shall be ≥ 1. This is called the compression rate c_{rate} in the following. To increase the entropy per bit, the compression rate must be > 1. If the compression rate is < 1, the PTRNG cannot be compliant to class PTG.3 (but compliance to class DRG.4 is possible).

269 [PTG.3] Of course, cryptographic post-processing can only increase the entropy per bit if it compresses the input data. If the cryptographic post-processing algorithm can be modeled by a random mapping, the difference $\mathrm{c}_{\mathrm{diff}}$ = (#number of input bits - #number of output bits) is significantly more relevant for the increase of entropy in the output than the compression rate $\mathrm{c}_{\mathrm{rate}}$. This may seem surprising at first sight, but the reason is that the ratio between the sizes

of the domain and of the image space equals $2^{c_{\mathrm{diff}}}$. Section 4.4 treats this topic intensively. Note: Of course, given c_{rate} and m, it is easy to calculate n and c_{diff}. What is meant here is that (with regard to the entropy per bit) the case $(n, m) = (140, 128)$ is similar to $(n, m) = (268, 256)$ whereas $(n, m) = (280, 256)$ ensures a smaller entropy defect than $(n, m) = (140, 128)$ although c_{rate} is identical in both cases. (Expressed in other words: The term c_{diff} seems to be more illustrative quantity than c_{rate}.)

[PTG.3] If the cryptographic post-processing of a PTG.3-compliant PTRNG would run autonomously, it would be (algorithmically) compliant to functionality class DRG.3. That means, even if the PTG.2-compliant part of a PTG.3 (assuming the usual PTG.3 design) were suddenly to deliver predictable output, then the PTRNG would still have the security features of a DRG.3 compliant DRNG (because of the cryptographic post-processing). This does hold, of course, only under the assumption that the internal state of the cryptographic post-processing algorithm has already received enough entropy by the intermediate random numbers. This should be the case shortly after the PTRNG has been started. 270

[PTG.3] Note that the 'DRNG fallback' in the previous paragraph is an additional security layer. The requirements for PTG.2 and PTG.3 dictate a reliable online test (health testing) and a reliable total failure test that shall prevent undetected degradation of the entropy of the internal random numbers or an undetected total breakdown of the physical noise source (par. 278). Further beneficial effects of cryptographic post-processing are described in pars. 282 and 283. 271

Raw random numbers, intermediate random numbers, and internal random numbers are interpreted as realizations (i.e., of values taken on) of random variables. For the concept of randomness, random variables, and realizations we refer the interested reader, e.g., to Sect. 4.1. 272

For PTRNGs the entropy analysis shall be based upon a stochastic model. The stochastic model takes the concrete design of the physical noise source into account and models its stochastic behavior. Based on this behavior, the impact of algorithmic post-processing on the internal random numbers is analyzed. Blackbox testing of the raw random numbers or of the internal random numbers is not sufficient to assess their entropy. 273

The formulation, verification, and analysis of the stochastic model is the crucial part of a PTRNG evaluation. We refer the reader to detailed explanations in Sect. 4.5, and to Sect. 5.4 for illustrating examples. 274

When the PTRNG has been started, a start-up test shall check for a total failure and severe statistical weaknesses; cf. Subsect. 4.5.5. 275

The entropy per internal random number bit shall be large enough when the PTRNG is in operation. (For functionality classes PTG.2 and PTG.3 the indistinct term 'large enough' is quantified; cf. requirements PTG.2.2 and PTG.3.2.) The entropy claim shall be assured by an online test; cf. Subsect. 4.5.3. The effectiveness of the online test shall be verified based on a stochastic model of the physical noise source. 276

Assume that $Z_1, Z_2, \ldots$ are stationarily distributed binary-valued random variables. One way to 277 quantify k-step dependencies is to evaluate $|\operatorname{Prob}(Z_{j+k} = 0 \mid Z_j = 0) - \operatorname{Prob}(Z_{j+k} = 0 \mid Z_j = 1)|$.

278 During operation a total failure of the physical noise source can occur. A total failure would imply that future raw random numbers contain almost no entropy. A total failure test shall detect a total failure of the physical noise source virtually immediately (relaxations are possible, depending on the design of the PTRNG); cf. Subsect. 4.5.4. This means, a total failure test must detect a total failure in time to prevent the output of low-entropy random numbers.

279 Ideal RNGs do not exist. And even if ideal RNGs existed, it would be impossible to verify them. Thus, functionality classes PTG.2 and PTG.3 allow small entropy defects. Compliance to functionality classes PTG.2 and PTG.3 guarantees a lower entropy bound per random bit close to 1.

280 The PTG.2 class specification tolerates a small entropy defect, e.g., caused by a bias or by dependencies of the internal random numbers. For many cryptographic applications, e.g., for the generation of AES keys, challenges, IVs, etc. such defects should not practically impact security.

281 For some applications such defects yet might bear security risks. For ECDSA signatures, for example, the ephemeral keys are linked by an underdetermined system of linear equations over a finite field. An adversary might try to combine information from many signatures. Although no concrete attack is known to date that could leverage the small entropy defect allowed for PTG.2-compliant PTRNGs, at least in principle, this represents a security risk.

282 Consequently, the direct employment of PTG.2-compliant PTRNGs for arbitrary cryptographic applications is not recommended. Generally, PTG.2-compliant PTRNGs should be used to seed DRNGs or serve as a ‘core’ of a PTG.3-compliant PTRNGs.

283 Furthermore, the cryptographic post-processing of PTG.3-compliant hybrid PTRNGs should also increase their resistance to side-channel and fault attacks (e.g., induced transient breakdowns of the physical noise source or enforcing certain values). Implementation attacks are not covered by AIS 31, but, of course, are relevant in the overall evaluation of the TOE; cf. Sect. 2.1, par. 26.

284 Principally, the tolerated entropy defect defined in this document (i.e., PTG.2.3, PTG.3.6) could have been set even considerably smaller. We have refrained from doing so for two reasons: First of all this would have increased the requirements concerning the verification of the stochastic model. Furthermore, it would increase the difficulties of implementing efficient online tests (with reasonable sample sizes) that would effectively detect when the entropy falls below the specified entropy bound.

285 For functionality classes PTG.2 and PTG.3, the entropy of the raw random numbers can be quantified in Shannon entropy and / or in min-entropy. Shannon entropy is justified by the fact that the raw random numbers are stationary (cf. Sect. 4.3) and that the Shannon entropy satisfies useful functional equations (4.66) and (4.67), which simplify the entropy calculation for dependent random variables. For classes PTG.2 and PTG.3 entropy claims for the internal random numbers and the intermediate random numbers (only PTG.3) in Shannon entropy, in min-entropy, or in both Shannon entropy and min-entropy, are permitted. This is different from

functionality class NTG.1, which only allows min-entropy claims.

3.4.3 Functionality Class PTG.2

Functionality class PTG.2 defines requirements for physical RNGs. 286

Roughly speaking, PTG.2-compliant RNGs generate high-entropy internal random numbers. The entropy shall, in particular, prevent successful guessing attacks, but the internal random numbers may be practically distinguishable from ideal randomness (i.e., independent and uniformly distributed random numbers) when testing large amounts of data. 287

The TSF has to protect the internal state (if applicable) of the RNG from being compromised. 288

PTG.2-specific deliverables by the applicant 289
The security architecture description and developer evidence shall contain

- a description of the physical noise source (including the digitization mechanism),
- a comprehensive description of the 'algorithmic behavior' of the PTRNG, beginning with the digitization of the raw random numbers,
- a stochastic model of the raw random numbers with substantiated justification, statistical evidence, and thorough analysis,
- evidence that PTG.2.1 and PTG.2.2 are fulfilled,
- a description of the start-up test and evidence that PTG.2.3 is fulfilled,
- a description of the online test and evidence that PTG.2.4 is fulfilled,
- a description of the total failure test and evidence that PTG.2.5 is fulfilled,
- evidence that PTG.2.6 is fulfilled.

PTG.2: Security functional requirements 290
Security functional requirements of 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) 291

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

(PTG.2.1) The TSF shall generate raw random numbers that can be viewed as realizations of a (time-local) stationary stochastic process $R_1, R_2, \ldots$.

(PTG.2.2) The internal random numbers shall be interpreted as realizations of random variables $Y_1, Y_2, \ldots$. *If the random variables* Y_j *are binary-valued it shall be* $\mathrm{Prob}(Y_j = 1) \in (0.493, 0.507)$. *If the random variables* Y_j *assume binary vectors this condition shall be met by the projections onto the particular bits. Furthermore, it shall be guaranteed that [selection:*
the average Shannon entropy per internal random number bit is greater or equal than 0.9998*,*
the average min-entropy per internal random number bit is greater or equal than 0.98*,*
the average Shannon entropy per internal random number bit is greater or equal than 0.9998 *and the average min-entropy per internal random number bit is greater or equal than* 0.98*].*

(PTG.2.3) The start-up test shall be applied after the RNG has been started. It shall be designed to detect a total failure of the physical noise source and severe statistical weaknesses. The TSF shall not output any internal random numbers before the start-up test has been passed.

(PTG.2.4) The online test shall check the quality of the raw random numbers while the RNG is in operation. The online test shall detect non-tolerable entropy defects of the raw random numbers sufficiently soon. The TSF shall not output any internal random numbers if a non-tolerable entropy defect has been detected.

(PTG.2.5) The total failure test shall detect if a total failure of the physical noise source occurs while the PTRNG is in operation. The total failure test shall prevent the output of internal random numbers that depend on any raw random number that has been generated after the total failure of the physical noise source. If the PTRNG applies a cryptographic post-processing algorithm that is compliant to functionality classes DRG.2 or DRG.3, then this relaxes this requirement: Assume that t *equals the bit size of the effective internal state of the cryptographic post-processing algorithm. The total failure test shall prevent the output of more than* t *internal random numbers bits after the total failure of the physical noise source has occurred.*

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

(PTG.2.6) The raw random numbers shall pass test suite T_{rrn} *(cf. Subsection 4.6.2). The internal random numbers shall pass test suite* T_{irn} *(cf. Subsection 4.6.3)*

Application notes

292 [stochastic model, refers to PTG.2 and PTG.3] The evaluation of a PTRNG shall be based on a verifiable, substantiated stochastic model. There is only one level of detail in the description of the stochastic model, irrespective of the chosen EAL. We refer to Sect. 4.5, which provides additional information, illuminates the mathematical background, and discusses elementary examples of stochastic models. More complex 'real-world' examples of stochastic models can be found in Chapter 5.

293

[stochastic model, refers to PTG.2 and PTG.3] The evaluator checks the stochastic model, its justification (typically supported by empirical studies), and the analysis that has been provided by the applicant. The evaluator may perform additional tests if they feel that this might be relevant.

[stochastic model, refers to PTG.2 and PTG.3] Functionality classes PTG.2 and PTG.3 do not limit the 'depth' of possible dependencies of the raw random numbers. This is motivated by the fact that otherwise widespread types of physical noise sources would have been excluded. However, usually it should be easier to verify and to analyze stochastic models that show (at most) short-term dependencies. 294

[several physical noise sources, refers to PTG.2 and PTG.3] The functionality classes PTG.2 and PTG.3 allow to use several physical noise sources instead of one. In general, it is *not sufficient* to analyze these physical noise sources separately, unless it can be given evidence that the physical noise sources can be viewed as independent. Such an independence claim would be part of the security evaluation of the TOE. Otherwise, if mutual influence of the physical noise sources cannot be excluded, *all* physical noise sources have to be analyzed and evaluated jointly; cf. pars. 240 and 241. 295

[several physical noise sources, refers to PTG.2 and PTG.3] If already a single physical noise source provides enough entropy to meet the requirements of PTG.2 or PTG.3, respectively, and if it can be shown that the other physical noise sources do not negatively affect this physical noise source, then it suffices to evaluate this single physical noise source and disregard the other physical noise sources. 296

[PTG.2.1] The verification of requirement PTG.2.1 shall be based on the stochastic model. For the analysis of the stochastic model, usually (time-local) stationarily distributed random variables can be treated as if they were stationarily distributed; cf. Subsect. 4.5.1, pars. 668 to 670, time-local stationarity. 297

[PTG.2.1] If the raw random numbers are not binary-valued but ℓ-bit vectors, it may be the case that the sequence of bits, or more precisely, the corresponding random variables, are not stationarily distributed. This has to be taken into account in the analysis. 298

[PTG.2.2] The developer may claim the Shannon entropy bound, the min-entropy bound, or both. For both Shannon entropy and min-entropy, functionality class PTG.2 only allows fixed, class-specific values. The verification of the min-entropy claim may require additional efforts. 299
Note: Although the computed entropy values (based on the stochastic model) should normally exceed the specified entropy bounds significantly, class PTG.2 does not allow larger entropy claims; cf. par. 284.

[PTG.2.2] To verify requirement PTG.2.2, the effect of algorithmic post-processing on the entropy has to be taken into account. Algorithmic post-processing (if applied) does not need to be cryptographic. If the PTRNG has no post-processing, then this is formally interpreted as the identity mapping. 300

[PTG.2.2] Pars. 301 and 302 refer to both Shannon entropy and min-entropy. Min-entropy is 301 mentioned in brackets. If the Shannon entropy per raw random number bit is below 0.9998 (the min-entropy is below 0.98), the algorithmic post-processing algorithm must increase the average entropy per internal random number bit. This is not possible without data compression. The evaluator has to verify that the Shannon entropy per internal random number bit is ≥ 0.9998 (the min-entropy is ≥ 0.98). It is not necessary to quantify the exact entropy value.

302 [PTG.2.2] If the entropy per raw random number bit already is equal or larger than the PTG.2-specific boundary (Shannon entropy: 0.9998, min-entropy: 0.98), it suffices to show that the post-processing does not decrease the average entropy per bit. Usually, this is much easier than to quantify the gain of entropy per bit. An example is a post-processing algorithm (with memory) that is injective for each admissible value of the memory, and if the elements of the domain and the range have the same bit length. Then the post-processing algorithm maintains the (average) entropy per bit.
If the post-processing algorithm uses widely recognized cryptographic primitive (not necessary for class PTG.2), then post-processing it often can be modeled as a random bijection or as a random mapping, just as for classes DRG.2, DRG.3, and DRG.4; cf. pars. 109 to 114.

303 [PTG.2.2] Exemplarily, pars. 308 to 311 discuss requirement PTG.2.2 by two stochastic models. Note that in par. 308 to 311 the stochastic models *are only claimed but not justified*. Stochastic models are treated in detail in Sects. 4.5 and 5.4.

304 [PTG.2.2] The verification of requirement PTG.2.2 shall be supported by statistical tests of the raw random numbers. The tested raw random numbers shall be generated under representative relevant environmental conditions (cf. par. 322).

305 [PTG.2.2] This paragraph gives advice about how the Shannon entropy and the min-entropy can be computed for iid random variables and for Markov chains. We consider exemplarily three cases.
(i) The random variables $Z_1, Z_2, \ldots$ are Independent and identically distributed (iid) and $\text{Prob}(Z_j = 1) \in [0.4931, 0.5069]$. Then, $H(Z_j) > 0.9998$ and $H_{min}(Z_j) > 0.98$.
(ii) The random variables $Z_1, Z_2, \ldots$ form a homogeneous Markov chain on $\Omega = \{0, 1\}$ with state transition matrix P. The Shannon entropy and the min-entropy can be computed by (4.73) and (4.92), respectively. If, for example, $\text{Prob}(Z_j = 1) \in [0.494, 0.506]$ and $|\text{Prob}(Z_{j+1} = 0 \mid Z_j = 0) - \text{Prob}(Z_{j+1} = 0 \mid Z_j = 1)| \leq 0.001$, then $H(Z_j) > 0.99989$ and $H_{min}(Z_j) > 0.981$.
(iii) The random variables $Z_1, Z_2, \ldots$ form a homogeneous Markov chain on a finite state space Ω. Then (4.93) (in place of (4.92)) can be applied to determine a set of appropriate parameters that meet the min-entropy bound. If $|\Omega| > 2$ the min-entropy per internal random number bit (averaged over all bit positions) is relevant. If $Z_1, Z_2, \ldots$ form a 2-step Markov chain on Ω, then at first a (1-step) Markov chain has to be constructed as described in par. 537.

306 [PTG.2.2] Par. 305 can be applied to both the raw random numbers and the internal random numbers, or more precisely, to the corresponding random variables $R_1, R_2, \ldots$ and $Y_1, Y_2, \ldots$, if these random variables are iid or form a Markov chain. If the raw random numbers already fulfill requirement PTG.2.2 and if the post-processing neither reduces the Shannon entropy nor the min-entropy, the entropy claim can be directly transferred to the internal random numbers. If the PTRNG provides significantly more entropy than needed, it may be reasonable to apply non-optimal (but easy-to-prove) entropy estimates. To give an example: For Markov chains, the average gain of min-entropy per bit is trivially bounded from below by $-\log_2(\max_{i,j}\{p_{ij}\})$.

Of course, for Markov chains this coarse entropy estimation is not necessary because more accurate formulae (4.92) and (4.93) exist.

[PTG.2.2] It may be the case that even for unlocked test devices, the evaluator does not have access to the raw random numbers. This can constitute a serious (unsolvable) problem for the evaluation of a PTRNG that should have been considered during the design phase of the PTRNG. In consultation with the evaluator, the developer may try to capture the necessary data using external measurement equipment (e.g., a logic analyzer). In exceptional cases, it might be possible to alternatively test the internal random numbers instead, provided that this allows well-founded conclusions on the stochastic properties (e.g., entropy, bias, dependencies) of the raw random numbers. In any case the applicant must be able to define, to verify, and to analyze a stochastic model of the internal random numbers. 307
Note: This option is not recommended. The certification process will fail in practice if the above mentioned requirements are not fulfilled.

[PTG.2.2: Example A] The raw random numbers are interpreted as realizations of binary-valued random variables $R_1, R_2, \ldots$. On the basis of the stochastic model, the developer provides evidence that the random variables $R_1, R_2, \ldots$ are stationarily distributed. (Requirement PTG.2.1 is satisfied.) Furthermore, based on the stochastic model and supported by tailored statistical tests, the developer provides evidence a bias may exist, but no (significant) k-step dependencies for $k \geq 1$. In particular, the random variables $R_1, R_2, \ldots$ can be considered as iid. Assume that $\mathrm{Prob}(R_j = u) \in (0.5 - \epsilon_0, 0.5 + \epsilon_0)$ for $u \in \{0, 1\}$. 308

A.1 Assume that

$$\epsilon_0 = 0.003 \quad \text{or, equivalently,} \quad \mathrm{Prob}(R_j = 1) \in (0.497, 0.503)\,. \tag{3.5}$$

Then $H(R_j) \geq 0.99997$ (and $H_{\min}(R_j) \geq 0.991$).
Conclusion: If algorithmic post-processing does not reduce the entropy per bit, the PTRNG fulfills requirement PTG.2.2 (Shannon entropy claim and min-entropy claim).
Note: Alternatively, the developer could point to par. 305(i), saving their own calculations.

A.2 Assume that $\epsilon_0 = 0.03$.
Without data-compressing algorithmic post-processing, this PTRNG violates requirement PTG.2.2. XORing non-overlapping pairs of raw random number bits, i.e., $Y_1 = R_1 \oplus R_2, Y_2 = R_3 \oplus R_4, \ldots$, guarantees $\mathrm{Prob}(Y_j = 1) \in (0.4982, 0.5018)$, $H(Y_j) \geq 0.99999$, and $H_{\min}(Y_j) \geq 0.994$.
Conclusion: The PTRNG fulfills requirement PTG.2.2 (Shannon entropy claim and min-entropy claim).

[PTG.2.2] Assertion A.1 follows by substituting the least favourable parameters into the one-dimensional Shannon entropy formula (4.58) and the one-dimensional min-entropy formula (4.59). The second claim of Assertion A.2 follows from (5.2) with $k = 2$; cf. pars. 745 and 746. 309

[PTG.2.2: Example B] The raw random numbers are interpreted as realizations of binary-valued random variables $R_1, R_2, \ldots$. On the basis of the stochastic model, the developer gives evidence that the random variables $R_1, R_2, \ldots$ are stationarily distributed. (Requirement PTG.2.1 is satisfied.) Furthermore, based on the stochastic model and supported by tailored statistical tests, 310 the developer gives evidence that a bias and 1-step dependencies may exist but no (significant) k-step dependencies for $k \geq 2$. In particular, the random variables $R_1, R_2, \ldots$ can be considered as Markovian. Assume that $\text{Prob}(R_j = u) \in (0.5 - \epsilon_0, 0.5 + \epsilon_0)$ for $u \in \{0, 1\}$ and $|\text{Prob}(R_{j+1} = 0 \mid R_j = 0) - \text{Prob}(R_{j+1} = 0 \mid R_j = 1)| \leq \epsilon_1$.

B.1 Assume that

$$\epsilon_0 = 0.004\,, \quad \text{or, equivalently,} \quad \text{Prob}(R_j = 1) \in (0.496, 0.504)\,, \quad \epsilon_1 = 0.003 \tag{3.6}$$

Then $H(R_j \mid R_{j-1}, \ldots, R_1) = H(R_j \mid R_{j-1}) \geq 0.99995$ and $\frac{H_{\min}(R_{n+1}, \ldots, R_{n+m})}{m} \to_{m\to\infty} 0.9845$.
Conclusion: If the algorithmic post-processing does not reduce the entropy per bit, the PTRNG fulfills requirement PTG.2.2 (Shannon entropy claim and min-entropy claim).

B.2 Assume that $\epsilon_0 = 0.01$, and $\epsilon_1 = 0.012$.
Without using data-compressing algorithmic post-processing this violates requirement PTG.2.2, e.g., because the bias is too large. XORing non-overlapping pairs of raw random number bits (= algorithmic post-processing), i.e., $Y_1 = R_1 \oplus R_2, Y_2 = R_3 \oplus R_4, \ldots$, guarantees $H(Y_{n+1} \mid Y_1, \ldots, Y_n) \geq 0.99989$, and $H_{\min}(Y_j) \geq 0.9712$), and $\text{Prob}(Y_j = 1) \in (0.4938, 0.5062)$.
Conclusion: The PTRNG satisfies the Shannon entropy condition but not the min-entropy condition of PTG.2.2; cf. par. 311. The bias of the internal random numbers lies in the permitted interval.

311 [PTG.2.2] Assertion B.1 and the first claim of Assertion B.2 of par. 310 follow by substitution into the Shannon entropy formula and in the min-entropy formula for Markov chains; cf. par. 305(ii). The Shannon entropy claim and the distribution of Y_j in Assertion B.2 follow from (5.5) the inequation (5.6) with $k = 2$; cf. pars. 746 and 747. The result on the min-entropy was obtained as in (5.6). In particular,

$$\begin{aligned} &H_{min}(Y_{m+1} \mid Y_1, \ldots, Y_m) \geq H_{min}(R_{2m+1} + R_{2m+2} (\text{mod } 2) \mid R_m) \geq \\ &-\log_2 \left(\max\{p_{ij}p_{jk} + p_{i(1-j)}p_{(1-j)(1-k)} \mid 0 \leq i, j, k \leq 1\}\right)\,. \end{aligned} \tag{3.7}$$

provides a lower min-entropy bound. A larger min-entropy bound may be achievable but would require a more sophisticated approach.

312 [PTG.2.2] In par. 310, Example B.2, the entropy per bit is increased by XORing non-overlapping pairs of raw random numbers. Another option would be to thin out the raw random numbers by a factor of 2, i.e., by outputting only every second raw random number bit. This would also slightly increase the entropy per bit. The internal random numbers then would be Markovian with transition matrix P^2 in place of P.
Note: Thinning the raw random numbers out does not reduce the bias. Generally speaking, thinning the raw random numbers out is not very efficient unless the 1-step dependencies are rather large, but this would violate requirement PTG.2.2. It is an option, however, to thin out beforehand (as part of the digitization mechanism, equivalent to reducing sample rate) and interpret the resulting values as the raw random numbers.

313 [PTG.2.3] The start-up test shall be applied when the RNG is started after the TOE has been powered up, 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 physical noise source requires a warm-up period after powering up, the start-up test shall be applied after the warm-up period. It is important that no internal random numbers are output (or stored and output later) before the start-up test has been passed. The start-up test shall be designed to detect a total failure of the physical noise source and severe statistical weaknesses; cf. Subsect. 4.5.5. The start-up test can apply the online test, possibly with different evaluation rules; cf. Subsect. 4.5.5.

[PTG.2.4] When the PTRNG is in operation, the online test shall detect if requirement PTG.2.2 (or PTG.2.1) is violated. If a defect occurs, it should usually affect requirement PTG.2.2. This cannot (or at least not reliably) be achieved by blackbox testing without considering the nature of the physical noise source. Instead, the online test shall be tailored to the stochastic model, and its effectiveness shall be proven on the basis of the stochastic model. 314

[PTG.2.4] Of course, if the developer claims both Shannon entropy and min-entropy, the online test shall detect if any claim is violated. In particular, the developer needs to specify appropriate parameters. The online test shall detect sufficiently soon when the PTRNG leaves the specified set of appropriate parameters (implicitly given by the class requirements). If the PTRNG generates internal random numbers that have significantly more entropy than required, this usually simplifies the task of designing an effective (and efficient) online test. These aspects are discussed and explained in detail in Subsect. 4.5.3. 315

[PTG.2.4] The impact of the algorithmic post-processing algorithm (cf. par. 313) has to be analyzed in order to specify a set of appropriate distributions of the raw random numbers. An effective online test shall have a low probability of failing if the true distribution of the raw random numbers is appropriate (false positive) and a high probability to triggering a noise alarm if the distribution is inappropriate. The reader is referred to Subsect. 4.5.3 for detailed explanations. 316

[PTG.2.4] The online test should be applied to the raw random numbers. In exceptional cases it might be possible to test the internal random numbers instead. This requires that the applicant is able to determine the possible distributions of the internal random numbers, i.e., to formulate, justify, and analyze a stochastic model of the internal random numbers. This may be possible in favorable cases (e.g., for iid stochastic models with simple mathematical post-processing but usually the proofs will be more difficult than for online tests on the raw random numbers.
Note: This approach is not recommended and can fail in practice. 317

[PTG.2.4] The online test may be applied continuously, at regular (short) intervals, or upon specified internal events. The analysis shall take into account the calling scheme of the online test in the verification of its suitability. The applicant shall specify the consequences of a noise alarm. These aspects are also a subject of the evaluation. For general considerations, further explanations, and examples, we refer to Subsect. 4.5.3. 318

[PTG.2.5] A total failure of the physical noise source implies that without intervention, requirement PTG.2.2 would drastically be violated (e.g., because the next raw random number bits have no entropy at all or at best have very low entropy). If the internal random numbers are buffered before they are output, then this feature can relax the detection and reaction time. The effectiveness of the total failure test shall be proven on the basis of a substantiated failure 319

analysis of the physical noise source and the impact of the algorithmic post-processing on the entropy (cf. par. 313).
The total failure test may include statistical tests, but other solutions (voltage sensors etc.) may be acceptable as well. For general considerations, further explanations, and examples we refer to Subsect. 4.5.4.

320 [PTG.2.4, PTG.2.5] If the total failure test and / or the online test are not part of the TOE but are to be implemented later as an external security measure, then the applicant must submit an accurate specification of the online test and / or of the total failure test as well as a reference implementation. The tasks concerning the verification that PTG.2.4 and / or PTG.2.5 are fulfilled remain unaffected. The specification of the tests shall be part of the user manual (guidance documents). The online test of the final PTRNG implementation shall exactly fulfill the specification of the user manual (to be checked later in a composite evaluation) in order to be PTG.2-compliant.

321 [PTG.2.6] The statistical test suites T_{rrn} and T_{irn} (cf. Subsection 4.6.2 and 4.6.3) shall be applied under representative environmental conditions (cf. par. 322). Depending on the PTRNG design, the developer or evaluator may apply further statistical tests. For functionality class PTG.2, the importance of comprehensive statistical tests is incomparably higher than for classes DRG.2, DRG.3, DRG.4, and PTG.3, because the raw random numbers may be biased or have dependencies. Both defects can affect the statistical properties of the internal random numbers.

322 [environmental conditions] Environmental conditions (temperature, voltage, etc.) are viewed as relevant in this context if they belong to the specified range of permitted operating conditions. A parameter set (temperature, voltage, etc.) is representative if the tests under these environmental conditions allow drawing conclusions on the behavior of the raw random numbers (or the internal random numbers) under other environmental conditions within the permitted operating conditions.
Note: Within the vulnerability analysis one may perform tests for environmental conditions that lie outside of the permitted range. If the physical noise source works properly under these environmental conditions, too, then this may, to some extent, relax the requirements on the anti-tamper measures (e.g., by sensors). This is, however, not part of AIS 20/31; cf. pars. 22 and 23.

3.4.4 Functionality Class PTG.3

323 Functionality class PTG.3 defines requirements for hybrid PTRNGs. The differences from classes PTG.2 and DRG.4 are explained in par. 332.

324 Class PTG.3 is the strongest functionality class. It defines requirements for RNGs that are appropriate for any cryptographic application. Unlike for PTG.2-compliant PTRNGs, the security of PTG.3-compliant PTRNGs does not only rely on information-theoretic security ensured by the physical noise source (in combination with algorithmic post-processing) but, additionally, also on computational security ensured by cryptographic post-processing. In particular, the internal random numbers will not show any bias or short term dependencies. The cryptographic post-processing can reduce the entropy defect per intermediate random number bit by data

compression.

Functionality class PTG.3 demands a cryptographic post-processing algorithm that (interpreted as a DRNG) is DRG.3-compliant even if its input data, the intermediate random numbers, would become predictable at some point in time. Intermediate random numbers can be used as seed material (for the seeding procedure or reseeding procedure) or additional input. 325

The TSF has to protect the internal state of the RNG from being compromised. 326

PTG.3-specific deliverables by the applicant 327
The security architecture description and developer evidence shall contain

- a description of the physical noise source (including the digitization mechanism),
- a comprehensive description of the 'algorithmic behavior' of the PTRNG beginning with the digitization of the raw random numbers,
- a stochastic model of the raw random numbers with substantiated justification, statistical evidence, and thorough analysis,
- evidence that PTG.3.1, PTG.3.3, and PTG.3.5 are fulfilled,
- a description of the cryptographic post-processing and evidence that PTG.3.2, and PTG.3.4 are fulfilled,
- a description of the start-up test and evidence that PTG.3.6 is fulfilled,
- a description of the online test and evidence that PTG.3.7 is fulfilled,
- a description of the total failure test and evidence that PTG.3.8 is fulfilled,
- evidence that PTG.3.9 is fulfilled.

PTG.3: Security functional requirements 328
Security functional requirements of class PTG.3 are defined by component FCS_RNG.1 with specific operations as given below.

FCS_RNG.1 Random number generation (Class PTG.3) 329

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

(PTG.3.1) The TSF shall generate raw random numbers that can be viewed as realizations of a (time-local) stationary stochastic process $R_1, R_2, \ldots$.

(PTG.3.2) If the cryptographic post-processing algorithm runs autonomously or if its input data are known, the algorithm shall belong to functionality class DRG.3.

(PTG.3.3) *The intermediate random numbers that are input to the cryptographic post-processing algorithm [selection: are generated by a PTG.2-compliant PTRNG, are generated by a PTRNG that fulfills requirement PTG.3.1 and guarantees that [selection:*
the Shannon entropy per intermediate random number bit is greater or equal than [assignment: $v_1 \in [0.4, 0.9998]$)],
the min-entropy per intermediate random number bit is greater or equal than [assignment: $v_2 \in [0.1, 0.98]$)],
the Shannon entropy per intermediate random number bit is greater or equal than [assignment: $v_1 \in [0.4, 0.9998]$)] and the min-entropy per intermediate random number bit is greater or equal than [assignment: $v_2 \in [0.1, 0.98]$)].]
The intermediate random numbers are input into the cryptographic post-processing algorithm by the seeding procedure, the reseeding procedure, or as additional input. The PTRNG may apply several methods to include intermediate random numbers.

(PTG.3.4) *Cryptographic post-processing shall not expand its input sequence. In other words: The input rate (the intermediate random numbers, counted in bits) shall be greater or equal than the output rate (the internal random numbers, counted in bits). The* compression rate c_{rate} *is the ratio between input rate and output rate and it shall be ≥ 1. The compression shall be effective.*

(PTG.3.5) *The TSF shall guarantee that [selection:*
the (average) Shannon entropy per intermediate random number bit is greater or equal than 0.9998 *and the cryptographic post-processing does not expand its input,*
the (average) min-entropy per intermediate random number bit is greater or equal than 0.98 *and the cryptographic post-processing does not expand its input,*
the (average) Shannon entropy and min-entropy per intermediate random number bit is greater or equal than 0.9998 *and* 0.98 *and the cryptographic post-processing does not expand its input,*
the (average) Shannon entropy per internal random number bit is greater or equal than [assignment: $v_S \in [0.9998, 1 - 2^{-32}]$],
the (average) min-entropy per internal random number bit is greater or equal than [assignment: $v_m \in [0.98, 1 - 2^{-32}]$],
the (average) Shannon entropy per internal random number bit is greater or equal than [assignment: $v_S \in [0.9998, 1 - 2^{-32}]$] and the (average) min-entropy per internal random number bit is greater or equal than [assignment: $v_m \in [0.98, 1 - 2^{-32}]$]]

(PTG.3.6) *The start-up test shall be applied after the RNG has been started. It shall be designed to detect a total failure of the physical noise source and severe statistical weaknesses. The TSF shall not output any internal random numbers before the start-up test has successfully been passed.*

(PTG.3.7) *The online test shall check the quality of the raw random numbers while the RNG is being operated. The online test shall be designed to detect non-tolerable entropy defects of the raw random numbers sufficiently soon. The TSF shall not output any internal random numbers if a non-tolerable entropy defect has been detected.*

(PTG.3.8) *The total failure test shall detect if a total failure of the physical noise source occurs while the PTRNG is in operation. Assume that t equals the bit size of the effective internal state of the DRG.3-compliant cryptographic post-processing algorithm . The total failure test shall prevent the output of more than $\lfloor \lfloor t/c_{\mathrm{rate}} \rfloor / m \rfloor$ internal random numbers after the total failure of the physical noise source has occurred, where m denotes the bit length of the internal random numbers.*

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

(PTG.3.9) *The raw random numbers shall pass test suite T_{rrn} (cf. Subsection 4.6.2). The internal random numbers shall have statistical inconspicuousness. This conclusion shall be based on [selection: theoretical considerations, theoretical considerations supported by statistical tests, statistical tests with justification of the choice].*

Application notes

[PTG.3 vs. PTG.2] The general application notes in pars. 292, 293, 294 (stochastic model), 295 and 296 (several physical noise sources), and 322 (environmental conditions) apply to class PTG.3 as well. 330

[PTG.3 vs. PTG.2] Functionality class PTG.3 includes many of the requirements of class PTG.2. In particular, the requirements PTG.2.1 and PTG.3.1, PTG.2.4 and PTG.3.7, and PTG.2.5 and PTG.3.8 coincide. The corresponding application notes for functionality class PTG.2 apply to PTG.3, too. 331

[PTG.3 vs. PTG.2, PTG.3 vs. DRG.4] By using cryptographic post-processing rather than only algorithmic post-processing, class PTG.3 assures computational security even if the raw random numbers or the intermediate random numbers are compromised, provided that the internal state of the cryptographic post-processing algorithm has already received enough entropy by the intermediate random numbers, which should be the case shortly after the PTRNG has been started. In contrast to class DRG.4, class PTG.3 does not allow that the cryptographic post-processing 'extends' its input data. 332

[PTG.3.1] Requirement PTG.3.1 and PTG.3.2 are concerned with the raw random numbers from which the intermediate random numbers are generated. For the 'typical design' (where the intermediate random numbers are generated by a PTG.2-compliant 'inner' PTRNG), both requirements are / were already part of the PTG.2-evaluation. 333

[PTG.3.2] Requirement PTG.3.3 requires the evaluation of the algorithmic properties of the cryptographic post-processing algorithm with regard to functionality class DRG.3. More precisely, this is addressed by requirements DRG.3.3 to DRG.3.8. Some requirements from class DRG.3 need modifications. In DRG.3.2 the upper bound of the internal random numbers between two seeding procedures/reseeding procedures has been dropped for obvious reasons. In the context of functionality class PTG.3, requirement DRG.3.4 is concerned with the start-up of the PTRNG, i.e., before the PTRNG is permitted to output internal random numbers. The entropy may be introduced by the seeding procedure, reseeding procedure, or by inserting additional input. Seeding is permitted only once, when the PTRNG starts. 334

335 [PTG.3.2] The post-processing algorithm shall not be stateless, i.e., the state shall not be deleted after the generation of one or more random numbers because otherwise the PTG.3 loses its 'with memory' property, which constitutes a security feature. If the internal state is deleted, a startup test has to be applied before the next internal random numbers can be output (PTG.3.6). The intermediate random numbers shall influence the next internal state.

336 [PTG.3.3] Usually, a PTG.2-compliant PTRNG generates the intermediate random numbers for the cryptographic post-processing. For PTG.3-designs without an 'inner' PTG.2-compliant PTRNG, the Shannon entropy claim for the intermediate random numbers can be smaller than 0.9998 per bit, and / or the min-entropy claim can be smaller than 0.98 per bit (PTG.2-specific entropy bounds; cf. PTG.2.2). Higher entropy claims for the intermediate random numbers than in requirement PTG.2.2 are not accepted. The reasons are explained for functionality class PTG.2.

337 [PTG.3.3] The intermediate random numbers that are fed into the cryptographic post-processing algorithm shall be untampered with (integrity), authentic, and kept secret. This, of course, is essential for the entropy claim of the internal random numbers. Verification of these security claims is part of the overall evaluation of the TOE; cf. application note 231.

338 [PTG.3.4] Assume that the PTRNG outputs v internal random numbers (m-bit vectors) per intermediate random number (n_{in} bits). Here, $v = 1$ should be typical but $v > 1$ is possible. For a fixed internal state s, the cryptographic post-processing of an intermediate random number can be viewed as a mapping $\chi_s\colon \{0,1\}^{n_{in}} \to \{0,1\}^{v \cdot m}$, depending on the internal state s. The mapping χ_s describes the generation of v random numbers. Depending on the method for feeding the intermediate random numbers into the PTRNG, this mapping may include the seeding procedure or the reseeding procedure of the cryptographic post-processing algorithm, and / or the application of the state transition function.

339 [PTG.3.4, PTG.3.5] Functionality class PTG.3 allows the claim of a PTRNG-specific entropy bound. Therefore, in a first step the *input size* $n = \lfloor n_{in}/v \rfloor$ (counted in bits) per internal random number is determined. In pars. 340 to 345 the general strategy is explained by several examples. In case of doubt, the certification body should be contacted. The ratio $\mathrm{c}_{\mathrm{rate}} = n/m$ equals the (average) compression rate of cryptographic post-processing.

340 [PTG.3.4] Requirement PTG.3.4 separates PTG.3-compliant PTRNGs from DRG.4-compliant DRNGs because $n \geq m$ is demanded, or equivalently, $\mathrm{c}_{\mathrm{rate}} \geq 1$. To increase the (average) entropy per internal random number bit beyond the (average) entropy per intermediate random number bit it is necessary to apply a compression rate $\mathrm{c}_{\mathrm{rate}} > 1$.

341 [PTG.3.4] The compression rate $\mathrm{c}_{\mathrm{rate}}$ is computed per intermediate random number, i.e., the bit length n_{in} of an intermediate random number is divided by the number of internal random number bits (may belong to several internal random numbers) that are generated before the next intermediate random number is input. If the bit length of the intermediate random numbers is not constant, n_{in} is set to the minimal guaranteed bit size. In order to not make the evaluation too complicated, the use of a sliding average over several intermediate random numbers is not allowed.

342

[PTG.3.4, effective compression] The calculation of the compression rate c_{rate} alone is not sufficient. The following example may be pathological but points to the fundamental problem.
Example: Assume that the seed material used for the reseeding procedures of the cryptographic post-processing algorithm is generated by a PTG.2-compliant PTRNG and comprises 4096 bits. Assume further that the internal state of cryptographic post-processing algorithm only comprises 256 bits. Outputting 4096 internal random number bits after each reseeding procedure would result in $c_{\text{rate}} = 1$. However, the output of more than 256 internal random number bits is not permitted because the internal state obviously cannot 'store' more than 256 bits of entropy.
Note 1: The requirement that the effective compression rate is ≥ 1 is important if the developer decides for one of the three first selections in PTG.3.5. If in PTG.3.5 the developer decides for a concrete entropy claim this problem is covered anyway.
Note 2: This requirement is closely related to the narrowest internal width in [SP800-90B].

[PTG.3.4] PTG.3-compliant designs without a clear inner 'PTG.2-boundary' are also permitted 343
if the raw random numbers fulfill requirement PTG.2.1. If the entropy of the intermediate random numbers does not meet requirement PTG.2.2, this has to be compensated for by data compression; the case $n = m$ (or equivalently, $c_{\text{rate}} = 1$) is not permitted then. The lower entropy bound per intermediate random number bit has to be taken into account when determining an entropy bound for the internal random number bits.

[PTG.3.4] Example: We determine (n, m) and c_{rate} for several designs and explain the calcula- 344
tions. In this paragraph we assume that the cryptographic post-processing algorithm is given by the DRNG that is defined in pars. 803 and 804. We summarize its relevant features: It is $S = S_{req} = R = \{0,1\}^{256}$, and both the state transition function $\phi_{(H2)}\colon S \times A \times I \to S$ and the output function $\psi_{(H2)}\colon S_{req} \to R$ are closely related to the SHA-256 hash function. More precisely, if s and a denote the current internal state and the current intermediate random number (treated as additional input), the next internal state is given by SHA-256$(s\|11\|a)$ while the next internal random number equals SHA-256$(s\|00\|a)$. Requests are limited to 256 bits, the bit length of a single internal random number. Alternatively, the intermediate random numbers can be used as seed material for the seeding procedure or the reseeding procedure of the cryptographic post-processing algorithm; cf. requirement PTG.3.3. In this example, the seeding procedure and the reseeding procedure are rather simple: The first internal state s' is given by a bit string of length 256 (seeding procedure), or a 256-bit seed material string is XORed to the current internal state (reseeding procedure).
The internal state comprises 256 bits and thus cannot store more than 256 bits of entropy. Consequently, per request, i.e., between two consecutive applications of the state transition function, between two reseeding procedures, etc., only one internal random number can be output. Otherwise, requirement PTG.3.4 would be violated.
Furthermore, we assume that the intermediate random numbers are generated by a PTG.2-compliant PTRNG. We treat several examples.

[(i)] Per intermediate random number a with $|a| = 256$ (input either for the seeding procedure, reseeding procedure, or as additional input), one internal random number is output.
Conclusion: Requirement PTG.3.4 is fulfilled with $(n, m) = (256, 256)$, and thus $c_{\text{rate}} = 1$.

[(ii)] An intermediate random number a with $|a| = 320$ is input as additional input, and one internal random number is output.
Conclusion: Requirement PTG.3.4 is fulfilled with $(n, m) = (320, 256)$, and thus $c_{\text{rate}} = 320/256 = 1.25 \geq 1$.

[(iii)] 512-bit intermediate random numbers are used as seed material for the seeding procedure and the reseeding procedures. Two internal random numbers are output between two successive reseeding procedures.
Conclusion: It is $c_{\text{rate}} = 512/512 = 1$, but requirement PTG.3.4 is violated because the internal state cannot store more than 256 bits of entropy.
Note: The output of only one internal random number between two successive reseeding procedures would satisfy requirement PTG.3.4.

[(iv)] Periodically, intermediate random numbers a with $|a| = 320$ and $|a| = 270$ are used as additional input, and per request one internal random number is output.
Conclusion: Requirement PTG.3.4 is fulfilled with $(n, m) = (\min\{320, 270\}, 256)$, and thus $c_{\text{rate}} = 270/256 = 1.05 \geq 1$; cf. par. 343.

[(v)] Periodically, the additional input is given by intermediate random numbers a with $|a| = 1024$, $a = o$ ('empty string', i.e., no additional input), $a = o$, $a = o$, and after each additional input (including the empty additional inputs) an internal random number is output.
Conclusion: This design does not fulfill requirement PTG.3.4 because the internal state cannot 'store' more than 256 bits of entropy.
Note: Instead, a sequence of intermediate random numbers with $|a| = 1024, a = o$, etc. or with $|a| = 512, a = o$, etc. would be possible with $(n, m) = (256, 256)$, and thus $c_{\text{rate}} = 1$.
Justification: The model is as follows: The entropy of the first 256 bits of the non-empty intermediate random number is 'directly' used for the generation of an internal random number while its second 256 bits provides fresh entropy to the internal state. The second internal random number 'uses' this fresh entropy.

345 [PTG.3.4] Example: Below, three further examples are discussed. As in par. 344 we assume that the intermediate random numbers are generated by a PTG.2-compliant PTRNG.

[(i)] Hash_ DRBG, see Subsect. 5.3.1 An intermediate random number a with $|a| \geq outlen$ is input as additional input, and one internal random number is output.
Conclusion: Requirement PTG.3.5 is fulfilled with $(n, m) = (|a|, outlen)$, and thus $c_{\text{rate}} \geq 1$.
Justification: The additional input (here: an intermediate random number) is first mapped to an $outlen$-bit value $f(v, a)$, cf. (5.35), (5.37), (5.39). The entropy of $f(v, a)$ is limited by $outlen$ bits.

[(ii)] Hash_ DRBG, see Subsect. 5.3.1 An intermediate random number a with $|a| \geq seedlen$ is input by the seeding procedure and reseeding procedure.
Conclusion: If one internal random number is output, requirement PTG.3.5 is fulfilled with $(n, m) = (\geq seedlen, outlen)$, and thus $c_{\text{rate}} \geq seedlen/outlen \geq 1$.
Special case: For $Hash =$ SHA-384 we have $seedlen > 2 \cdot outlen$. If two internal random numbers are output, requirement PTG.3.5 is fulfilled with $(n, m) = (seedlen, 2 \cdot outlen) = (888, 768)$, and thus, $c_{\text{rate}} = 888/768 = 1.16 \geq 1$.
Justification: cf. Example (i)

[(iii)] The cryptographic post-processing is performed by the DRNG that is described in pars. 805 to 807. Per intermediate random number a with $|a| = 128$ (additional input), one internal random number is output.

Conclusion: Requirement PTG.3.5 is fulfilled with $(n, m) = (128, 128)$, and thus $c_{\mathrm{rate}} = 1$.
Note: Intermediate random numbers for which the entropy per bit is lower than defined in requirement PTG.2.2 (for PTG.2-compliant PTRNGs) cannot be used here; cf. par. 339

[PTG.3.5] Requirement PTG.3.1 is crucial, as it considers or even quantifies the entropy of the internal random numbers. The average entropy per intermediate random number bit shall be greater or equal than some specified threshold value. This pertains to Shannon entropy and min-entropy. For PTG.2-compliant 'inner' PTRNGs, this is covered by requirement PTG.2.2. A min-entropy claim for the internal random numbers is only possible if there is a (verified) min-entropy claim for the intermediate random numbers. 346

[PTG.3.5, Typical design] Usually, a PTG.2-compliant PTRNG generates the intermediate random numbers that are used as input for cryptographic post-processing. The PTRNG then can be viewed as a composition of PTG.2-compliant PTRNG and DRG.3-compliant cryptographic post-processing algorithm. If the applicant is satisfied with the lowest entropy claim of requirement PTG.3.5, i.e., that the cryptographic post-processing does not expand the input data (intermediate random numbers), the verification of requirement PTG.3.5 is easy because no entropy analysis of the impact of cryptographic post-processing is required. Specified entropy claims for the internal random number bits require entropy analysis. Various aspects are covered in the next paragraphs. 347

[PTG.3.5] For algorithmic post-processing the stochastic model of the raw random numbers has to be taken into account. In contrast, the evaluation of the cryptographic post-processing does not need to consider the stochastic model of the intermediate random numbers or of the underlying raw random numbers, respectively. Instead, only the entropy claim of the intermediate random numbers is taken into account. This allows composite evaluations where, for example, a software developer uses the output of a certified PTG.2-compliant PTRNG that was designed and manufactured by another company. The software developer does not need to know any details of the PTRNG design (the usual scenario). If applicable they have to implement specifications from the user manual (e.g., concerning the online test or the total failure test; cf. application note 320). Of course, within the composite evaluation, the designer and evaluator have to give evidence that the design fulfills the missing requirements of class PTG.3, in particular PTG.3.2, PTG.3.4, and the second part of PTG.3.9. 348

[PTG.3.5] For a given internal state s, the cryptographic post-processing of an intermediate random number can be viewed as a mapping $\chi_s \colon \{0,1\}^n \to \{0,1\}^m$ that is parametrized by the internal state s. The letters n and m denote the bit length of the intermediate random numbers and of the generated internal random numbers ($n \geq m$ by requirement PTG.3.4). This model applies to all admissible techniques (i.e., seeding procedure, reseeding procedure, or inserting additional input). The Shannon entropy claim refers to the average entropy per bit (averaged over a sequence of internal random numbers), while the min-entropy claim holds for the entropy per bit of any internal random number (averaged over all bit positions) with probability $\geq 1 - 2^{-16}$. 349
Note 1: Since the internal state changes permanently, smaller entropy values for some internal random numbers average out.
Note 2: Class PTG.3 only considers the gain of entropy for the sequence of generated internal random numbers that is caused by the respective intermediate random numbers. That is, here we pessimistically assume that an adversary knows the current internal state s. Relative to an

adversary who does not know at least a few bits of the internal state, the situation is even more favourable. Since it is assumed that an adversary does not know the internal state, this approach might seem to be overly cautious. Note that this worst-case approach also covers scenarios where an adversary is able to temporarily compromise the internal state.

350 [PTG.3.5] To verify an entropy claim for the internal random numbers, the cryptographic post-processing algorithm usually has to be modeled; bijective mappings are an exception, but they do not increase the entropy per bit. Usually, cryptographic post-processing can be modeled as a random mapping $\chi_s\colon \{0,1\}^n \to \{0,1\}^m$; see Sect. 4.4.2 for a comprehensive treatment. In order not to overstress the model, the entropy claim per internal random number bit is bounded by $1 - 2^{-32}$.

351 [PTG.3.6, $n = m$] For $n = m$, i.e., for $c_{\mathrm{rate}} = 1$, the cryptographic post-processing cannot increase the entropy per bit. For many designs the cryptographic post-processing algorithm $\chi_s\colon \{0,1\}^n \to \{0,1\}^n$ can be viewed as a random mapping, e.g., when the post-processing algorithm hashes an input vector that includes the current internal state and the intermediate random number. In this case the average entropy per bit even decreases to some degree. Note that for any $a_2 \in \{0,1\}^m$, the pre-image size $|\chi_s^{-1}(\{a_2\})|$ can be interpreted as a realization of a random variable that is Poisson distributed with parameter $\tau = 1$. The entropy claims shall consider Sect. 4.4.2, pars. 602 ff. Although here cryptographic post-processing even decreases the entropy per bit to some degree, it has positive effects on the practical security. This is because it removes possible bias and short-term dependencies of the intermediate random number by 'smearing' the weaknesses over the internal random number, thereby counteracting practical attacks and increasing the computational security.
Note: If the mapping $\chi_s\colon \{0,1\}^n \to \{0,1\}^n$ is bijective for each s, it maintains the entropy of the intermediate random numbers; cf. pars. 805 to 807, and par. 352, Example (iii).

352 [PTG.3.5, $n > m$] For $n > m$, the cryptographic post-processing applies data compression, which has to be taken into account for the verification of the entropy claim; cf. Sect. 4.4.2. Assume that the cryptographic post-processing algorithm can be modeled by a random mapping (usual case). To determine a lower min-entropy bound per internal random number bit, the following procedure can be applied:
First, $n^- := \lfloor n \cdot h_m \rfloor$ is computed where h_m for the moment denotes the min-entropy per intermediate random number bit. That is, for a given bit length n of the intermediate random numbers, the designer / evaluator applies (4.163) in the opposite direction to determine an input bit length n^- for a fictitious PTRNG design, for which the intermediate random numbers are generated from an ideal RNG and the post-processing χ_s is given by a random mapping, As explained in Subsect. 4.4.2, the real-world PTRNG is at least as good as the fictitious PTRNG. If $n^- \geq m + 16$ in a second step (4.162) can be applied with $z = z_{16}$; cf. par. 613. This provides an upper bound for the entropy defect per internal random number bit of the PTRNG under evaluation. Par. 353 illustrates the procedure by an example.
Note: The condition $n^- \geq m + 16$ results from the condition $\frac{2^{n^- - m}}{m \log(2)} \longrightarrow \infty$ (par. 610) and the normal approximation of the pre-image sizes $f^{-1}(a_2)$ (par. 618). The case $m < n- < m + 16$ is not categorically excluded but requires additional evidence from the applicant.

353 [PTG.3.5, $n > m$] Example: Assume that n-bit intermediate random numbers are generated by a (certified) PTG.2-compliant PTRNG with the min-entropy claim ≥ 0.98. Assume further that $(n, m) = (327, 256)$. We first conclude that the min-entropy per intermediate random number

exceeds $327 \cdot 0.98 = 320.45 > 320 =: n^-$. Applying (4.162) with $(n^-, m, z = z_{16})$ gives an upper bound for the min-entropy defect per internal random number bit of $2^{-32.93}$ (cf. Tab. 4, last column) that is only exceeded with probability 2^{-16} for a fictitious ideal RNG. This justifies the following min-entropy claim for the (real) PTRNG under evaluation: The min-entropy per internal random number bit exceeds $1 - 2^{-32.93}$.

[PTG.3.5, $n > m$] Pars. 352 and 353 discuss designs where the data-compressing cryptographic post-processing is applied that can be modeled by a random mapping and where $n^- > m + 16$. This should cover the usual designs. If $m < n^- < m+16$ the applicant has to provide arguments of their own that support their entropy claim. 354
Note: This is also the case, of course, if the cryptographic post-processing algorithm, cannot be modeled by a random mapping. An example is given if the pre-images $\chi_s^{-1}(\{a_2\})$ have the same size for all $a_2 \in \{0,1\}^m$. (Identical pre-image sizes should have positive impact on the entropy claim.)

[PTG.3.8] Requirement PTG.3.8 customizes PTG.2.5 to the given situation. In particular, it takes the compression rate $\mathrm{c_{rate}}$ (introduced in requirement PTG.3.4) into account. 355

[PTG.3.9] For PTRNG designs for which the intermediate random numbers are generated by a PTG.2-compliant PTRNG (the standard case, cf. par. 338), the first part of requirement PTG.3.9 (concerning the test suite T_{rrn}) has already been covered in the evaluation of the PTG.2-compliant PTRNG (cf. PTG.2.6). Otherwise, the test requirements on the raw random numbers remain unchanged, but the statistical tests on the intermediate random numbers (after an algorithmic post-processing algorithm, if existent) are waived. 356

[PTG.3.9] Concerning the second part of PTG.3.9, the requirement PTG.3 inherits the properties of the DRG.3-compliant cryptographic post-processing; cf. DRG.3.10. 357

3.5 NPTRNG: Functionality classes

Subsect. 3.5.3 defines functionality class NTG.1. The differences from the previous version of the AIS 31 [AIS2031An_11] are pointed out in Subsect. 3.5.1. Subsect. 3.5.2 contains explanations that are relevant for functionality class NTG.1. We begin with general remarks about NPTRNGs. 358

NPTRNGs generate 'true' random bits, but unlike PTRNGs they do not employ dedicated hardware designs or physical experiments as noise sources. Instead, NPTRNGs prevalently exploit non-physical noise sources such as system data or human interaction. From the point of view of the RNG, these non-physical noise sources may be viewed as 'external', although they belong to the device or are exploited by the device on which the NPTRNG is implemented. The distribution of the output data from non-physical noise sources (i.e., raw random numbers) usually cannot be modeled as precisely as the raw random numbers generated by dedicated physical noise source designs of PTRNGs. Thus, their entropy shall be conservatively estimated. 359

NPTRNGs are used to generate 'true' random numbers if PTRNGs with dedicated physical noise sources are not available. If one compares the evaluation of NPTRNGs with the evaluation of PTRNGs, certain fundamental differences become apparent. First of all, non-physical noise 360

sources used by NPTRNGs often only work well under specific circumstances and the NPTRNG often is unable to check whether these conditions are met. Secondly, the entropy estimate is usually based on complex assumptions about the knowledge and capabilities of an adversary and the operational environment (cf. pars. 366 and 369). As a consequence, functionality class DRG.4 prohibits the use of NPTRNGs for the seeding procedure, the reseeding procedure, and for high-entropy additional input.
Note: For these reasons, in general BSI has lower trust in NPTRNGs than in PTRNGs.

361 It should be noted that the definition of functionality class NTG.1 has been reworked in this version of the document. The definition of functionality class NTG.1 and the objectives are similar to that in [AIS2031An_11] (which justifies maintaining the class names), although it is different in detail.

3.5.1 NPTRNG: Main Differences to [AIS2031An_11]

362 In [AIS2031An_11] functionality class NTG.1 requires that the evaluator applies statistical tests (at least several specified blackbox tests) to the output of the cryptographic post-processing algorithm. As for functionality classes DRG.2, DRG.3, and DRG.4, the requirements concerning statistical test suites for the internal random numbers during the evaluation have been relaxed.

363 The entropy of the internal random numbers is measured in min-entropy. The tolerated min-entropy defect is numerically significantly smaller than the Shannon entropy defect in [AIS2031An_11].

364 In this document the requirement of mutual disjointness of random vectors (requirement NTG.1.4 in [AIS2031An_11]) has been dropped. A similar change has been made to functionality classes DRG.2, DRG.3 and DRG.4; cf. Sect. 3.3, pars. 93 to 95.

3.5.2 NTG.1: Definitions, requirements, and justification

365 Typically, NPTRNGs are used if ‘true’ random numbers are needed (e.g., for generating cryptographic keys or (re-)seeding a DRNG), but no PTRNG is available. Therefore, NPTRNGs use any available noise sources that are hard to predict. Since NPTRNGs are often computer programs, output data of non-physical noise source are usually system data and data produced by the interaction of human users or other external entities. A common approach is using time stamps from a high-resolution timer at ’random’ points in time.

366 For any TRNG evaluation, the central task is to verify that the entropy per internal random number bit exceeds a specified lower bound. For NPTRNGs this means assessing how much entropy the collected raw random numbers contain relative to external observers. Unlike in the case of physical noise sources, the unpredictability of the raw random numbers collected by a NPTRNGs from non-physical noise sources may, to a large degree, depend on the platform and the operational environment. Furthermore, the unpredictability depends on the means of an adversary to monitor or influence the noise sources. As a consequence, the entropy estimation of an NPTRNGs is usually not based on a precise stochastic model but instead on conservative

estimates assuming (realistic) worst-case conditions.

In analogy to physical noise sources the output data of non-physical noise sources are called raw random numbers. 367

Attack paths and risks (Examples) 368

1. A software-implemented NPTRNG is not secure against an adversary with full system access (in particular: read access to the internal state). An adversary with a lower access level can try to monitor the noise sources (e.g., perform coarse time measurements) or generate predictable raw random numbers (e.g., by running an unprivileged process on the same CPU core).
2. An adversary may connect a malicious device that generates predictable events (e.g., keyboard and mouse events or network traffic).
3. A software NPTRNG may be operated in an environment for which it was not intended, i.e., on a CPU where instructions behave differently, in a virtual machine or in a scenario where no or only a subset of the noise sources are present.

In order to determine what an adversary can and can't do, as well as stating necessary operational conditions (e.g., the adversary's privileges), the security boundary of the TOE must be precisely specified. It should be noted that NPTRNGs usually have more and stronger operational security requirements than PTRNGs (cf. par. 366); cf. Sect. 5.6, par. 1166, for example (Linux /dev/random). Furthermore, the attacker should not have root access privileges. 369

The raw random numbers collected by an NPTRNG are often huge in data size compared to their estimated entropy. Since the entropy estimate usually is made with heuristic rules that assume a (realistic) worst-case scenario, the raw random numbers may contain more entropy in practice (e.g., if real-world adversaries are not as knowledgeable as assumed or if the noise sources are 'more random' than assumed). Thus, NPTRNGs often compress and mix the collected raw random numbers into a large intermediate data structure, the entropy pool, in order to reduce the data size while still preserving the extra entropy. When the NPTRNG generates internal random numbers, data from the entropy pool is extracted and possibly compressed again such that the estimated entropy per internal random number almost equals the bit length. 370

An NPTRNG shall not generate more internal random number bits than the estimated overall entropy of the collected raw random numbers. NPTRNGs usually feature an entropy counter to keep track of how much entropy has entered the entropy pool and how much entropy has been extracted; the latter value corresponds to the number of internal random number bits that have been output. The counter is capped at the maximum amount of entropy that the mixing function can insert into the entropy pool. If a request for random bits exceeds the available amount of entropy contained in the entropy pool, the NPTRNG shall block the output. The request can be served only after sufficient additional entropy has been gathered. 371

[generic design] There are many conceivable designs of NPTRNGs. This paragraph describes a generic design that uses typical components. The design is exemplary, and variations are possible. 372

- The non-physical noise sources are used to gather or generate raw random numbers from system data or interaction with external entities. The entropy of the raw random numbers is estimated.
- Periodically, or driven by events, the raw random numbers are mixed into the entropy pool. The entropy counter is increased accordingly (taking into account the maximal amount of entropy that the entropy pool can store).
- Upon a request for random bits, data from the entropy pool is extracted. If the entropy pool contains insufficient entropy, the request is refused or blocked (suspended). After data extraction the entropy counter is decreased accordingly. To achieve enhanced backward secrecy, the previous value of the entropy pool is erased or cryptographically overwritten after each output (cf. par. 375).
- The function that extracts data from the entropy pool and generates output (internal random numbers) can be stateless (e.g., simply a hash function) or can have memory that persists between calls (e.g., a DRNG for cryptographic post-processing). The use of a DRNG does not waive the requirement of blocking in case of insufficient entropy.

373 [NTG.1] As an analogy to functionality class PTG.3, we call the data that are input to the cryptographic post-processing algorithm intermediate random numbers.

374 For stateful extraction, stateful output functions, or multiple entropy pools, the entropy must be counted consistently over all data structures to prevent generating pseudorandom output (cf. par. 371). This can be accomplished by having multiple entropy counters or fixed transfer sizes. Note that in order to achieve enhanced backward secrecy, the previous values of *each* data structure involved in generating output need to be erased or cryptographically overwritten (cf. par. 375) after providing entropy to the next stage.

375 Similar to pure PTRNGs without cryptographic post-processing algorithms, an NPTRNG can principally be stateless, i.e., collect a certain amount of entropy, generate output, and then completely erase its internal state in order to achieve enhanced backward secrecy. It is, however, recommended (and required for functionality class NTG.1) that an internal state be maintained in such a way that the NPTRNG exhibits the computational security properties of a (properly seeded) DRNG. This provides an additional security anchor in case the entropy of some raw random numbers is overestimated.

376 To be 'secure' even if at some point in time the raw random numbers of the non-physical noise sources do not contain enough entropy), a DRNG within the NPTRNG must be properly seeded. In particular, it must receive a sufficient amount of entropy before generating output. Otherwise, even if the raw random numbers would contain some entropy, the NPTRNG, when viewed as a hybrid DRNG, would potentially be susceptible to the generic guessing attack described in par. 171.

377 A DRNG security anchor can be achieved, for example, by designing the NPTRNG to be a DRNG with the internal state being the entropy pool. Alternatively, the NPTRNG can consist of an entropy pool for collecting entropy and a dedicated DRNG for cryptographic post-processing that is continually reseeded from the entropy pool. In order to achieve enhanced backward secrecy, the entropy pool as well as the internal state of the DRNG need to be updated after having

generated output (cf. par. 374). Using a dedicated DRNG for cryptographic post-processing can simplify the security evaluation of DRNG properties.

The concept of having a dedicated DRNG in par. 377 roughly corresponds to building a PTG.3 by combining a PTG.2 with DRG.3-compliant cryptographic post-processing. 378

For a long time a prominent example of an NTG.1-compliant NPTRNG (under suitable operational conditions) has been the mechanism behind /dev/random; to be precise, until Linux kernel version 5.5; cf. [Linux_RNG_overview]. In later kernel versions, /dev/random delivers pseudorandom bits. Under suitable assumptions /dev/random is compliant to functionality class DRG.3; see Sect. 5.6 for details. The document [RNG_virtual_env] considers the generation of random numbers in virtualized environments. 379

3.5.3 Functionality Class NTG.1

Functionality class NTG.1 defines requirements for NPTRNGs that rely on information-theoretic security (similar to PTRNGs) but use external input signals as noise sources. Additionally, a suitable cryptographic post-processing algorithm shall provide an additional security anchor. 380

NTG.1-compliant NPTRNGs are usually operated on devices like PCs, servers, etc. that do not have access to a PTRNG. 381

The TSF has to protect the internal state of the RNG from being compromised. 382

NTG.1-specific deliverables by the applicant 383
The security architecture description and developer evidence shall contain

- a description of the required operational conditions and a specification of the security boundary,
- a description of the noise sources and a justification for entropy estimates,
- a comprehensive description of the 'algorithmic behavior' of the NPTRNG,
- evidence that NTG.1.1 through NTG.1.6 are fulfilled.

3.5.4 Security functional requirements for the NPTRNG class NTG.1

Security functional requirements of class NTG.1 are defined by the component FCS_RNG.1 with specific operations as given below. 384

FCS_RNG.1 Random number generation (Class NTG.1) 385

FCS_RNG.1.1 The TSF shall provide a non-physical true RNG that implements:

(NTG.1.1) The NPTRNG shall collect and test the raw random numbers provided by the used noise sources in order to estimate the entropy and detect failures of the noise sources.

(NTG.1.2) The NPTRNG shall have an entropy pool and an entropy counter that tracks the estimated amount of entropy currently stored in the entropy pool. The NPTRNG shall never generate more internal random number bits than indicated by the entropy counter.

(NTG.1.3) The NPTRNG shall apply a cryptographic post-processing algorithm with memory. Viewed as a hybrid DRNG, the NPTRNG is compliant to functionality class DRG.3 (cf. pars. 375, 376, and 377). The fresh entropy can be input by the seeding procedure, the reseeding procedure, or as additional input. The NPTRNG may apply several of the above-mentioned methods to input fresh entropy.

(NTG.1.4) The NPTRNG shall not generate any random numbers until the following condition has been met. The entropy pool has collected at least 220 bits of min-entropy from at least two different noise sources each. These two noise sources shall employ different principles to provide randomness. Viewed as a DRNG, the NPTRNG has been seeded using contributions from the two noise sources.

(NTG.1.5) The estimated min-entropy per internal random number bit shall exceed [assignment $v \in [0.98, 1 - 2^{-32}]$ *].*

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

(NTG.1.6) The internal random numbers shall have statistical inconspicuousness. This conclusion shall be based on [selection: theoretical considerations, theoretical considerations supported by statistical tests, statistical tests with justification of the choice].

Application notes

386 [NTG.1.1] An NTG.1-compliant NPTRNG may utilize any source of data as noise source for which there is a compelling technical explanation why the data are hard to predict by an adversary. The explanation shall specify the necessary operational requirements for the noise sources to function (e.g., the type of CPU, whether virtualization is allowed, or assumptions regarding the security features that protect against an adversary) and deliver a conservative lower bound for the expected amount of entropy.
Note: By requirement NTG.1.5, an NTG.1-compliant NPTRNG needs at least access to two noise sources.

387 [NTG.1.1] The explanation shall specify all possible failure modes for the noise sources. The explanation shall comprise a heuristic analysis of the noise sources as a justification for the entropy estimator during operation.

388

[NTG.1.1, overall evaluation] The explanation shall survey conceivable attack vectors on the noise sources (cf. par. 368) and assess (under realistic assumptions) the ability of an adversary to influence or observe the data and the impact on its entropy.

[NTG.1.1] The entropy estimator may assign a constant value as an entropy estimate to data from a noise source (unless a failure has been detected) or, alternatively, heuristically determine an entropy value. Sets of data that are estimated to contain zero entropy may also be added to the entropy pool unless this weakens the security of the NPTRNG (cf. par. 390). 389

[NTG.1.1, overall evaluation] The evaluator shall consider the different noise sources and an adversary's ability to weaken the NPTRNG through the insertion of malicious raw random numbers. An adversary outside the security boundary shall not be able to weaken the security of the NPTRNG, provided that the operational requirements (e.g., necessary assumptions made for the evaluation that become part of the user guidance) are met. 390

[NTG.1.2] The entropy counter of the entropy pool shall start from zero. It shall be increased by the estimated amount of entropy provided by the raw random numbers that are mixed into the entropy pool and decreased by the bit length of the output random numbers (intermediate random numbers) when the bits are extracted from the entropy pool. The value of the entropy counter shall never exceed the maximal amount of entropy that the entropy pool can store. The maximal amount of storable entropy is determined by the data size of the entropy pool as well the function that transfers entropy from raw random numbers into the entropy pool. 391
Note: There are no concrete regulations for the mixing function (i.e., update function) of the entropy pool.

[NTG.1.2] If the NPTRNG has separate logical data structures in which entropy is stored (i.e., entropy pools), then the (local) entropy counter for each entropy pool and the (global) entropy counter for the sum of the entropy in all the entropy pools must be kept consistent (cf. par 374). Note that NTG.1.5 (minimal amount of entropy of the internal random numbers) implies that extracting entropy from multiple data structures in order to produce output also needs to be kept consistent. 392

[NTG.1.3] The documentation to be provided by the developer for a DRG.3 evaluation comprises a formal description of how the DRNG updates its internal state and generates output. The same modeling is thus required for an NTG.1 evaluation; see par. 377. The internal state of the cryptographic post-processing algorithm can coincide with the entropy pool, but this need not be the case. 393
Note: In the first case, the internal state is updated by raw random numbers or (statelessly) postprocessed raw random numbers.

[NTG.1.3, NTG.1.4] Requirements NTG.1.3 and NTG.1.4 shall ensure that the NPTRNG is at least as secure as a properly seeded DRG.3 (cf. par 375). To fulfill them, the corresponding requirements of DRG.3 shall be checked with the following modification. Because of the issues described in pars. 366 and 369, it is required to use two different noise sources that shall each provide a sufficient amount of data for the seeding procedure. In order to increase resilience, it may be advisable to collect even more data and from more noise sources before generating internal random numbers. 394

395

[NTG.1.4] This requirement for the seeding procedure of the DRNG security anchor also stipulates that the NPTRNG shall have at least two different noise sources. It also means that the NPTRNG cannot generate random bits until the slower of the two noise sources has produced the required amount of entropy. If the NPTRNG possesses more than two noise sources, requirement NTG.1.4 is satisfied when the two fastest noise sources have each contributed enough entropy. The amount of entropy contributed by any other noise sources may be less than that provided by the two fastest noise sources.

396 [NTG.1.4] Seeding the entropy pool requires bits from at least two different noise sources and thus the *second fastest* noise source determines the delay until the first random number can be generated. After this seeding step, it is not required to wait until the second fastest noise source has produced enough entropy. If the NTG.1 has a noise source that delivers a lot of entropy per time period, it may then continue to produce output with this (or even greater) bandwidth. This means that after the seeding step only the total entropy is relevant, regardless of how many noise sources have contributed.
Note: In order to increase resilience and not to depend on a single noise source, however, it may be advisable to prevent designs where the NPTRNG is dominated by a single noise source whose entropy rate greatly exceeds that of the other noise sources.

397 [NTG.1.4] NPTRNGs often exploit time points or time intervals of random events. Different types of events, e.g., events driven by user interaction, incoming network packages, or system events, can be interpreted as different non-physical noise sources, even if they apply the same sampling mechanism (time stamps of interrupts).

398 [NTG.1.5] Requirement NTG.1.5 is the equivalent to requirement PTG.3.6. Unlike for class PTG.3 only min-entropy is allowed. While for class PTG.3 the entropy claim is based on a stochastic model, the entropy claim for class NTG.1 is derived from heuristic entropy estimates.

3.6 Cross-class Topics

399 Sect. 3.3, 3.4, and 3.5 consider DRNGs, PTRNGs, and NPTRNGs. In particular, the functionality classes are specified. This section considers 'cross-class' problems where RNGs from different classes are involved.

400 [PTG.3 to DRG.3] After a total failure, a PTG.3-compliant PTRNG shall not output further internal random numbers. To be precise, its internal state allows (to some degree) a delayed reaction; cf. Requirement PTG.3.9. Unless the total failure has occurred immediately after the start of the PTRNG (i.e., before the internal state has been seeded with sufficient entropy), the internal state of the cryptographic post-processing algorithm may be assumed to have maximal entropy. In principle, the RNG could continue outputting internal random numbers, but then the RNG would no longer be compliant to functionality class PTG.3. Instead, the RNG would drop down to functionality class DRG.3 (provided that Requirement DRG.3.8 is fulfilled).

401 [PTG.3 to DRG.3] In certain scenarios (e.g., in the case of safety requirements) shutting down an RNG is not a valid option. From the perspective of a system design, it may be preferable to let a PTRNG compliant to class PTG.3 continue to operate even if a failed online test indicates

that the noise source has become inadequate. The fact that in such systems noise alarms are basically ignored does not, however, waive the respective requirements for class PTG.3. In order to be compliant to class PTG.3, the PTRNG MUST be able to detect failures AND signal them to the consuming application immediately. From the moment when the PTRNG asserts that the noise source does not deliver the required entropy to satisfy the requirements of PTG.3, the PTRNG is no longer conformant to class PTG.3 but may continue to operate. Whether the device's response is suitable or not for the application is outside the scope of this document

[combining several RNGs] It is possible to combine several individually evaluated RNGs if they are *logically and physically independent*. Logical independence means that there are no correlations by design; an extreme example of logical dependence is given, for example, by two instances of the same DRNG that were initialized identically. Usually, logical independence applies to several DRNGs. Physical independence means that the physical noise sources (including the digitization mechanism) of different RNGs do not influence each other. Within the evaluation process the developer has to give evidence that these assumptions are valid. In many cases this task may be rather easy, in other cases very difficult. 402

[combining several RNGs] This paragraph provides several examples of how to combine RNGs. It is assumed that the RNGs are physically and logically independent. For simplicity, we further assume that the internal random numbers of the RNGs have been concatenated to binary strings $y_{1(i)}, y_{2(i)}, \ldots$. The index (i) refers to the RNG $(i = 1, 2, \ldots)$. The output sequence of the combined RNG is denotes by $z_1, z_2, \ldots$. 403

(a) RNG no. 1: PTG.3-compliant, RNG no. 2: DRNG:
$z_j = y_{j(1)} + y_{j(2)} (\bmod 2)$ for $j = 1, 2, \ldots$ (corresponding bits are XORed).
The combined RNG belongs to functionality class PTG.3.

(b) RNG no. 1: PTG.2-compliant, RNG no. 2: DRG.3-compliant:
$z_j = y_{j(1)} + y_{j(2)} (\bmod 2)$ for $j = 1, 2, \ldots$ (corresponding bits are XORed).
The combined RNG belongs to functionality classes PTG.2 and DRG.3.

(c) RNG no. 1: PTG.3-compliant, RNG no. 2: PTG.2-compliant:
$z_j = y_{j(1)} + y_{j(2)} (\bmod 2)$ for $j = 1, 2, \ldots$ (corresponding bits are XORed).
The combined RNG belongs to functionality class PTG.3.

(d) RNG no. 1: DRG.3-compliant, RNG no. 2: DRG.2-compliant:
$z_j = y_{j(1)} + y_{j(2)} (\bmod 2)$ for $j = 1, 2, \ldots$ (corresponding bits are XORed).
The combined RNG belongs to functionality class DRG.3.

Note: Composition (b) is not compliant to functionality class PTG.3 because the internal state of the of the DRNG is not regularly updated with fresh entropy.

From a logical point of view, it might seem to be reasonable to extend the definition of functionality class PTG.3 by construction (b) in par. 403. However, with regard to the resistance against implementation attacks, this construction (b) has a disadvantage compared to the design demanded by functionality class PTG.3 because the DRNG never gets fresh entropy. This feature might make an attack on the DRNG easier: An adversary might try to mount a side-channel attack on the DRG.3-compliant DRNG first in order to learn its internal state and to determine 404 (and remove) its contribution to the XOR sum. In a second step the adversary could try to perform a fault injection attack on the remaining PTG.2-compliant RNG. For a PTG.3-compliant PTRNG, implementation attacks on the physical part and on the deterministic part cannot be separated in this way.

405 [combining several RNGs] In the examples of par. 403 the bitwise XOR operation may be replaced by other group operations. For instance, one could divide the sequences $(y_{j(1)})_{j\in\mathbb{N}}$) and $(y_{j(2)})_{j\in\mathbb{N}}$) into non-overlapping k-bit blocks and apply a group operation to these blocks (e.g., the addition modulo 2^8 to 8-bit blocks).

406 [combining several RNGs] In this paragraph we assume that RNG no. 1 is PTG.3-compliant and that RNG no. 2 is DRG.3-compliant. Furthermore, the output sequence of RNG no. 1 is fed into the DRNG in compliance with requirement PTG.3.5. We consider the combined RNG to be compliant with functionality class PTG.3.
Note: (i) If RNG no. 1 was PTG.2-compliant this would directly follow from the specification of functionality class PTG.3.
(ii) If the developer (applicant for a certificate) aims for an RNG-specific entropy claim for the overall PTRNG (cf. requirement PTG.3.6), this requires a specific entropy claim for RNG no. 1.

4 Mathematical Background

Chapter 4 introduces and explains central mathematical concepts that are relevant and can be useful for the evaluation of RNGs according to AIS 20 and AIS 31. In Sects. 4.1 and 4.2 definitions and facts from probability theory and stochastics are collected. In particular, random variables and stochastic processes are treated. Sect. 4.3 considers the concepts of entropy and work factor, while Sect. 4.4 deals with random mappings. In Sect. 4.5 the 'core' of any PTRNG evaluation, the concept of a stochastic model, is introduced, explained, and motivated. Furthermore, online tests and total failure tests are also addressed. Finally, Sect. 55 specifies statistical black box test suites that are applied in the evaluation of PTRNGs. The concepts and their central ideas are illustrated by examples, within the sections but also later in Chapter 5. 407

4.1 Randomness and Random Experiments

True randomness is a crucial requirement for any RNG. For non-deterministic (true) random number generators (TRNGs), loosely speaking, the noise source 'generates' randomness. For deterministic random number generators (DRNGs), the randomness is extracted from the seed material. In this section we treat randomness in a qualitative manner. 408

Probability theory describes, analyzes, and quantifies randomness by means of abstract mathematical objects, in particular by random variables and stochastic processes (cf. Sect.4.2). The core of any PTRNG evaluation is the stochastic model (Section 4.5). 409

Statistics links abstract mathematical models with real-world RNGs by experiments. These experiments may be used to estimate parameters that describe the model or to test hypotheses deduced from this model. 410

A statistical test checks whether the output sequence of an experiment is 'typical' in a specified sense. Any finite collection of statistical tests can only check finitely many criteria for 'regularity'. Hence, it is important to understand the nature of the noise source to rate the randomness of random number generation. 411

An experiment is called *unpredictable* if the observable outcome of the experiment is (to a certain extent) unknown before it is conducted. 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.
Note: Deterministic behavior can be viewed as a special case of randomness that is described by a one-point distribution. 412

After the experiment has been performed, the degree of uncertainty depends on the observer's ability to observe the outcome. Entropy (cf. Section 4.3) quantifies the degree of unpredictability relative to an observer. 413

Experiments are called *independent* if the outcomes of previous experiments do not influence the outcome of the current experiment. 414

415

Real-world RNGs cannot generate ideal randomness; they can at most approximately achieve this goal. Roughly speaking, the key point of any TRNG evaluation is to verify that the TRNG is 'sufficiently close' to ideal randomness.

4.2 Probability, stochastics, random variables

416 In Subsection 4.2.1 we introduce definitions and many concepts from probability theory and stochastics that allow making the qualitative statements from the previous section precise in a mathematical sense. Furthermore, Subsection 4.2.2 collects useful facts that are needed in this document or may be used in evaluations of real-world RNGs. In the context of this document, an important field of application are stochastic models of physical noise sources (cf. Section 4.5).

4.2.1 Definitions and basic concepts

417 In the following Ω denotes a non-empty set.

418 In this document Ω usually represents the admissible values of random numbers, random experiments, or measurements. Usually, Ω is finite (typically, $\Omega = \{0,1\}^k$ or $\Omega = \mathbb{Z}_n$) or it equals $\mathbb{R}^m$ or a subset of $\mathbb{R}^m$ ($m \geq 1$).
Note 1: Experiments with finite Ω are, for example, coin tosses and dice rolls. In the context of RNGs, random numbers are important examples that assume values in a finite or in a countable set Ω, e.g., $\Omega = \{0,1\}$ and $\Omega = \mathbb{N}_0$.
Note 2: Examples for $\Omega \subseteq \mathbb{R}^m$ are timing measurements and voltage measurements.

419 $\mathcal{P}(\Omega)$ denotes the power set of Ω. The power set contains all subsets of Ω. If Ω is finite then $|\mathcal{P}(\Omega)| = 2^{|\Omega|}$.

420 Paragraphs 421 to 430 contain basic definitions and facts from probability and measure theory, which will be needed below for proper definitions of independence or stationary stochastic processes, for example. However, these concepts are rather 'technical'. Paragraphs 432 to 435 provide a 'light version' thereof, which should suffice to understand the subsequent definitions and concepts.

421 A σ-algebra $\mathcal{A}$ over Ω is a set of subsets of Ω, i.e., $\mathcal{A} \subseteq \mathcal{P}(\Omega)$, that fulfills the following conditions:

(a) $\Omega \in \mathcal{A}$

(b) If $A \in \mathcal{A}$, then also its complement $A^c := \Omega \setminus A \in \mathcal{A}$

(c) If $A_1, A_2, \ldots \in \mathcal{A}$ then $\bigcup_{n \geq 1} A_n \in \mathcal{A}$

422 Remark: Condition 421 (c) includes finite sequences $A_1, A_2, \ldots, A_k$. Note that such a finite sequence can formally be extended by $A_{k+1} = A_{k+2} = \ldots = \{\}$ to an infinite sequence with the same union set.

Example: (i) $\mathcal{P}(\Omega)$ is a σ-algebra over Ω. 423
(ii) The Borel σ-algebra $\mathcal{B}(\mathbb{R})$ over $\mathbb{R}$ is the smallest σ-algebra that contains the open intervals (equivalently, the open subsets of $\mathbb{R}$).
(iii) More generally, for $m \geq 1$ the Borel σ-algebra $\mathcal{B}(\mathbb{R}^m)$ over $\mathbb{R}^m$ is the smallest σ-algebra that contains the open subsets of $\mathbb{R}^m$.

A probability measure ν on $\mathcal{A}$ is a mapping $\nu\colon \mathcal{A} \to [0,1]$ with the following properties 424

(a) $\nu(\Omega) = 1$

(b) If the sets $A_1, A_2, \ldots \in \mathcal{A}$ are mutually disjoint, then $\nu\left(\bigcup_n A_n\right) = \sum_{n\geq 1} \nu(A_n)$. (The sequence $A_1, A_2, \ldots$ may be finite or countable.)

More generally, if a mapping $\nu\colon \mathcal{A} \to [0,\infty]$ fulfills Condition 424 (b) and if $\nu(\Omega) < \infty$, we refer to ν as a finite measure; otherwise, ν is an infinite measure. If there exists a countable sequence $C_1 \subseteq C_2 \subseteq C_3 \ldots \in \mathcal{A}$ such that $\nu(C_n) < \infty$ for all $n \in \mathbb{N}$ and $\bigcup_{n\geq 1} C_n = \Omega$, then ν is a σ-finite measure. 425

Any $A \in \mathcal{A}$ is said to be an event or a measurable set. A pair $(\Omega, \mathcal{A})$ is denoted as a measurable space, while the triple $(\Omega, \mathcal{A}, \nu)$ is called a measure space. If ν is a probability measure the triple $(\Omega, \mathcal{A}, \nu)$ is a probability space. 426

Example: (i) Let $B(n,p)$ denote a binomial distribution with parameters n and p. Then $B(n,p)$ is a probability measure on $\mathcal{P}(\{0,\ldots,n\})$. 427
(ii) The Lebesgue measure λ is a σ-finite measure on $\mathcal{B}(\mathbb{R})$. (The Lebesgue measure corresponds to the 'geometric' measure on $\mathbb{R}$, i.e., $\lambda([a,b)) = b - a$ if $a \leq b$.).
(iii) The standard normal distribution (standard Gaussian distribution) $N(0,1)$ is a probability measure on $\mathcal{B}(\mathbb{R})$.
(iv) The Lebesgue measure λ_m on $\mathbb{R}^m$ is a σ-finite measure.

If there is no ambiguity about the σ-algebra $\mathcal{A}$, we often loosely speak of 'measures on Ω'. Unless otherwise stated in this document, $\mathcal{A} = \mathcal{P}(\Omega)$ for countable Ω (finite or infinite), and for $\mathbb{R}$, $\mathbb{R}^m$ and measurable subsets $\Omega \subseteq \mathbb{R}^m$ we use the Borel σ-algebras $\mathcal{A} = \mathcal{B}(\mathbb{R})$, $\mathcal{A} = \mathcal{B}(\mathbb{R}^m)$ or $\mathcal{A} = \mathcal{B}(\Omega)$, respectively. 428

Assume that $(\Omega_1, \mathcal{A}_1, \nu)$ is a probability space and $(\Omega_2, \mathcal{A}_2)$ a measurable space. Furthermore, let $\phi\colon \Omega_1 \to \Omega_2$ be a mapping. We call ϕ measurable (or more precisely, $(\mathcal{A}_1, \mathcal{A}_2)$-measurable) if for each $A' \in \mathcal{A}_2$ the pre-image $\phi^{-1}(A') \in \mathcal{A}_1$. If ν is a measure on $\mathcal{A}_1$ then $\nu^{\phi}(A') := \nu(\phi^{-1}(A'))$ for all $A' \in \mathcal{A}_2$ defines a measure on $\mathcal{A}_2$. We denote ν^{ϕ} the image measure (or: transformed measure) of ν under ϕ. 429

Assume that $\mathcal{A}_1$ and $\mathcal{A}_2$ are σ-algebras over Ω_1 and Ω_2. A random variable X is a measurable mapping $X\colon \Omega_1 \to \Omega_2$. In our context, Ω_2 is finite, countable, or a subset of $\mathbb{R}^m$. 430

Outside of mathematical proofs the probability space of a random variable is usually not explicitly stated. We point out that a random variable $X\colon \Omega_1 \to \Omega_2$ with probability space $(\Omega_1, \mathcal{A}_1, \nu)$ may 431 also be interpreted as a random variable on the measure space $(\Omega_2, \mathcal{A}_2, \nu^X)$. Here, ν^X denotes the image measure (or: transformed measure) of X, i.e., $\nu^X(A_2) = \nu(X^{-1}(A_2))$ for all $A_2 \in \mathcal{A}_2$. Furthermore, $\text{Prob}(X \in A_1) = \nu(A_1)$ quantifies the probability that the random variable X assumes a value in A_1.

432 ['light version' of pars. 421 to 430] As already mentioned above these definitions and concepts are needed for mathematically precise definitions in the following. Fortunately, in the context of RNG evaluations, problems concerning measurability hardly occur. The paragraphs 433 to 435 thus provide a 'light version'. This light version should suffice for at least an intuitive understanding of the following definitions and concepts and to apply them correctly. This, in particular, refers to the material collected in Subsection 4.2.2.

433 ['light version' of pars. 421 to 430 ctd.] Some of the following definitions and conditions refer to 'measurable subsets' of some space Ω (equivalently, to elements of a σ-algebra on Ω). If Ω is finite or countable, all subsets of Ω are measurable. If $\Omega \subseteq \mathbb{R}^m$ one may think of 'regular' subsets as (depending on the dimension m) intervals, rectangles, circles, cuboids, balls, etc. and countable unions thereof. (There exist further measurable and non-measurable subsets, but this is of little importance for RNG evaluations.)

434 ['light version' of pars. 421 to 430 ctd.] In this document and, more generally, in the context of the evaluation of RNGs, random variables usually assume values in finite or countable sets or in subsets of $\mathbb{R}$ or $\mathbb{R}^m$. We may speak of random variables on finite or countable set Ω (e.g., $\Omega = \{0, 1\}$), or random variables on $\mathbb{R}$ (also: 'real-valued random variables'), random variables on $\mathbb{R}^m$, or random variables on Ω.

435 ['light version' of pars. 421 to 430 ctd.] The expression $\text{Prob}(X \in A)$ quantifies the probability that the random variable X assumes a value in the set $A \subseteq \Omega$.

436 $X \sim \nu$ means that the random variable X has distribution ν, i.e., that $\text{Prob}(X \in A) = \nu(A)$. The term $\text{Prob}(X \in A)$ quantifies the probability that X assumes a value in the set A. Values that are assumed (or: taken on) by a random variable X are called *realizations* of X.

437 [Notation] In this document, we denote random variables by capital letters and their realizations usually by the corresponding small letters.

438 Example: Assume that the random variable X models the tossing of a fair coin. Then $\text{Prob}(X = 0) = \text{Prob}(X = 1) = 0.5$ if we identify 'head' and 'tail' with 1 and 0. These probabilities quantify the knowledge on the outcome of a future coin toss (and on a past experiment to a person who does not know its outcome). Possible realizations of X are 0 and 1.

439 In this document we model non-deterministic phenomena by random variables. Their realizations are observable as random numbers, voltage, or timing, for example.

440 Definition: The term $B(n, p)$ denotes the binomial distribution with parameters n and p, which is given by

$$\text{Prob}\,(X = k) = \binom{n}{k} p^k \,(1 - p)^{n-k} \quad \text{for } k = 0, \ldots, n. \tag{4.1}$$

Definition: The Poisson distribution with parameter $\tau > 0$ is given by 441

$$\text{Prob}\,(X = k) = \frac{\tau^k}{k!} e^{-\tau} \quad \text{for } k \in \mathbb{N}_0. \tag{4.2}$$

Note: The parameter $\tau > 0$ equals the mean number of events per time interval of length 1.

Definition: The geometric distribution $\mathcal{G}_p$ with parameter $p \in (0, 1]$ denotes a discrete distribution on $\mathbb{N}$. More precisely, 442

$$\mathcal{G}_p(k) := p(1-p)^{k-1} \quad \text{for } k \in \mathbb{N}. \tag{4.3}$$

The term $\mathcal{G}_p(k)$ equals the probability that a sequence of iid Bernoulli trials with individual success probability p is successful for the first time in the k^{th} trial.
Note: There also exists an alternative definition that only counts the number of failures, i.e., $k - 1$ in place of k.

Definition: The letters λ and λ_m denote the Lebesgue measures on $\mathbb{R}$ or $\mathbb{R}^m$, respectively. It is $\lambda([a, b)) = b - a$ if $a \leq b$. Accordingly, $\lambda_m\left(\prod_{j=1}^m [a_j, b_j)\right) = \prod_{j=1}^m (b_j - a_j)$ if $a_j \leq b_j$ for $1 \leq j \leq m$. 443
Note: The Lebesgue measure λ corresponds to the 'geometric' measure on $\mathbb{R}$.

Definition: The term $N(\mu, \sigma^2)$ denotes the normal (Gaussian) distribution with expectation μ and variance σ^2. It has the density 444

$$\phi(x) := \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}. \tag{4.4}$$

In particular, $N(0, 1)$ is called *standard normal distribution*. Its cumulative distribution function $\Phi(\cdot)$ is given by

$$\Phi(x) := \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{t^2}{2}}\, dt\,. \tag{4.5}$$

Note: To be precise, 'density' means 'Lebesgue density'. In this document densities with respect to other measures than the Lebesgue measure are not considered. For this reason, we briefly speak of 'density' in place of 'Lebesgue density' in the following.

Definition: The Gamma distribution with the shape parameter $\alpha > 0$ and rate parameter $\beta > 0$ has the density 445

$$\gamma_{\alpha,\beta}(x) := \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x} \quad \text{for } x > 0. \tag{4.6}$$

Note: Occasionally, the Gamma distribution is not characterized by a shape parameter and a rate parameter but by a shape parameter and a scale parameter. Thus, caution is advised when results from different books are applied. The scale parameter is the reciprocal value of the shape parameter.

The random variable X is called discrete if Ω is countable (finite or infinite). If Ω is finite we also call X a finite random variable. Examples are binomially distributed random variables and Poisson-distributed random variables. Section 4.4 deals with random mappings. There, the realizations of the random variables are mappings between sets. 446

447

Let X be a random variable that assumes values in a finite set Ω. We say that X is *uniformly distributed* (or equivalently: unbiased, equidistributed) if it assumes all $\omega \in \Omega$ with the same probability, namely $\text{Prob}(X = \omega) = |\Omega|^{-1}$. Otherwise, X is said to be biased.
Note: Precisely formulated, it should actually read $\text{Prob}(X = \{\omega\})$ instead of $\text{Prob}(X = \omega)$. However, the shorter expression '$\text{Prob}(X = \omega)$' is common for finite and countable Ω.

448 A random variable X has density $f\colon \Omega \to [0, \infty]$ with respect to a measure τ if $\text{Prob}(X \in A) = \int_A f(\omega)\, d\tau(\omega)$ for all measurable sets A. Equivalently, a measure ν has density $f\colon \Omega \to [0, \infty]$ with respect to a measure τ if $\nu(A) = \int_A f(\omega)\, d\tau(\omega)$ for all measurable A.
Note: Densities do not exist for each pair (ν, τ).

449 In our context, usually $\Omega \subseteq \mathbb{R}^m$ with $m \geq 1$, and $\tau = \lambda_m$. Then

$$\text{Prob}(X \in A) = \int_A f(x)\, \lambda_m(dx) = \int_A f(x)\, dx\,. \tag{4.7}$$

450 Let X denote a random variable that assumes values in $\mathbb{R}^m$, and has distribution ν. If the integral

$$E(X) := \int_\Omega x\, \nu(dx) \tag{4.8}$$

exists (i.e., if $\int_\Omega |x|\, \nu(dx) < \infty$) then $E(X)$ is called the expectation of X.
The expectation $E(X)$ does not exist for every random variable. Counterexamples are, for example, Cauchy-distributed random variables.

451 For discrete random variables X with values in $\Omega \subseteq \mathbb{R}$ (e.g., $\Omega = \{0, 1\}, \mathbb{N}, \mathbb{Z}$) formula (4.8) simplifies to

$$E(X) := \sum_{x \in \Omega} x\, \text{Prob}(X = x)\,. \tag{4.9}$$

If X assumes values in $\mathbb{R}^m$ and has Lebesgue density f then (4.8) reads

$$E(X) := \int_{\mathbb{R}^m} x f(x)\, dx \tag{4.10}$$

In the context of PTRNG evaluations, we are usually faced with these two special cases.

452 Remark: For random variables with values in $\{0, 1\}^n$, no meaningful definition for the mean is evident.

453 The *variance* of a real-valued random variable X is defined by

$$\text{Var}\,(X) := E\,(E\,(X) - X)^2\,. \tag{4.11}$$

provided that both expectations exist. This is not always the case.

454 Assume that $\text{Var}(X)$ exists. Then

$$\sigma_X := \sqrt{\text{Var}\,(X)}. \tag{4.12}$$

is the standard deviation of X.

455

[sum of normal distributions] If X_1 and X_2 denote independent normally distributed random variables with expectations μ_1, μ_2 and variances σ_1^2, σ_2^2, then $X_1 + X_2$ is normally distributed with expectation $\mu_1+\mu_2$ and variance $\sigma_1^2+\sigma_2^2$. More generally, if the random variables $X_1, \ldots, X_n$ are iid $N(\mu, \sigma^2)$-distributed then the sum $X_1 + \cdots + X_n$ is $N(n\mu, n\sigma^2)$-distributed.

[Gamma distribution] The Gamma distribution with the shape parameter $\alpha > 0$ and rate parameter $\beta > 0$ has the density $\gamma_{\alpha,\beta}(\cdot)$, cf. par. 445. A random variable that is Gamma distributed with parameters α and β has mean $\mu = \alpha/\beta$ and variance $\sigma^2 = \alpha/\beta^2$. 456

[sum of Gamma distributions] If X and Y are independent random variables with densities $\gamma_{\alpha_1,\beta}(\cdot)$ and $\gamma_{\alpha_2,\beta}(\cdot)$, respectively, then $X + Y$ is Gamma-distributed with density $\gamma_{\alpha_1+\alpha_2,\beta}(\cdot)$. Consequently, if the random variables $X_1, \ldots, X_n$ are iid Gamma distributed with parameters α and β then the sum $X_1 + \cdots + X_n$ is Gamma distributed with parameters $n\alpha$ and β. 457

The random variables $X_1, X_2, \ldots, X_k$ are said to be *independent* if for *each* k-tuple $(A_1, \ldots, A_k)$ of measurable sets the equality 458

$$\operatorname{Prob}(X_1 \in A_1, \ldots, X_k \in A_k) = \prod_{j=1}^{k} \operatorname{Prob}(X_j \in A_j) \,. \tag{4.13}$$

holds.

More generally, the (infinite) sequence $X_1, X_2, \ldots$ of random variables is said to be *independent* if for *each* integer $k' \geq 1$ and for *each* k'-tuple $(A_1, \ldots, A_{k'})$ of measurable sets, condition (4.13) is valid (with k' in place of k). 459
Note: Independence can be generalized to uncountable index sets.

For discrete random variables $X_1, X_2, \ldots$ with values in Ω, condition (4.13) simplifies to 460

$$\operatorname{Prob}(X_1 = x_1, \ldots, X_k = x_k) = \prod_{j=1}^{k} \operatorname{Prob}(X_j = x_j) \tag{4.14}$$

for each k-tuple $(x_1, \ldots, x_k) \in \Omega^k$.

In the context of random variables $X_1, X_2, \ldots$, the abbreviation iid stands for 'independent and identically distributed'. 461

Mathematically, a sequence of iid uniformly distributed random variables $X_1, X_2, \ldots$ on a finite set Ω (e.g., $\Omega = \{0, 1\}$) describes an ideal RNG. 462

Assume that the random variables $X_1, X_2, \ldots, X_n$, resp. $X_1, X_2, \ldots$ are independent. If $X_j \sim \nu_j$ the joint distribution of $(X_1, X_2, \ldots, X_n)$, resp., of the sequence $X_1, X_2, \ldots$ is given by the product measure $\otimes_{j=1}^{n} \nu_j$ resp. by $\otimes_{j=1}^{\infty} \nu_j$. These product measures are characterized by the conditions from pars. 458 and 459. If the random variables $X_1, X_2, \ldots$ are identically distributed, i.e., if $\nu_1 = \nu_2 = \cdots = \nu_n$, we alternatively also use the notation ν^n and $\nu^{\mathbb{N}}$. 463

Assume that for the real-valued random variables X and Y, expectations and variances exist. 464

Then the right-hand sides of (4.15) and (4.16) exist

$$\mathrm{Cov}\,(X,Y) := E\,(XY) - E\,(X)\,E\,(Y) \quad \text{(covariance)} \tag{4.15}$$

$$\mathrm{corr}\,(X,Y) := \frac{\mathrm{Cov}(X,Y)}{\sigma_X \cdot \sigma_Y} = \frac{E\,(XY) - E\,(X)\,E\,(Y)}{\sqrt{\mathrm{Var}(X)}\sqrt{\mathrm{Var}(Y)}} \quad \text{(correlation coefficient)} \tag{4.16}$$

If $\mathrm{Cov}(X,Y) = 0$ we say that X and Y are uncorrelated.

465 Independence implies uncorrelatedness but, in general, the converse is not true (cf. pars. 466 and 467).

466 Counterexample ([Geor15], Beispiel (4.26)): Assume that X and Y are random variables that assume values in $\Omega_1 = \{-1,0,1\}$ and in $\Omega_2 = \{0,1\}$, respectively. Assume further that $\mathrm{Prob}(X = 1, Y = 0) = \mathrm{Prob}(X = 0, Y = 1) = \mathrm{Prob}(X = -1, Y = 0) = 1/3$. Hence $\mathrm{Prob}(X = 0) = \mathrm{Prob}(X = 1) = \mathrm{Prob}(X = -1) = 1/3$ and thus $E(X) = 0$. Similarly, $\mathrm{Prob}(Y = 0) = 2/3$, $\mathrm{Prob}(Y = 1) = 1/3$ and thus $E(Y) = 1/3$. Finally,

$$\mathrm{Cov}(X,Y) = \mathrm{E}\,(XY) - 0 \cdot \frac{1}{3} = \sum_{x\in\Omega_1, y\in\Omega_2} xy\,\mathrm{Prob}(X = x, Y = y) =$$
$$\left(1 \cdot 0 \cdot \frac{1}{3} + 0 \cdot 1 \cdot \frac{1}{3} - 1 \cdot 0 \cdot \frac{1}{3}\right) = 0\,.$$

Thus, X and Y are uncorrelated but $\mathrm{Prob}(X = 1, Y = 1) = 0 \neq 1/9 = \mathrm{Prob}(X = 1) \cdot \mathrm{Prob}(Y = 1)$ shows that the random variables X and Y are not independent.

467 Assume that the random variables X and Y are bivariate normally distributed. If X and Y are uncorrelated, then X and Y are independent.

468 Let $(\Omega, \mathcal{A}, P)$ a probability space. Formally, a *stochastic process* $(X_t)_{t\in T}$ with state space Ω is a collection of real-valued random variables $\{X_t \mid t \in T\}$, where the index t is usually interpreted as 'time'.

469 If $T \subseteq \mathbb{R}$ is an interval (e.g., $T = (a,b)$, $T = [0,\infty)$ or $T = \mathbb{R}$), we speak of (time-)continuous stochastic processes. If $T \subseteq \Delta\mathbb{Z}$ for some $\Delta > 0$, e.g., $T = \mathbb{Z}, T = \mathbb{N}$ or $T = \mathbb{N}_0$, the stochastic process is called (time-)discrete.

470 Example: Markov chains (time-discrete stochastic process); cf. par. 499, Wiener process (time-continuous stochastic process)

471 A stochastic process $(X_t)_{t\in T}$ is called *stationary* (or: stationary in a strict sense) if

$$\mathrm{Prob}\,(X_{t_1} \in A_1, X_{t_2} \in A_2, \ldots, X_{t_k} \in A_k) = P\,(X_{t_1+\tau} \in A_1, X_{t_2+\tau} \in A_2, \ldots, X_{t_k+\tau} \in A_k) \tag{4.17}$$
for each $k \in \mathbb{N}$, $\tau > 0$, all $t_1 < \cdots < t_k$ with $t_j, t_j + \tau \in T$ $(j \leq k)$,
and all measurable sets $A_1, \ldots, A_k$.

If the random variables X_j are discrete, (4.17) simplifies to

$$\mathrm{Prob}\,(X_{t_1} = x_1, X_{t_2} = x_2, \ldots, X_{t_k} = x_k) = \mathrm{Prob}\,(X_{t_1+\tau} = x_1, X_{t_2+\tau} = x_2, \ldots, X_{t_k+\tau} = x_k) \tag{4.18}$$
for each $k \in \mathbb{N}$, $\tau > 0$, all $t_1 < \cdots < t_k$ with $t_j, t_j + \tau \in T$ $(j \leq k)$, and all $x_1, \ldots, x_k \in \Omega$.

Stationarity means that the distribution of the stochastic process is time-invariant. In other words: For admissible shifts τ (that is, $T + \tau \subseteq T$), the stochastic processes $(X_t)_{t \in T}$ and $(X_{t+\tau})_{t \in T}$ are identically distributed. If $T = \mathbb{R}$ or $T = [0, \infty)$, for example, any $\tau > 0$ is admissible. For $T = \mathbb{Z}$ or $T = \mathbb{N}$ (time-discrete stochastic processes), the shift parameter τ must be a (non-negative) integer. 472

A stochastic process $(X_t)_{t \in T}$ is *stationary in a weak sense* (or: stationary in a wide sense) if 473

$$E(X_t) = E(X_{t+\tau}) \tag{4.19}$$

$$E((X_{t_1} - \mu)(X_{t_2} - \mu)) = E((X_{t_1+\tau} - \mu)(X_{t_2+\tau} - \mu)) \tag{4.20}$$

for all $t, t + \tau \in T$, $\tau > 0$. In particular, then

$$K_X(t_2 - t_1) := E((X_{t_1} - \mu)(X_{t_2} - \mu)) \tag{4.21}$$

is the autocovariance of the stochastic process $(X_t)_{t \in T}$.

Stationarity implies stationarity in the weak sense. Par. 506 collects useful facts. Stationarity plays an important role in stochastic models (Sect. 4.5) for PTRNGs. It captures the desired feature that if a PTRNG is analyzed at a certain period in time, its stochastic behaviour should be the same at different times. 474
Note: For stochastic models of physical noise sources, the requirement is relaxed to time-local stationarity; cf. pars. 668 to 670

If a (time-continuous or time-discrete) stationary stochastic process is ergodic, then statistical properties of this stochastic process can be deduced from a single, sufficiently long realization of this stochastic process with probability 1. 475
Note 1: In the context of the evaluation of PTRNGs, this feature is exploited for the estimation of parameters, by online tests, and by evaluator tests, for example.
Note 2: There exist several equivalent formal definitions for ergodicity, e.g., that the invariant events are attained with probability 0 or 1. We refer the interested reader to the relevant literature, e.g., to [KaTa75], Chap. 9.
Note 3: Par. 476 and 477 provide an example and counterexamples of an ergodic process. Loosely speaking, to ensure ergodicity, it suffices if the long-term dependencies of the stochastic process decrease sufficiently fast.

Example: Assume that the random variables $X_1, X_2, \ldots$ are iid $B(1, p)$-distributed. If we observe a realization sequence $x_1, x_2, \ldots$, the empirical mean $n^{-1} \sum_{j=1}^{n} x_j$ converges to p with probability 1 (Strong law of large numbers). If the random variables model the repeated tossing of a particular coin (cf. Subsec. 4.5.2), a sequence of realizations can easily be obtained by tossing this coin several times, which allows the estimation of the (unknown) parameter p. The random variables $X_1, X_2, \ldots$ are an example of a stationary ergodic process (cf. par. 477). 476

Counterexample: Assume that the random variables $X_1, X_2, \ldots$ are identically $B(1, p)$. Unlike in par. 476 these random variables are not independent but fully dependent, namely $X_1 = X_2 = \cdots$. Then the realization of X_1 determines the whole realization sequence. In this case one can only observe the realization sequences $1, 1, \ldots$ (with probability p) or $0, 0, \ldots$ (with probability $1 - p$). Hence, it is not possible to estimate p on the basis of a single realization sequence. The stochastic process is stationary but not ergodic. 477

478 [empirical mean and empirical variance] Assume that $x_1, x_2, \ldots, x_m$ are realizations of the iid random variables $X_1, X_2, \ldots, X_m$. Assume further that the expectation $\mu = E(X_j)$ and the variance $\sigma^2 = \mathrm{Var}(X_j)$ exist. The *arithmetic mean* $\overline{x}$ and the *empirical variance* $\overline{s}^2$ of $x_1, x_2, \ldots, x_m$ are given by

$$\overline{x} := \frac{x_1 + x_2 + \cdots + x_m}{m} \tag{4.22}$$

$$\overline{s}^2 := \frac{1}{m-1} \sum_{j=1}^{m} (x_j - \overline{x})^2 \tag{4.23}$$

$\overline{x}$ and $\overline{s}^2$ are *unbiased* estimators of μ and σ^2. Both estimators are *unbiased.* In this context, unbiased means, that if the sample values x_j in the right-hand sides of (4.22) and (4.23) are replaced by random variables X_j, the expectation of these terms is μ and σ^2, respectively.
Note: Occasionally, formula (4.23) is used with factor $1/m$ in place of $1/(m-1)$. In this case the estimator is biased (but asymptotically unbiased).

479 [empirical mean and empirical variance] Assume that the random variables $X_1, X_2, \ldots, X_m$ are iid $N(\mu, \sigma^2)$-distributed. Then

$$\frac{X_1 + X_2 + \cdots + X_m}{m} \sim N\left(\mu, \frac{\sigma^2}{m}\right) \quad \text{and} \tag{4.24}$$

$$\frac{m-1}{\sigma^2} \cdot \frac{1}{m-1} \sum_{j=1}^{m} \left(X_j - \overline{X}\right)^2 \sim \chi^2_{m-1} \tag{4.25}$$

where χ_{n-1} denotes the χ^2-distribution with $n-1$ degrees of freedom. Formula (4.25) is a well-known corollary from Cochrane's Theorem.

480 [empirical mean and empirical variance] If the random variables $X_1, X_2, \ldots, X_m$ are iid (but not necessarily normally distributed) then

$$E\left(\frac{1}{m-1} \sum_{j=1}^{m} \left(X_j - \overline{X}\right)^2\right) = \sigma^2 \tag{4.26}$$

$$\mathrm{Var}\left(\frac{1}{m-1} \sum_{j=1}^{m} \left(X_j - \overline{X}\right)^2\right) = \frac{1}{m}\left(E\left((X-\mu)^4\right) - \frac{m-3}{m-1}\sigma^4\right) \tag{4.27}$$

481 [Allan variance] When estimating the jitter of digital clock signals, for example, the empirical variance may overestimate the jitter if low frequency noise as flicker noise is present. In such scenarios often the (empirical) Allan variance is used instead; cf., e.g., [ASPB+18]. Assume that the measurement values $x_1, x_2, \ldots, x_m$ are taken at times $\tau, 2\tau, \ldots, m\tau$. In practice, the x_j often are fractional frequencies that have averaged over an interval of length τ. The (empirical) Allan variance of $x_1, x_2, \ldots, x_m$ is defined by

$$\overline{\mathrm{AVar}} = \frac{1}{2(m-1)} \sum_{j=1}^{m-1} (x_j - x_{j+1})^2 \tag{4.28}$$

Note 1: By construction, the Allan variance is only a little sensitive to slow drifts of the distributions of the corresponding random variables $X_1, X_2, \ldots, X_m$.
Note 2: The definition of the Allan variance is not unique in the literature.

[Allan variance] Assume that the measurement values $x_1, x_2, \ldots$ are realizations of the random variables $X_1, X_2, \ldots$. If the random variables are stationarily distributed and uncorrelated (i.e., $\mathrm{Cov}(X_i, X_j) = 0$ for $i \neq j$), the Allan variance coincides with the 'usual' variance [ASPB+18], Theorem 1. 482
Note 1: Under these conditions the expectation of $\overline{\mathrm{AVar}}$ in (4.28) equals $0.5\left(E\left((X_j - X_{j+1})^2\right)\right)$.
Note 2: Independence implies uncorrelatedness.

4.2.2 Useful theorems and facts

This subsection provides facts and theorems that can be useful in the context of this document. 483

[Stirling's approximation] 484

$$\sqrt{2\pi n}\left(\frac{n}{e}\right)^n e^{\frac{1}{12n+1}} < n! < \sqrt{2\pi n}\left(\frac{n}{e}\right)^n e^{\frac{1}{12n}} \quad \text{(Stirling's approximation)} \tag{4.29}$$

[Stirling's approximation] If n, k, and $n-k$ are large, then applying the lower bound in Stirling's formula (4.29) to the factorials of $\binom{n}{k}$ yields the approximation 485

$$\binom{n}{k} \approx \sqrt{\frac{n}{2\pi k(n-k)}} \cdot \frac{n^n}{k^k (n-k)^{n-k}}\,. \tag{4.30}$$

[Expectation: computation rules] Assume that for the (not necessarily independent nor identically distributed) random variables $X_1, \ldots, X_k$, the expectations $E(X_j)$ exist. Let $Y = a_1X_1 + \cdots + a_kX_k$ with $a_1, \ldots, a_k \in \mathbb{R}$. Then the expectation of Y exists. More precisely, 486

$$E(Y) = E\left(a_1X_1 + \cdots + a_kX_k\right) = \sum_{j=1}^{k} a_j E(X_j)\,. \tag{4.31}$$

If the random variables are iid and $a_j = 1/k$ for each $j \leq k$ then $E(Y) = E(X_1) = \cdots = E(X_k)$.

[Variance: computation rules] Assume that for the independent (but not necessarily identically distributed) random variables $X_1, \ldots, X_k$, the variances $\mathrm{Var}(X_j)$ exist. Let $Y = a_1X_1 + \cdots + a_kX_k$ for $a_1, \ldots, a_k \in \mathbb{R}$. Then the expectation of Y exists. More precisely, 487

$$\mathrm{Var}(Y) = \mathrm{Var}\left(a_1X_1 + \cdots + a_kX_k\right) = \sum_{j=1}^{k} a_j^2 \mathrm{Var}(X_j)\,. \tag{4.32}$$

If we drop the assumption that the random variables $X_1, \ldots, X_k$ are independent, then (4.32) becomes more complicated

$$\mathrm{Var}(Y) = \mathrm{Var}\left(a_1X_1 + \cdots + a_kX_k\right) = \sum_{j=1}^{k} a_j^2 \mathrm{Var}(X_j) + \sum_{i \neq j} a_i a_j \mathrm{Cov}(X_i, X_j)\,. \tag{4.33}$$

488 Example: Expectation and variance of $B(n,p)$-distributed random variables.
The random variable $Y = Y_1 + \cdots + Y_n \sim B(n,p)$ if $Y_1, \ldots, Y_n$ are iid $B(1,p)$-distributed. By (4.31) and (4.32) we conclude that $\mathrm{E}(Y) = \mathrm{E}(Y_1) + \cdots + \mathrm{E}(Y_n) = np$ and $\mathrm{Var}(Y) = \mathrm{Var}(Y_1) + \cdots + \mathrm{Var}(Y_n) = np(1-p)$.

489 [Central Limit Theorem (CLT)] Assume that the real-valued random variables $X_1, X_2, \ldots$ are iid with expectation μ and variance σ^2. For $n = 1, 2, \ldots$

$$S_n^* := \frac{X_1 + \cdots + X_n - n\mu}{\sqrt{n}\sigma} \tag{4.34}$$

define normalized partial sums. The Central Limit Theorem (CLT) applies to the sequence $X_1, X_2, \ldots$. More precisely,

$$\lim_{n\to\infty} \mathrm{Prob}(S_n^* \le x) = \Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{t^2}{2}}\, dt \quad \text{for all } x \in \mathbb{R}. \tag{4.35}$$

490 [tail of the standard normal distribution] For $x > 0$ it is

$$\left(\frac{1}{x} - \frac{1}{x^3}\right) \frac{1}{\sqrt{2\pi}} e^{-x^2/2} \le 1 - \Phi(x) = \Phi(-x) \le \frac{1}{x} \frac{1}{\sqrt{2\pi}} e^{-x^2/2} \tag{4.36}$$

([GaSt77], Lemma 1.19.2).

491 [CLT, parameter estimation] Assume that $X_1, X_2, \ldots$ are iid $B(1,p)$-distributed. Then the CLT implies

$$\mathrm{Prob}\left(\left|\frac{1}{N}\sum_{j=1}^{N} X_j - p\right| > \epsilon\right) = \mathrm{Prob}\left(\left|\frac{\sum_{j=1}^{N} X_j - Np}{N}\right| > \epsilon\right) =$$
$$\mathrm{Prob}\left(\left|\frac{\sum_{j=1}^{N} X_j - Np}{\sqrt{N}\sqrt{p(1-p)}}\right| > \frac{\epsilon\sqrt{N}}{\sqrt{p(1-p)}}\right) = 2\Phi\left(\frac{-\epsilon\sqrt{N}}{\sqrt{p(1-p)}}\right) \le 2\Phi\left(-2\epsilon\sqrt{N}\right). \tag{4.37}$$

492 [CLT] Par. 489 formulates the Central Limit Theorem (CLT) for iid random variables. The CLT is very robust and holds under weak conditions. Under suitable conditions the iid assumption and even the independence property may be dropped. Some special cases are covered in paragraphs 493, 504, 505.
Background information: If the CLT applies the random variables $S_1^*, S_2^*, \ldots$ *converge to* $N(0,1)$ *in distribution*. We do not go deeper but refer the interested reader to ([Geor15], Subsect. 5.3).

493 [CLT] Assume that the real-valued random variables $X_1, X_2, \ldots$ are independent (but not necessarily iid with with expectations $\mathrm{E}(X_j) = \mu_j$ and variances $\mathrm{Var}(X_j) = \sigma_j^2$ for $j \in \mathbb{N}$. For $n = 1, 2, \ldots$

$$S_n^* := \frac{\sum_{j=1}^{n} (X_j - \mu_j)}{\sqrt{s_n^2}} \quad \text{with } s_n^2 := \sum_{j=1}^{n} \sigma_2^2 \tag{4.38}$$

defines normalized partial sums. Assume further that the Lindeberg condition holds

$$\lim_{n\to\infty} L_n(\delta) = 0 \quad \text{for all } \delta > 0 \qquad \text{where } L_n(\delta) := \frac{1}{s_n^2}\sum_{j=1}^{n} \mathrm{E}\left((X_j-\mu_j)^2 1_{\{|X_j-\mu_j|\geq \delta s_n\}}\right) \tag{4.39}$$

Then the Central Limit Theorem (CLT) applies to the sequence $X_1, X_2, \ldots$. In particular

$$\lim_{n\to\infty} \mathrm{Prob}(S_n^* \leq x) = \Phi(x) \quad \text{for all } x \in \mathbb{R}. \tag{4.40}$$

[CLT] Assume that the random variables $X_1, X_2, \ldots$ are iid and that besides $E(X_1)$ and $E(X_1^2)$, also the third moment $E(X_1^3)$ exist. Then the well-known Berry-Esséen-Theorem provides an upper bound for the maximal difference between the exact cumulative distribution function of S_n^* and $\Phi(\cdot)$. It is 494

$$\text{(Berry-Esséen-Theorem)} \quad |\mathrm{Prob}(S_n^* \leq x) - \Phi(x)| \leq C\frac{\mathrm{E}\left(|X_1 - E(X_1)|^3\right)}{(\mathrm{Var}(X_1))^{1.5}}\frac{1}{\sqrt{n}} \quad \text{for each } x \in \mathbb{R} \tag{4.41}$$

for a suitable constant C (cf. [Geor15], Bemerkung (5.31), with $C = 0.8$). In [Shev11] it is proved that $C < 0.4748$. In particular, (4.41) says that the rate of convergence is $O\left(n^{-0.5}\right)$.

A sequence $X_1, X_2, \ldots$ of random variables is called q-dependent if the random vectors $(X_1, \ldots, X_u)$ and $(X_v, \ldots, X_n)$ are independent for all $1 \leq u < v \leq n$ with $v - u > q$. 495
Note: The components of each vector need not be independent.

[CLT for q-dependent random variables, [HoRo48]] Let $X_1, X_2, \ldots$ be q-dependent (not necessarily stationary) sequence of random variables such that $E(|X_i|^3)$ is uniformly bounded for all $i \in \mathbb{N}$. 496

$$A_i := \mathrm{Var}(X_{i+q}) + 2\sum_{j=1}^{q} \mathrm{Cov}(X_{i+q-j}, X_{i+q}) \quad \text{for } i \in \mathbb{N} \tag{4.42}$$

If the limit $A := \lim_{u\to\infty} u^{-1}\sum_{h=1}^{u} A_{i+h}$ exists uniformly for all $i \in \mathbb{N}$ then

$$\lim_{n\to\infty} \mathrm{Prob}\left(\frac{\sum_{j=1}^{n}(X_j - E(X_j))}{\sqrt{An}} \leq x\right) = \Phi(x) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{x} e^{\frac{t^2}{2}}\, dt \quad \text{for all } x \in \mathbb{R}. \tag{4.43}$$

[CLT for q-dependent random variables] If the random variables $X_1, X_2, \ldots$ in par. 496 are stationary, the necessary conditions simplify considerably: It suffices that $E(|X_i|^3)$ exists, and 497

$$A = \sigma^2 := \mathrm{Var}(X_1) + 2\sum_{j=1}^{q} \mathrm{Cov}(X_1, X_{1+j})\,. \tag{4.44}$$

In particular, for $\mu := E(X_1)$ we obtain the equivalent to (4.43)

$$\lim_{n\to\infty} \mathrm{Prob}\left(\frac{\sum_{j=1}^{n}(X_j - \mu)}{\sigma\sqrt{n}} \leq x\right) = \Phi(x) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{x} e^{-0.5t^2}\, dt \quad \text{for all } x \in \mathbb{R}. \tag{4.45}$$

498

[CLT, dependent random variables] The CLT may even hold if $X_1, X_2, \ldots$ has no finite memory, provided that the dependencies decrease sufficiently fast. If the sequence $X_1, X_2, \ldots$ is stationary and, e.g., strongly mixing then the CLT holds if some further conditions are fulfilled. If needed, the reader is referred to [Jone04], Sect. 4, for details.

499 [Markov chains] Assume that the random variables $X_0, X_1, \ldots$ take on values in a countable set Ω. Assume that

$$\mathrm{Prob}\left(X_{n+1} = x_{n+1} \mid X_1 = x_1, \ldots, X_n = x_n\right) = \mathrm{Prob}\left(X_{n+1} = x_{n+1} \mid X_n = x_n\right) \tag{4.46}$$

for each $n \in \mathbb{N}_0$ and all $x_0, x_1, \ldots, x_{n+1} \in \Omega$, provided that both conditional probabilities in (4.46) are well-defined. (The latter is the case when $\mathrm{Prob}(X_1 = x_1, \ldots, X_n = x_n) > 0$.) We say that $X_0, X_1, \ldots$ is a (time-discrete) Markov chain on the state space Ω. If the right-hand conditional probabilities in (4.46) do not depend on n the Markov chain is homogeneous.

500 [Markov chains] Condition (4.46) says that X_{n+1} may depend on X_n but any further information on the preceding random variables $X_0, \ldots, X_{n-1}$ does not provide additional information on the outcome of X_{n+1}.
Note 1: Equation (4.46) does not imply that X_{n+1} and X_{n-1} are independent.
Note 2: Condition (4.46) can be generalized to time-continuous stochastic processes on arbitrary state spaces (Markov processes).

501 [Markov chains] Assume that $X_0, X_1, \ldots$ is a homogeneous Markov chain on the finite state space $\Omega = \{\omega_1, \ldots, \omega_k\}$. The transition matrix $P = (p_{ij})_{1 \le i,j \le k}$ is defined by $p_{ij} = \mathrm{Prob}(X_{n+1} = \omega_j \mid X_n = \omega_i)$. If the row vector ν_j denotes the distribution of X_j, then $\nu_{n+1} = \nu_n P$.
Note: In the literature on Markov chains, traditionally row vectors are used instead of column vectors.

502 [Markov chains] Assume that $X_0, X_1, \ldots$ is a homogeneous Markov chain on the finite state space $\Omega = \{\omega_1, \ldots, \omega_k\}$ with transition matrix P. Assume further that there is an integer $m \in \mathbb{N}$ for which all entries of P^m are positive.
Then for each initial distribution ν_0 the sequence of distributions $\nu_0, \nu_1, \ldots$ converges to a limit distribution ν with $\nu(\omega_j) > 0$ for all $j \le k$. The limit distribution ν is the unique left eigenvector of P to the eigenvalue 1. The convergence rate is exponentially (e.g., [Geor15], Subsect. 6.3.1).

503 If the Markov chain from par. 502 already starts in the equilibrium state, namely if $\nu_0 = \nu$, then $\nu_0 = \nu_1 = \cdots = \nu$. Then the Markov chain $X_0, X_1, \ldots$ is stationary and ergodic (see, e.g., [Geor15], Subsect. 6.3.1).

504 [CLT, Markov chain] Assume that $X_0, X_1, \ldots$ is a homogeneous Markov chain on the finite state space $\Omega = \{\omega_1, \ldots, \omega_k\}$ with transition matrix P. Assume further that the Markov chain converges to a limit distribution ν regardless of ν_0 (as in par. 502). Let $g\colon \Omega \to \mathbb{R}$ any mapping. Then the Central Limit Theorem applies to $g(X_1), g(X_2), \ldots$. The normalized partial sums (see par. 489) are given by

$$S_n^* := \frac{g(X_1) + \cdots + g(X_n) - n\mu}{\sqrt{n}\sigma} \quad \text{with} \tag{4.47}$$

$$\mu := \mathrm{E}\left(g(X_1)\right) \quad \text{and } \sigma^2 := \mathrm{Var}\left(g(X_1)\right) + 2\sum_{k=1}^{\infty} \mathrm{Cov}\left(g(X_1), g(X_{1+k})\right).$$

The version of the CLT from par. 504 also applies under more general conditions. 505
Note: If the random variables $X_1, X_2, \ldots$ are independent (special case of a Markov chain), (4.47) corresponds to (4.34), applied to the random variables $g(X_1), g(X_2), \ldots$.

[Stationarity] This paragraph contains some useful facts on stationary sequences. Assume that 506
the stochastic process $(X_n)_{n\in\mathbb{N}}$ is stationary on a state space $\Omega = \mathbb{R}^m$ for some integer $m \geq 1$. Furthermore, let $f\colon \Omega \to \Omega' := \mathbb{R}^s$ denote a measurable mapping. (As usual, we consider the Borel-σ-algebras on $\mathbb{R}^m$ and $\mathbb{R}^s$.)
a) The stochastic process $(f(X_n))_{n\in\mathbb{N}}$ is stationary, too.
b) The stochastic process $Y_1, Y_2, \ldots$ with $Y_n := (X_{(n-1)t+1}, \ldots, X_{nt})$ is a stationary process on $\mathbb{R}^{st}$. The vectors $Y_1, Y_2, \ldots$ are non-overlapping.
Proof: a) Since ϕ is measurable $X \in f^{-1}(B') \in \mathcal{B}(\mathbb{R}^s)$ for all $B' \in \mathcal{B}(R^s)$, and the stationarity of $(X_n)_{n\in\mathbb{N}}$ implies that $(f(X_n))_{n\in\mathbb{N}}$ is stationary, too.
b) The stationarity of $(X_n)_{n\in\mathbb{N}}$ implies

$$\begin{aligned}
&\mathrm{Prob}(Y_j \in (B_{(j-1)t+1} \times \cdots \times B_{jt}) \text{ for } j \leq k) = \mathrm{Prob}(X_i \in B_i \text{ for } i \leq kt) = \\
&\mathrm{Prob}(X_{i+t\tau} \in B_i \text{ for } i \leq kt) = \mathrm{Prob}(Y_{j+\tau} \in (B_{(j-1)t+1} \times \cdots \times B_{jt}) \text{ for } j \leq k) \\
&\text{for each } \tau \in \mathbb{N} \text{ and all } B_j \in \mathcal{B}(\mathbb{R})
\end{aligned} \tag{4.48}$$

Since the set $\{B'_1 \times B'_2 \times \cdots \times B'_{kt} \mid B'_j \in \mathcal{B}(\mathbb{R})\}$ is stable under intersections and generates $\mathcal{B}(\mathbb{R}^{kt})$, the vectors $(Y_1, \ldots, Y_k)$ and $(Y_{1+\tau}, \ldots, Y_{k+\tau})$ are identically distributed on $\mathbb{R}^{kt}$. Therefore, (4.48) generalizes to

$$\mathrm{Prob}(Y_j \in A_j \text{ for } j \leq k) = \mathrm{Prob}(Y_{j+\tau} \in A_j \text{ for } j \leq k) \text{ for each } \tau \in \mathbb{N} \text{ and } A_j \in \mathcal{B}(\mathbb{R}^t)\,. \tag{4.49}$$

[Stationarity] The feature that stationarity is 'inherited' is very useful for the analysis of PTRNGs. 507
The assertions from par. 506 are also valid for time-continuous stochastic processes $(X_t)_{t\in T}$.

[Renewal process] Assume that $T_1, T_2, \ldots$ denote iid non-negative random variables, and that 508
the expectation $E(T_j) > 0$ exists. Furthermore, unless otherwise stated, $\mathrm{Prob}(T_j = 0) = 0$, and there is no $\Delta > 0$ such that $\mathrm{Prob}(T_j \in \{j\Delta \mid j \in \mathbb{N}\}) = 1$. Then $Z(t) := \inf\{k \mid T_1 + \cdots + T_k > t\}$ defines a *renewal process*, where t ranges in $[0, \infty)$. The random variables T_j are often interpreted as lifetimes of machines and called the j^{th} holding time. In the context of physical noise sources, the random variable T_j often quantifies the intermediate time between the $(j-1)^{th}$ and the j^{th} event; see Subsects. 5.4.2, 5.4.3, and 5.4.4.

[Renewal process] A *delayed renewal process* considers independent, non-negative random variables $T_0, T_1, T_2, \ldots$. Again, the random variables $T_1, T_2, \ldots$ are iid, while T_0 can have a different distribution. Furthermore, it is assumed that $E(T_0) > 0$ and $E(T_j) > 0$ exist. Also, let 509

$$J_n := T_0 + T_1 + \cdots + T_n\,. \tag{4.50}$$

The delayed renewal process is given by

$$Z(t) := \inf\{k \mid T_0 + T_1 + \cdots + T_k > t\}\,. \tag{4.51}$$

Note: The renewal process from par. 508 can be interpreted as a special case of a delayed renewal process with $T_0 \equiv 0$. It is also called a *non-delayed renewal process*.

510

[Stationary renewal process] A delayed renewal process is a *stationary renewal process* (or: *equilibrium renewal process*) if the increments $(Z(t_2)-Z(t_1), Z(t_3)-Z(t_2), \ldots Z(t_{m+1})-Z(t_m))$ have the same distribution as $(Z(t_2+t)-Z(t_1+t), Z(t_3+t)-Z(t_2+t), \ldots Z(t_{m+1}+t)-Z(t_m+t))$ for all $m \in \mathbb{N}$, $0 \leq t_1 < \ldots < t_{m+1}$, and $t > 0$. Then the distribution of $J_{Z(t)} - t$ converges to a limiting distribution as t tends to ∞. More precisely, if G_T denotes the cumulative distribution function of the random variables T_j then

$$G_{T,\infty}(x) := \lim_{t\to\infty} \text{Prob}(J_{Z(t)} - t \leq x) = \frac{1}{\mu}\int_0^x (1 - G_T(u))\, du\,. \tag{4.52}$$

If the T_j have density $g(\cdot)$ then

$$G_{T,\infty}(\cdot) \quad \text{has density } g_\infty(x) := (1 - G_T(x))/\mu. \tag{4.53}$$

The formulae (4.52) and (4.53) are well-known, cf. [Fell65], Chap. XI, (4.10).
If the distribution of T_0 equals the limiting distribution, i.e., if $\text{Prob}(T_0 \leq x) = G_{T,\infty}(x)$, then the renewal process is stationary.
Note 1: This property is fulfilled if the random variables $\ldots, J_{-1}, J_0$ are in equilibrium.
Note 2: In the context of physical noise sources, stationary renewal processes are of particular interest; cf. Subsects. 5.4.2, 5.4.4, and, in particular, 5.4.3.

511 [Stationary renewal process] Assume that $\{Z(t) \mid t \geq 0\}$ defines a stationary renewal process for which $\sigma^2 = \text{Var}(T_j)$ exist. Then (e.g., [Cox62], Sect. 4.5, Formula (18)),

$$E\,(Z(t)) \;=\; \frac{t}{\mu}\,, \tag{4.54}$$

$$\text{Var}\,(Z(t)) \;=\; \left(\frac{\sigma^2}{\mu^3}\right)t + \frac{1}{6} + \frac{\sigma^4}{2\mu^4} - \frac{E\left((T-\mu)^3\right)}{3\mu^3} + o(1)\,. \tag{4.55}$$

Note: Of course, for large t the expectation and the variance of the non-delayed renewal process are rather similar to (4.54) and (4.55). In particular, for the non-delayed renewal process (4.54) applies only asymptotically.

512 [Stationary renewal process] If $\{Z(t) \mid t \in [0,\infty)\}$ defines a stationary renewal process, then

$$\left(\left(T_{Z(t)}, T_{Z(t)+1}, \ldots\right), T_{Z(t)} - t\right) \quad \text{is stationary in } t. \tag{4.56}$$

For (4.56) the requirement that the random variables $T_1, T_2, \ldots$ are iid can be relaxed to the assumption that $T_1, T_2, \ldots$ are stationarily distributed and ergodic; cf. [Lall86], (1.5), with $\chi = [0,\infty)$ while $\xi \colon [0,\infty)^{\mathbb{Z}} \to [0,\infty)$ is given by the projection onto the 0^{th} component.
Note: Reference [Lall86] considers doubly infinite sequences of random variables; in our notation $\ldots, T_{-1}, T_0, T_1, \ldots$.

4.3 Entropy and Guess Work

513 The central goal in the evaluation of TRNGs is to quantify the amount of randomness of the generated random numbers. In this section the concepts of entropy, guess work, and work factor are introduced, and their relation is pointed out.

4.3.1 Entropy

For the evaluation of RNGs, Shannon entropy (4.58) and min-entropy (4.59) play an outstanding role. Both can be viewed as special cases of Renyi entropy, a more general definition of entropy. Collision entropy (4.60) has some relevance, too. 514

Let X be a random variable that assumes values in the finite set $\Omega = \{\omega_1, \omega_2, \ldots, \omega_k\}$. The most general notion of entropy is the *Renyi* entropy H_α, where 515

$$H_\alpha(X) = \frac{1}{1-\alpha} \log_2 \sum_{i=1}^{k} \left(\text{Prob}\left(X = \omega_i\right)\right)^\alpha, \; 0 \leq \alpha < \infty\,. \tag{4.57}$$

Formula (4.57) comprises infinitely many different definitions of entropy. Its most important representatives are Shannon entropy, min-entropy, and collision entropy. For a given random variable X, the entropy values $H_\alpha(X)$ are monotonically decreasing in α.

The entropy $H_\alpha(X)$ only depends on the distribution μ of X. Thus, we synonymously use the notation $H_\alpha(\mu)$. 516

The special case $\alpha = 1$ yields the well-known Shannon entropy. In particular, *L'Hopital*'s rule then recovers the definition of Shannon entropy 517

$$H_1(X) = H(X) = -\sum_{i=1}^{k} \text{Prob}\left(X = \omega_i\right) \log_2\left(\text{Prob}\left(X = \omega_i\right)\right) \tag{4.58}$$

If $\text{Prob}(X = \omega_i) = 0$, by convention $\text{Prob}(X = \omega_i)\log_2(\text{Prob}(X = \omega_i)) = 0$. Usually, we use H in place of H_1 to indicate the Shannon entropy.

Shannon entropy $H = H_1$ is sometimes also called average entropy or simply entropy due to its prevalence in information theory. 518

The min-entropy represents a special case $\alpha = \infty$ 519

$$\lim_{\alpha\to\infty} H_\alpha(X) = -\log_2\left(\max_{1\leq i\leq k}\left\{\text{Prob}\left(X = \omega_i\right)\right\}\right) = H_{\min}(X)\,. \tag{4.59}$$

Besides $H_{\min}(\cdot)$ the notation $H_\infty(\cdot)$ is also common.

Finally, H_2 defines collision entropy. Let X and X' be two independent and identically-distributed random variables with values in a finite set Ω. The collision probability is $\text{Prob}\left(X = X'\right) = \sum_{x\in\Omega} \left(\text{Prob}(X = x)\right)^2$, and the collision entropy equals 520

$$H_2(X) = -\log_2\left(\sum_{\omega\in\Omega}\left(\text{Prob}\left(X = \omega\right)\right)^2\right). \tag{4.60}$$

521

The inequalities (4.61) quantify the relation between Shannon entropy, min-entropy, and collision entropy.

$$H_{\min} \le H_2 \le H_1, \quad H_{\min} \le H_2 \le 2H_{\min}\,. \tag{4.61}$$

By par. 515 the min-entropy is the most conservative entropy measure. Figure 6 plots H_1, H_2 and H_∞ for binary-valued random variables.

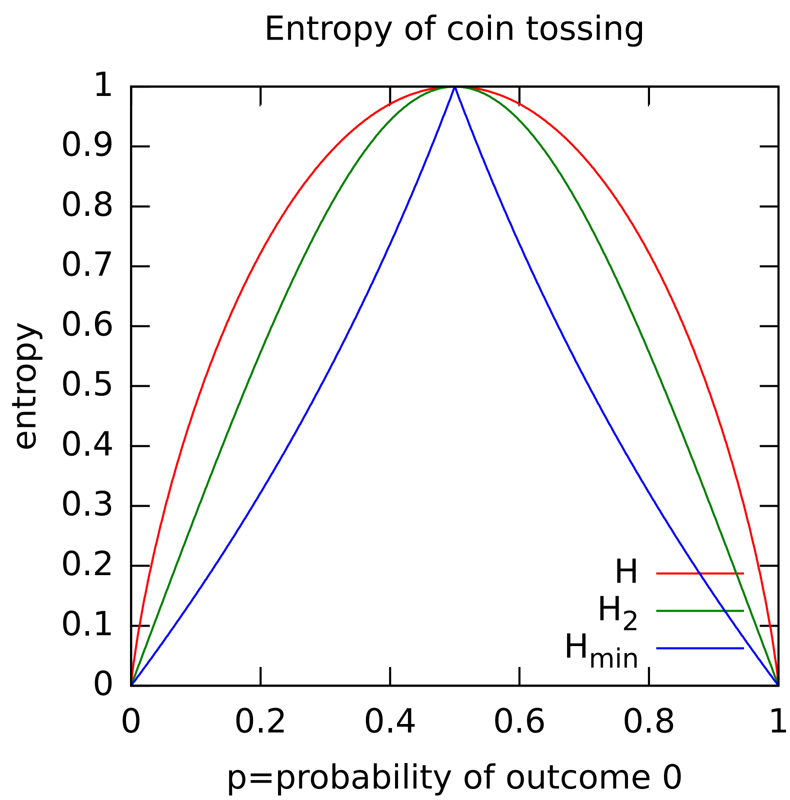

Figure 6: min-entropy $H_{\min}$ (blue), collision entropy H_2 (green), and Shannon entropy H (red) for binary-valued random variables

522 The *variation distance* between two probability distributions $\nu = (\nu(\omega_1), \ldots, \nu(\omega_k))$ and $\eta = (\eta(\omega_1), \ldots, \eta(\omega_k))$ on Ω is defined by

$$\|\nu - \eta\| := \max_{A \subseteq \Omega} \{|\nu(A) - \eta(A)|\}\,. \tag{4.62}$$

Note that $\|\nu - \eta\|$ is half of the L^1-distance. If $\eta = \bar{u} = \left(\frac{1}{k}, \ldots, \frac{1}{k}\right)$ (uniform distribution on Ω) then (4.62) simplifies to

$$\|\nu - \bar{u}\| = \nu\,(A) - \frac{|A|}{k}, \quad \text{with } A = \left\{ a \mid \nu\,(a) \ge \frac{1}{k},\ a \in \Omega \right\}. \tag{4.63}$$

523

(4.64) provides an inequality for the variation distance

$$\sum_{x\in\Omega} (\mathrm{Prob}\,(X=x))^2 \geq \frac{1+4\,\|\nu-u\|^2}{|\Omega|} \qquad ([\text{CFPZ09}]). \tag{4.64}$$

[Shannon entropy: joint entropy] Assume that X_1 and X_2 denote (not necessarily independent) random variables that assume values in Ω. Then the (joint) Shannon entropy of X_1 and X_2 is given by 524

$$H\,(X_1, X_2) = -\sum_{i,j=1}^{k} \mathrm{Prob}\,(X_1=\omega_i, X_2=\omega_j)\log_2\left(\mathrm{Prob}\,(X_1=\omega_i, X_2=\omega_j)\right) = \tag{4.65}$$

$$-\sum_{i,j=1}^{k} \mathrm{Prob}\,(X_2=\omega_j \mid X_1=\omega_i)\,\mathrm{Prob}\,(X_1=\omega_i)\log_2\left(\mathrm{Prob}\,(X_2=\omega_j \mid X_1=\omega_i)\,\mathrm{Prob}\,(X_1=\omega_i)\right)$$

Since $\log_2(ab) = \log_2(a) + \log_2(b)$ (here, $a = \mathrm{Prob}(X_2=\omega_j \mid X_1=\omega_i)$ and $b = \mathrm{Prob}(X_1=\omega_i)$), by rearranging the terms we obtain the useful functional equation

$$H\,(X_1, X_2) = H\,(X_2 \mid X_1) + H\,(X_1) \quad \text{where} \tag{4.66}$$

$$H(X_2 \mid X_1) = -\sum_{i=1}^{k} \mathrm{Prob}\,(X_1=\omega_i)\,H\,(X_2 \mid X_1=x_1)\ . \tag{4.67}$$

Here, $H(X_2 \mid X_1=x_1)$ denotes the entropy of X_2 under the condition that X_1 assumes the value x_1. The term $H(X_2 \mid X_1)$ is the *conditional entropy* of X_2 und X_1. It quantifies the average entropy of X_2 when X_1 is known. Clearly,

$$\min_{1\leq i\leq k}\{H(X_2 \mid X_1=\omega_i)\} \leq H(X_2 \mid X_1) \leq \max_{1\leq i\leq k}\{H(X_2 \mid X_1=\omega_i)\}\,. \tag{4.68}$$

The functional equation (4.66) generalizes to several random variables. More precisely, 525

$$H\,(X_1,\ldots,X_{m+1}) = H\,(X_1,\ldots,X_m) + H\,(X_{m+1} \mid X_1,\ldots,X_m)\ . \tag{4.69}$$

Formula (4.69) is well-known and very useful for the evaluation of physical RNGs.

Depending on the distribution of the random variables $X_1, X_2, \ldots$, the formula for the conditional entropy in (4.69) may simplify considerably. In particular, 526

$$H\,(X_{m+1} \mid X_1,\ldots,X_m) \leq H\,(X_{m+1})\ , \tag{4.70}$$

$$H\,(X_{m+1} \mid X_1,\ldots,X_m) = H\,(X_{m+1}) \quad \text{if } X_1, X_2,\ldots \text{ are independent}\,, \tag{4.71}$$

$$H\,(X_{m+1} \mid X_1,\ldots,X_m) = H\,(X_{m+1} \mid X_m) \quad \text{if } X_1, X_2,\ldots \text{ are Markovian.} \tag{4.72}$$

Assume that $X_1, X_2, \ldots$ defines a homogeneous ergodic Markov chain with limiting distribution ν. If the Markov chain has already (at least almost) reached equilibrium, then (4.72) simplifies to 527

$$H\,(X_{m+1} \mid X_m) = \sum_{i=1}^{k} \nu(\omega_i) H\,(X_{m+1} \mid X_m=\omega_i) \tag{4.73}$$

where $H(X_{m+1} \mid X_m = \omega_i)$ depends on the i^{th} row of the transition matrix.

528 If the sequence of random variables $X_1, X_2, \ldots$ is stationary for each shift parameter t and for each $h \leq m$

$$H\left(X_{m+t+1} \mid X_{m+t-h+1}, \ldots, X_{m+t}\right) = H\left(X_{m+1} \mid X_{m-h+1}, \ldots, X_m\right) . \tag{4.74}$$

529 For Renyi parameters $\alpha \neq 1$ no equivalent to (4.66) exist. The inequality (4.68), however, generalizes to

$$\begin{aligned}&\min_{i_1,\ldots,i_m \leq k} \{H_\alpha(X_{m+1} \mid X_1 = \omega_{i_1}, \ldots, X_m = \omega_{i_m})\} \leq H_\alpha(X_{m+1} \mid X_1, \ldots, X_m) \leq \\ &\max_{i_1,\ldots,i_m \leq k} \{H_\alpha(X_{m+1} \mid X_1 = \omega_{i_1}, \ldots, X_m = \omega_{i_m})\} .\end{aligned} \tag{4.75}$$

For independent random variables $X_1, \ldots, X_m$, we have

$$H_\alpha\left(X_1, \ldots, X_m\right) = H_\alpha\left(X_1\right) + \cdots + H_\alpha\left(X_m\right) . \tag{4.76}$$

Pars. 531 to 539 consider min-entropy in the context of homogeneous Markov chains, improving the general inequality (4.75).

530 Assume that the binary-valued random variable X is $B(1, 0.5 + 0.5\epsilon)$-distributed. If $|\epsilon|$ is small the Taylor expansion of the natural logarithm $\log(\cdot)$ yields

$$\log_2(0.5 \pm 0.5\epsilon) = \log_2(0.5(1 \pm \epsilon)) = \log_2(0.5) + \frac{\log(1 \pm \epsilon)}{\log(2)} = -1 + \frac{\pm\epsilon - 0.5\epsilon^2 + O(\epsilon^3)}{\log(2)} . \tag{4.77}$$

Elementary, but careful computations show

$$H_{\min}(X) = 1 - \frac{\epsilon - 0.5\epsilon^2}{\log(2)} + O(\epsilon^3) \quad \text{and} \quad H(X) = 1 - \frac{0.5\epsilon^2}{\log(2)} + O(\epsilon^3) . \tag{4.78}$$

531 [Markov chain, min-entropy] Assume that $X_1, X_2, \ldots$ defines a homogeneous ergodic Markov chain on the state space Ω with transition matrix P and limiting distribution ν. If the distribution of X_n has (at least almost) reached equilibrium ν then the joint min-entropy is given by

$$\begin{aligned}&H_{\min}(X_{n+1}, \ldots, X_{n+m}) = \\ &-\log_2\left(\max\left\{\nu(\omega) p_{\omega,\omega_{n+2}} \cdots p_{\omega_{n+m-1},\omega_{n+m}} \mid \omega, \omega_{n+2}, \ldots, \omega_{n+m} \in \Omega\right\}\right) .\end{aligned} \tag{4.79}$$

The average gain of min-entropy per random number is given by

$$\frac{H_{\min}(X_{n+1}, \ldots, X_{n+m})}{m} . \tag{4.80}$$

532 [Markov chain, min-entropy] Analogously to the Shannon entropy, we define the average *conditional* min-entropy of m consecutive random numbers by

$$\begin{aligned}&H_{\min}(X_{n+1}, \ldots, X_{n+m} \mid X_n) = \sum_{\omega \in \Omega} \nu(\omega) H_{\min}(X_{n+1}, \ldots, X_{n+m} \mid X_n = \omega) = \\ &-\sum_{\omega \in \Omega} \nu(\omega) \log_2\left(\max\left\{p_{\omega,\omega_{n+1}} \cdots p_{\omega_{n+m-1},\omega_{n+m}} \mid \omega_{n+1}, \ldots, \omega_{n+m} \in \Omega\right\}\right) .\end{aligned} \tag{4.81}$$

The average conditional gain of min-entropy per random number equals

$$\frac{H_{\min}(X_{n+1},\ldots,X_{n+m} \mid X_n)}{m}\,. \tag{4.82}$$

[Markov chain] Special cases: 533
(i) $|\Omega| = 2$: Then the random variables $X_1, X_2, \ldots$ quantify the distribution of random bits (e.g., of raw random numbers).
(ii) $X_1, X_2, \ldots$ forms a homogeneous k-step Markov chain. Then the random vectors $\vec{Y}_1, \vec{Y}_2, \ldots$ given by $\vec{Y}_j = (X_j, \ldots, X_{j+k-1})^t$ form a homogeneous Markov chain on the state space Ω^k.

[Markov chain, $|\Omega| = 2$] The next goal is to develop easy-to-use formulae for (4.81) (and thus for (4.82)) for arbitrary $m \geq 1$. After re-ordering any product $p_{\omega,\omega_1} \cdots p_{\omega_{m-1},\omega_m}$ of transition probabilities in (4.81) is of the form to $p_{00}^a p_{01}^b p_{10}^c p_{11}^d$ with integers $a, b, c, d \geq 0$ such that $a + b + c + d = m$. Our task is to determine the maximum for both $\omega = 0$ and $\omega = 1$. This requires case distinctions for the parameters $p_{00}, p_{01}, p_{10}, p_{11}$, or more precisely, only for p_{01}, p_{10} because $p_{00} = 1 - p_{01}$ and $p_{11} = 1 - p_{10}$. 534

[Markov chain, $|\Omega| = 2$, ctd.] To simplify the notation we introduce the following definition 535

$$\max_{\mathrm{P},2}(\omega, m) := \max\left\{p_{\omega,\omega_1} \cdots p_{\omega_{m-1},\omega_m} \mid \omega, \omega_1, \ldots, \omega_m \in \Omega\right\}\,. \tag{4.83}$$

We extend (4.83) to the case $m = 0$ by $\max_{\mathrm{P},2}(\omega, 0) := 1$. Below, we assume that $m \geq 1$ and $p_{01} \leq p_{10}$. The case where $p_{01} \geq p_{10}$ can be handled analogously; concrete formulae can be derived without further work by relabeling the state space $\Omega = \{0, 1\}$.

- Case I: $p_{01} \leq p_{10} \leq 0.5$. Thus, $p_{01} \leq p_{10} \leq 0.5 \leq p_{11} \leq p_{00}$, and
$$\max_{\mathrm{P},2}(0, m) := p_{00}^m\,, \quad \max_{\mathrm{P},2}(1, m) = \max\{p_{10}p_{00}^{m-1}, p_{11}^m\}\,. \tag{4.84}$$
- Case II: $0.5 \leq p_{01} \leq p_{10}$. Thus, $p_{11} \leq p_{00} \leq 0.5 \leq p_{01} \leq p_{10}$, and
$$\max_{\mathrm{P},2}(0, m) := \begin{cases} (p_{01}p_{10})^{m/2} & \text{for even m} \\ (p_{01}p_{10})^{(m-1)/2}\, p_{01} & \text{for odd m} \end{cases} \tag{4.85}$$
$$\max_{\mathrm{P},2}(1, m) := \begin{cases} (p_{10}p_{01})^{m/2} & \text{for even m} \\ (p_{10}p_{01})^{(m-1)/2}\, p_{10} & \text{for odd m} \end{cases} \tag{4.86}$$
- Case III: $p_{01} \leq 0.5 \leq p_{10}$. Hence $p_{01}, p_{11} \leq 0.5 \leq p_{10}, p_{00}$. We distinguish two subcases:
  - Subcase III$_1$: $p_{00} \geq p_{10}$. Thus $p_{01} \leq p_{11} \leq 0.5 \leq p_{10} \leq p_{00}$. Then
  $$\max_{\mathrm{P},2}(0, m) := p_{00}^m\,, \quad \max_{\mathrm{P},2}(1, m) = p_{10}p_{00}^{m-1} \tag{4.87}$$
  - Subcase III$_2$: $p_{00} \leq p_{10}$. Thus, $p_{11} \leq p_{01} \leq 0.5 \leq p_{00} \leq p_{10}$. Then
  $$\max_{\mathrm{P},2}(0, m) := \begin{cases} \max\left\{p_{00}^m, (p_{01}p_{10})^{m/2}\right\} & \text{for even m} \\ \max\left\{p_{00}^m, (p_{01}p_{10})^{(m-1)/2}\, p_{00}\right\} & \text{for odd m} \end{cases} \tag{4.88}$$
  $$\max_{\mathrm{P},2}(1, m) := \begin{cases} \max\left\{p_{10}p_{00}^{m-1}, (p_{10}p_{01})^{m/2}\right\} & \text{for even m} \\ \max\left\{p_{10}p_{00}^{m-1}, (p_{10}p_{01})^{(m-1)/2}\, p_{10}\right\} & \text{for odd m} \end{cases} \tag{4.89}$$

536 [Markov chain, $|\Omega| = 2$, ctd.] The results from par. 535 allow the simplification of (4.80) and (4.82). The average gain of min-entropy per random bit of $X_{n+1}, \ldots, X_{n+m}$ equals

$$\frac{H_{\min}(X_{n+1}, \ldots, X_{n+m})}{m} = \frac{-\log_2\left(\max\left\{\nu(0)\max_{\mathrm{P},2}(0, m-1), \nu(1)\max_{\mathrm{P},2}(1, m-1)\right\}\right)}{m}. \tag{4.90}$$

Similarly,

$$\frac{H_{\min}(X_{n+1}, \ldots, X_{n+m} \mid X_n)}{m} = \frac{-\sum_{\omega\in\Omega}\nu(\omega)\log_2\left(\max_{\mathrm{P},2}(\omega, m)\right)}{m}. \tag{4.91}$$

537 [Markov chain, $|\Omega| = 2$, ctd.] Par. 536 provides manageable formulae to determine the average min-entropy per bit for Markov chains on the state space $\Omega = \{0, 1\}$. Their derivation require careful considerations with case distinctions. It is obvious that for 2-step Markov chains the necessary efforts increase significantly. Since in our context we are usually interested in the average entropy within long sequences we may apply the simpler formula (4.92), neglecting complicating 'boundary effects', which may play a role for small m. In fact, the functional equation of the logarithm function, $\log_2(pq) = \log_2(p) + \log_2(q)$, yields

$$\lim_{m\to\infty}\frac{H_{\min}(X_{n+1}, \ldots, X_{n+m})}{m} = -\max\{\log_2(p_{00}), \log_2(p_{11}), 0.5\log_2(p_{01}p_{10})\}. \tag{4.92}$$

538 [Markov chain] Consider a Markov chain on a finite state space Ω with transition matrix P. We call $(\omega_1, \ldots, \omega_\ell, \omega_{\ell+1})$ a loop if $\omega_1 = \omega_{\ell+1}$ while $\omega_1, \ldots, \omega_\ell$ are mutually distinct. Here, ℓ denotes the length of the loop. Formula (4.93) equals Theorem 2 in [ASPB+18].

$$\lim_{m\to\infty}\frac{H_{\min}(X_{n+1}, \ldots, X_{n+m})}{m} = \min_{\ell}\min_{(\omega_1,\ldots,\omega_\ell,\omega_{\ell+1})\in\mathcal{C}_\ell}\frac{1}{\ell}\sum_{j=1}^{\ell}\log_2\left(\frac{1}{p_{\omega_j\omega_{j+1}}}\right). \tag{4.93}$$

Here, $\mathcal{C}_\ell$ denotes the set of all loops of length ℓ.
Note: For $\Omega = \{0, 1\}$ there are two loops of length 1 (namely, $(0, 0), (1, 1)$) and two loops of length 2 (namely, $(0, 1, 0), (1, 0, 1)$). Substituting into (4.93) yields (4.92).

539 [Markov chain, ctd.] If $X_0, X_1, \ldots$ forms a homogeneous k-step Markov chain on Ω then $\vec{Y}_0 = (X_0, X_1, \ldots, X_{k-1})$, $\vec{Y}_1 := (X_1, X_2, \ldots, X_k)$ is a homogeneous 1-step Markov chain on the product state space Ω^k. In particular, (4.93) can be applied to $\vec{Y}_0, \vec{Y}_1, \ldots$.

4.3.2 Guess Work and Work Factor

540 Although guesswork and work factor do not appear in the specificatons of the functionality classes, we briefly treat these concepts.

541 As in Subsect. 4.3.1 X denotes a random variable that assumes values in $\Omega = \{\omega_1, \omega_2, \ldots, \omega_k\}$. W.l.o.g. we may assume that

$$\operatorname{Prob}(X = \omega_1) \geq \operatorname{Prob}(X = \omega_2) \geq \ldots \geq \operatorname{Prob}(X = \omega_k). \tag{4.94}$$

A reasonable goal is to estimate the effort to guess the outcome of an experiment that is viewed as a realization of X.

If the random variable X has distribution ν, we set $\nu(\omega_j) := \text{Prob}(X = \omega_j)$ to simplify the notation. In particular, $\nu := (\nu(\omega_1), \ldots, \nu(\omega_k))$. 542

The λ-*work-factor* $w_\lambda(X)$ denotes the minimum number of guesses to get the correct result with probability $\geq \lambda$ $(0 < \lambda < 1)$ if the optimal guessing strategy (beginning with $\omega_1, \omega_2, \ldots$) is applied. That is, 543

$$w_\lambda(X) = \min \left\{ k \mid \sum_{i=1}^{k} \nu(\omega_i) \geq \lambda \right\}. \tag{4.95}$$

The *guess work* $W(X)$ denotes the expected number of guesses until a success if the optimal guessing strategy is applied 544

$$W(X) = \sum_{i=1}^{n} i\nu(\omega_i) \ . \tag{4.96}$$

The guess work and the λ-work-factor (for a suitable parameter λ) seem to be appropriate criteria to assess the strength of a TRNG that is used for cryptographic applications. However, in many scenarios of practical relevance, it can be very difficult to sort the probabilities of the admissible outcomes in descending order as in (4.94), in particular if X is a random vector $(X_1, \ldots, X_m)$ with dependent components. Usually, the calculation of the entropy (or at least the determination of a useful lower bound) is easier. In the next paragraphs we explain the relation between entropy, guess work, and work factor. 545

For $\lambda = 0.5$ the work factor of the optimal strategy meets the following inequality [Plia99] 546

$$\left\lfloor \frac{1}{2 \max \{\nu(x_j) \mid 1 \leq j \leq n\}} \right\rfloor \leq w_{\frac{1}{2}}(X) \leq \lceil (1 - \|\nu - \bar{u}\| \cdot n) \rceil \ . \tag{4.97}$$

As above, $\bar{u}$ denotes the uniform distribution on Ω.

For the most general case, the following inequality provides tight bounds for the guesswork 547

$$\frac{k}{2} \|\nu - \bar{u}\| \leq \frac{k-1}{2} - W(X) \leq k \|\nu - \bar{u}\| \,. \tag{4.98}$$

A memoryless binary-valued stationary random source can be described by independent identically $B(1,p)$-distributed random variables $X_1, X_2, \ldots, X_n$. The guesswork for n random bits, or equivalently for the random vector $X = (X_1, \ldots, X_n)$, may be estimated by the Shannon entropy [Maur92] 548

$$\log_2 w_{\frac{1}{2}}(X) \approx n \cdot H_1(X_1) \,. \tag{4.99}$$

More generally, for an ergodic stationary binary random source, the relation between the guesswork and the length of a sequence tends asymptotically to Shannon entropy [Maur92] 549

$$\lim_{n\to\infty} \frac{\log_2 w_\alpha(X)}{n} = H(X) \,, \text{ for } 0 < \alpha < 1. \tag{4.100}$$

550 [Example] Let ν denote a probability measure on $\Omega = \{0,1\}^{128}$ such that $\nu((0,\ldots,0)) = 0.5$, $\nu((1,\ldots,1)) = 2^{-128}$, and $\nu(\omega) = 2^{-129}$ else. Then $H_{\min}(X) = 1$, whereas

$$H(X) = -\left(0.5\log_2(0.5) + 2^{-128}\log_2(2^{-128}) + (2^{128}-2)\cdot 2^{-129}(\log_2(2^{-129}))\right) = \tag{4.101}$$
$$0.5 + 2^{-128}\cdot 128 + (2^{128}-2)\cdot 2^{-129}\cdot 129 = 0.5 + 64.5 - 2^{-128} \approx 65\,.$$

For $\lambda \leq 0.5$ we have $w_\lambda(X) = 1$, which would be disastrous for cryptographic applications. On the other hand,

$$W(X) = 1\cdot 0.5 + 2\cdot 2^{-128} + \sum_{i=3}^{2^{128}} i\cdot 2^{-129} = 0.5 + 2^{-127} + \left(\frac{2^{128}(2^{128}+1)}{2} - 3\right)2^{-129} =$$
$$2^{126} + 0.75 + 2^{-129} \approx 2^{126}\,. \tag{4.102}$$

is rather large. Note that the uniform distribution has guesswork $W(X) = 2^{127} + 0.5 \approx 2^{127}$.

551 [Example] Let ν denote a probability measure on $\Omega = \{0,1\}^{128}$ such that $\nu((0,\ldots,0)) = 2^{-127}$, $\nu((1,\ldots,1)) = 0$, and $\nu(\omega) = 2^{-128}$ else. Then $H_{\min}(\nu) = 127$, while

$$H(X) = -\left(2^{-127}\log_2(2^{-127}) + 0 + (2^{128}-2)\cdot 2^{-128}(\log_2(2^{-128}))\right) = \tag{4.103}$$
$$2^{-127}\cdot 127 + (2^{128}-2)\cdot 2^{-128}\cdot 128 = 128 - 2^{-127} \approx 128\,.$$

For $\lambda = 2^{-127}$ we have $w_\lambda(X) = 1$ (in place of $= 2$ for the uniform distribution), but for $\lambda = 2^{-100}$, for instance, $w_\lambda(X) = 2^{28} - 1$ (instead of 2^{28} for the uniform distribution). Furthermore, the guesswork equals

$$W(X) = 1\cdot 2^{-127} + \sum_{i=2}^{2^{128}-1} i\cdot 2^{-128} = 2^{-127} + \left(\frac{(2^{128}-1)\cdot 2^{128}}{2} - 1\right)2^{-128} =$$
$$2^{127} - 0.5 + 2^{-128} \approx 2^{127}\,, \tag{4.104}$$

which is very close to the guesswork of the uniform distribution.

552 The example in par. 550 shows that for very unbalanced distributions Shannon entropy and the guesswork may tremendously overestimate the resistance against guessing attacks. On the other hand in the example in par. 551, Shannon entropy and the guesswork provide a realistic assessment of the strength against guessing attacks, while the min-entropy underestimates this strength unless extremely small parameters λ are concerned.

553 Example: Assume that the random variables $X_1,\ldots,X_n$ are iid $B(1,p)$-distributed and $\vec{X} = (X_1,\ldots,X_n)$. Since the random variables X_i are iid, we obtain $H_1(\vec{X}) = n\cdot H_1(X_1) = -n\cdot(p\log_2(p) + (1-p)\log_2(1-p))$ and $H_{min}(\vec{X}) = -\log_2(\max\{p^n,(1-p)^n\}) = -n\log_2(\max\{p,1-p\})$. If $p \geq 0.5$ the most likely vector is $(1,\ldots,1)$ but $(0,\ldots,0)$ otherwise. Furthermore, $Y = \mathrm{ham}(X_1,\ldots,X_n)$, the Hamming weight of the random vector $(X_1,\ldots,X_n)$, is $B(n,p)$-distributed, and $\mathrm{Prob}(Y=y) = \binom{n}{y}p^y\cdot(1-p)^{n-y}$.

554 Example from par. 553 continued: The work factor $w_\lambda(\vec{X})$ (4.95) can be efficiently computed because $\mathrm{Prob}(\vec{X} = \vec{x})$ only depends on the Hamming weight of $\vec{x}$. In particular, only $n{+}1$ different

probabilities occur. W.l.o.g. we may assume that $p \geq 0.5$. At first, for success probability λ one determines

$$\alpha(\lambda) := \max\left\{ i \geq 0 \mid \sum_{j=i}^{n} \binom{n}{j} p^j (1-p)^{n-j} \geq \lambda \right\}. \tag{4.105}$$

Then

$$w_\lambda(\vec{X}) = \sum_{j=\alpha(\lambda)+1}^{n} \binom{n}{j} + \left\lceil \frac{\lambda - \sum_{j=\alpha(\lambda)+1}^{n} \binom{n}{j} p^j (1-p)^{n-j}}{p^{\alpha(\lambda)}(1-p)^{n-\alpha(\lambda)}} \right\rceil. \tag{4.106}$$

Unless n or j are rather small, Stirling's approximation formula (4.29) and (4.30) may be applied to compute the factorials and the binomial coefficients. Table 1 provides concrete figures.

Table 1: work factor $w_\lambda(\vec{X} = (X_1, \ldots, X_n))$ for several success probabilities λ: $X_1, \ldots, X_{128}$ are iid $B(1,p)$-distributed; the top row describes the ideal case.

	λ	$H(\vec{X})$	$\log_2\left(w_\lambda(\vec{X})\right)$	$H_{min}(\vec{X})$
$p = 0.500$	2^{-80}	128	48	128
	2^{-40}		88	
$p = 0.501$	2^{-80}	128.000	47.688	127.631
	2^{-40}		87.773	
$p = 0.503$	2^{-80}	127.997	47.066	126.895
	2^{-40}		87.319	
$p = 0.507$	2^{-80}	127.982	45.822	125.433
	2^{-40}		86.403	
$p = 0.510$	2^{-80}	127.963	44.880	124.343
	2^{-40}		85.706	

4.4 Random mappings

This section treats random mappings. We focus on aspects that are relevant in the context of AIS 20 and AIS 31. Section 4.4.1 summarizes well-known statistical properties for the iteration of random mappings. In Subsection 4.4.2 the impact of randomly selected mappings on the work factor and, in particular, on the entropy is analyzed. 555

These results shall be used to support the security evaluation of DRNGs (functionality classes DRG.2, DRG.3 and DRG.4) and of cryptographic post-processing in the context of PTG.3- and NTG.1-evaluations. 556

[Notation] In this section A, A_1, and A_2 denote finite sets and $\mathcal{F}_{A_1,A_2} := \{f' \colon A_1 \to A_2\}$. For given sets A_1 and A_2 a random mapping F is a random variable that assumes values uniformly in a specified subset $V \subseteq \mathcal{F}_{A_1,A_2}$. 557

4.4.1 Iteration of random mappings: statistical properties

558

In this subsection $A_1 = A_2 = A$, and F is a random mapping that is uniformly distributed on a subset $V \subseteq \mathcal{F}_{A,A}$. We focus on the special cases $V = \mathcal{F}_{A,A}$ and $V = Perm_A$, the set of all permutations on A (i.e., the bijective elements of $\mathcal{F}_{A,A}$).

559 In the context of AIS 20 the results of this subsection may be applicable to the state transition functions of DRNGs. These results are particularly interesting for pure DRNGs, while for hybrid DRNGs the situation should be more favorable anyway since additional input usually causes the internal state to 'jump' between cycles of the pure version (i.e., without additional input) of the DRNG.

560 For a given $\omega \in \Omega$ the term $F(\omega)$ denotes a fixed mapping in V. We 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 $t_0, t_1, \ldots$ describes a path in A that ends in a cycle. The functional graph consist of components, each of which consists of one cycle that is connected with several trees (0 trees are possible).

561 Table 2 collects well-known results on random mappings on $\mathcal{F}_{A,A}$; see, e.g. [Flod89] for details. If $|A| = n$ then $|\mathcal{F}_{A,A}| = n^n$.

562 Table 3 collects well-known results on random permutations (random bijections) that are chosen uniformly from the set of all $n!$ permutations $V = Perm_A$ (cf. [Golo64] for details). If $|A| = n$ then $|Perm_A| = n!$.

4.4.2 Impact on the work factor and on the entropy

563 In this subsection we analyze the impact of randomly selected mappings on the work factor (pars. 573 to 597), Shannon entropy (pars. 599 to 604), and min-entropy (pars. 605 to 621). The contributions to Shannon entropy and in particular to min-entropy are of most importance in context of the AIS 31.

564 In this subsection A_1, A_2 denote finite sets with cardinality $|A_1| = b_1$ and $|A_2| = b_2$. In particular, $|\mathcal{F}_{A_1,A_2}| = b_1^{b_2}$.

565 [random variables] In this subsection F denotes a random variable that assumes values uniformly in $\mathcal{F}_{A_1,A_2} := \{f' \colon A_1 \to A_2\}$. The random variable X assumes values in A_1, and F and X are independent. Unless otherwise stated in this subsection, X is uniformly distributed on A_1 while X' is allowed to be non-uniformly distributed. Furthermore, U denotes a random variable that is uniformly distributed on A_2.

566 The results of this subsection shall support the evaluation of requirements PTG.3.6. This in particular concerns the impact of data compression.

567 The typical scenario for PTG.3-compliant PTRNGs is the following: A PTG.2-compliant PTRNG has generated n-bit intermediate random numbers $x_1, x_2, \ldots$, which are interpreted as realizations of random variables $X_1, X_2, \ldots$. (The n-bit random number x_j can be the concatenation of

Table 2: Statistics of random mappings on A, $|A| = n$; cf. [Flod89]

Characteristic	**Expected value as $n \to \infty$**	**Definition and comments**
Number of components	$\frac{1}{2} \ln n$	A component consists of one cycle and several trees connected to this cycle. (0 trees are possible.)
Component size	$\frac{2n}{3}$	of a randomly selected point
Largest component	$\approx 0.75782n$	
Number of cyclic nodes	$\sqrt{\frac{\pi n}{2}}$	$\sqrt{\frac{\pi n}{2}} \approx 1.253314\sqrt{n}$
Cycle length (μ)	$\sqrt{\frac{\pi n}{8}}$	The number of edges in the cycle is called the cycle length of t, denoted $\mu(t)$, $\sqrt{\frac{\pi n}{8}} \approx 0.626657\sqrt{n}$.
Maximum cycle length	$\approx 0.78248\sqrt{n}$	
Tail length (λ)	$\sqrt{\frac{\pi n}{8}}$	The number of edges in the path to the cycle is called the tail length of t, denoted $\lambda(t)$, $\sqrt{\frac{\pi n}{8}} \approx 0.626657\sqrt{n}$.
Maximum tail length	$\approx 1.73746\sqrt{n}$	
Rho length (ρ)	$\sqrt{\frac{\pi n}{2}}$	$\rho(t) = \lambda(t) + \mu(t)$, number of steps until a node on the path repeats, $\sqrt{\frac{\pi n}{2}} \approx 1.253314\sqrt{n}$.
Maximum rho length	$\approx 2.4119\sqrt{n}$	
Tree size	$\frac{n}{3}$	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	$\left(1 - e^{-1}\right) n$	$\|f(A)\|$ = number of nodes that have a predecessor, $\left(1 - e^{-1}\right) n \approx 0.632121n$.
Number of k-th iterate image points	$(1 - r_k)\, n$, $r_0 = 0$, $r_{k+1} = \exp(-1 + r_k)$	$\left\|f^k(A)\right\|$ = number of nodes after application of f^k.
Predecessor size	$\sqrt{\frac{\pi n}{8}}$	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.

Table 3: Statistics of random permutations on A, $|A| = n$, cf. [Golo64; PuWi68]

Characteristic	**Expected value as $n \to \infty$**	**Distribution as $n \to \infty$**
Number of cycles	$\ln n + C + o(1)$ $C =$ $0.57721566\ldots$	Number ω_n of cycle of the permutation $\mathrm{Prob}\,(\omega_n = k) = \frac{\exp\left(-\frac{(k-\ln n)^2}{2\ln n}\right)}{\sqrt{2\pi \ln n}}\,(1 + o(1))$ normal distribution $(\ln n, \ln n)$.
Cycle length	$\frac{n}{\ln n + C + o(1)}$	Number ω_{nl} of cycle of length l $\mathrm{Prob}\,(\omega_{nl} = k) = P_{1/l}(k) = \frac{\exp(-\frac{1}{l})}{l^k k!}$ Poisson distribution with parameter $\tau = \frac{1}{l}$.
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 $\mathrm{Prob}\,(\omega\,(x) = k) = \frac{1}{n}$.

n random bits.) The current intermediate random number x_j is mixed as additional input into S_{req} of the cryptographic post-processing (a DRG.3-compliant DRNG), and finally also into the internal state; alternatively, x_j provides seed material for the seeding procedure. If s_j denotes the current internal state S of the cryptographic post-processing in the notation of Chap. 3, the next values in S_{req} and S can be described by the random variables $\phi_{req}(s_j, X_j)$ and $\phi(s_j, X_j)$, respectively. Similarly, the outputted internal random number r_j can be interpreted as a realization of the random variable $\psi(\phi_{req}(s_j, X_j))$. Let the mapping $f_s\colon A_1 = \{0,1\}^n \to A_2 = \{0,1\}^m$ be given by $f_s(x) := \psi(\phi_{req}(s, x))$. This yields a sequence of (different) functions $f_{s_1}, f_{s_2} \ldots \in \mathcal{F}_{A_1,A_2}$. This motivates the study of random mappings with regard to evaluations of PTRNGs of class PTG.3.

568 Since X models the output of a PTG.2-compliant PTRNG, we may assume that X is 'nearly' uniformly distributed. This justifies a study of the case where X is uniformly distributed on A_1. Furthermore, pars. 614 to 619 consider non-uniform distributions.

569 In view of functionality class PTG.3, we are interested in the impact of the cryptographic post-processing on the stochastic properties of the internal random numbers. This comprises Shannon entropy and min-entropy. Additionally, we consider the impact on the work factor.

570 For a fixed mapping $f \in \mathcal{F}_{A_1,A_2}$ the term $f(X)$ describes the transformed random variable X. Similarly, $F(X)$ denotes the random variable that is given when a mapping $f \in \mathcal{F}_{A_1,A_2}$ is uniformly selected (modeled by the random variable F) and applied to the random variable X, which models the intermediate random numbers. (The pair of random variables (X, F) assumes values in $(A_1, \mathcal{F}_{A_1,A_2})$ while $F(X)$ assumes values in A_2.) In particular, $H(F(X))$ and $H_{min}(F(X))$ denote the Shannon entropy and the min-entropy of $F(X)$, respectively.

571 Below, we assume that a mapping from $\mathcal{F}_{A_1,A_2}$ is selected randomly. Many properties of random mappings are 'typical' in the sense that they are shared by 'nearly all' mappings. This property is important for stateless post-processing algorithms (with a fixed cryptographic out-

put function) and for cryptographic post-processing algorithms, where the adversary knows the complete current internal state (which is the most favourable scenario from the adversary's point of view). For a PTG.3-compliant PTRNG the output function can be interpreted as a random mapping on the intermediate random numbers that is parametrized by the current internal state of the post-processing algorithm; see par. 567. If an adversary hasn't any information on the internal state from the standpoint of security, the situation is more favorable than for a fixed randomly selected mapping. Moreover, due to the constantly changing internal state, this can be interpreted as an averaging operation.

For functionality class NTG.1 the situation is similar to class PTG.3 in the sense that truly random data are post-processed. However, for NTG.1-compliant NPTRNGs, usually the distribution of the raw random numbers has only a little entropy per data bit, which requires a higher compression rate; cf. par. 619. 572

Impact on the work factor 573

Although the work factor does not appear in the class requirements of PTG.2 and PTG.3, we at first consider the impact on the work factor. More precisely, we determine the expected (average) work factor of $F(X)$ in A_2 and compare it to the work factor of a uniformly distributed random variable U on A_2. Furthermore, we determine the variance of the work factor.

Let $f \in \mathcal{F}_{A_1,A_2}$ be fixed for the moment. For $s \in \mathbb{N}_0$ we introduce the definitions 574

$$V_{(f)s} := \left\{z \in A_2 : \left|f^{-1}(z)\right| = s\right\} \quad \text{and} \quad v_{(f)s} := \left|V_{(f)s}\right| . \tag{4.107}$$

That is to say, $V_{(f)s}$ denotes the set of elements of A_2 that have exactly s pre-images, and $v_{(f)s}$ quantifies its cardinality.

Since X is uniformly distributed on A_1, we have $\operatorname{Prob}(f(X) = z) = \frac{s}{b_1}$ for each $z \in V_{(f)s}$. As an immediate consequence 575

$$\operatorname{Prob}\left(f(X) \in V_{(f)s}\right) = \frac{s v_{(f)s}}{b_1} \quad \text{and} \quad \operatorname{Prob}\left(f(X) \in \bigcup_{s=r}^{b_1} V_{(f)s}\right) = \sum_{s=r}^{b_1} \frac{s v_{(f)s}}{b_1} . \tag{4.108}$$

In our context the values $f(\cdot)$ are random numbers. The best guessing strategy for an adversary (without additional knowledge) is to try those $a_2 \in A_2$ first that have the most pre-images under f. Thus, the work factor corresponding to the success probability $\operatorname{Prob}\left(f(X) \in \bigcup_{s=r}^{b_1} V_{(f)s}\right)$ is given by $\sum_{s=r}^{b_1} s v_{(f)s}$, since by assumption, X is uniformly distributed on A_1. 576

[Notation] Subsequently, $E_F(\cdot)$ and $\operatorname{Prob}_X(\cdot)$ denote the expectation with regard to the random mapping F and the probability with regard to the random variable X, respectively. In particular, for a mapping $h\colon \mathcal{F}_{A_1,A_2} \to \mathbb{R}$, this means 577

$$E_F(h(F)) = \frac{1}{\left|\mathcal{F}_{A_1,A_2}\right|} \sum_{f \in \mathcal{F}_{A_1,A_2}} h(f) . \tag{4.109}$$

578

The term

$$e_r := \mathrm{Prob}_X \left(\left|F^{-1}\left(F(X)\right)\right| \geq r\right) \tag{4.110}$$

quantifies the average probability that a realization of $F(X)$ has $\geq r$ pre-images.

579 If $s \neq t$ then $V_{(f)s}$ and $V_{(f)t}$ are disjoint. Hence,

$$\begin{aligned}
e_r &= E_F\left(\mathrm{Prob}_X\left(F(X) \in \bigcup_{s=r}^{b_1} V_{(F)s}\right)\right) = \sum_{s=r}^{b_1} E_F\left(\mathrm{Prob}_X\left(F(X) \in V_{(F)s}\right)\right) = \sum_{s=r}^{b_1} \frac{sE_F\left(v_{(F)s}\right)}{b_1} \\
&= \sum_{s=r}^{b_1} \frac{sE_F\left(\sum_{z \in A_2} 1_{\{s\}}\left(\left|F^{-1}(z)\right|\right)\right)}{b_1} = \sum_{s=r}^{b_1} \frac{s \sum_{z \in A_2} \mathrm{Prob}\left(\left|F^{-1}(z)\right| = s\right)}{b_1} \\
&= \sum_{s=r}^{b_1} \frac{sb_2}{b_1}\binom{b_1}{s} p^s (1-p)^{b_1-s} = \sum_{s=r}^{b_1} \binom{b_1-1}{s-1} p^{s-1} (1-p)^{(b_1-1)-(s-1)} \\
&= \mathrm{Prob}\left(Y \geq r-1\right)
\end{aligned} \tag{4.111}$$

with $p = \frac{1}{b_2}$, and Y is a $B(b_1 - 1, p)$-distributed random variable. The third equation results from interchanging the order of integration (with regard to F and X).

580 The corresponding e_r-work factor (or more precisely, $e_{r(F)}$, averaged over all $f' \in \mathcal{F}_{A_1,A_2}$) equals

$$w_{e_r}\left(F(X)\right) = E_F\left(\left|\bigcup_{s=r}^{b_1} V_{(F)s}\right|\right) = \sum_{s=r}^{b_1} b_2 \binom{b_1}{s} p^s (1-p)^{b_1-s} = b_2\, \mathrm{Prob}\left(Y' \geq r\right) \tag{4.112}$$

where Y' denotes a $B(b_1, p)$-distributed random variable.

581 In particular, $\mathrm{E}(Y) = \frac{b_1-1}{b_2}$ and $\mathrm{Var}(Y) = \frac{b_1-1}{b_2}\left(1 - \frac{1}{b_2}\right)$. Similarly, $\mathrm{E}(Y') = \frac{b_1}{b_2}$ and $\mathrm{Var}(Y') = \frac{b_1}{b_2}\left(1 - \frac{1}{b_2}\right)$.

582 [Notation] In the remainder of this subsection, we assume

$$A_1 = \{0,1\}^n \quad \text{and} \quad A_2 = \{0,1\}^m \quad \text{with large } n \geq m. \tag{4.113}$$

Then $b_1 = 2^n$ and $b_2 = 2^m$. This case is relevant in the context of cryptographic post-processing. To simplify the notation we define

$$\gamma := 2^{n-m}\,. \tag{4.114}$$

Below, we distinguish the cases $\gamma \gg 1$ (data compression) and $\gamma = 1$.

583 Since n, m are assumed to be large, $(b_1 - 1)/b_1 = (2^n - 1)/2^n \approx 1$ and $1 - \frac{1}{b_2} = 1 - 2^{-m} \approx 1$. Using these approximations we obtain

$$E(Y) = E(Y') = \mathrm{Var}(Y) = \mathrm{Var}(Y') = \gamma\,. \tag{4.115}$$

In the remainder, we identify the distributions of Y and Y'.

Let U denote a uniformly distributed random variable on $A_2 = \{0,1\}^m$. The equations (4.116) and (4.117) provide useful relations 584

$$\text{Prob}\left(U \in \left(\bigcup_{s=r}^{b_1} V_{(F)s}\right)\right) = \frac{w_{e_r}(F(X))}{b_2} = \text{Prob}(Y' \geq r) = e_{r+1} \quad \text{, and thus} \tag{4.116}$$

$$w_{e_r}(F(X)) = w_{e_{r+1}}(U) \text{ and } w_{e_{r-1}}(F(X)) = w_{e_r}(U) \ . \tag{4.117}$$

Equations (4.116) and (4.117) imply (4.118). This term quantifies the *relative work factor defect* between U and $F(X)$ (on the elements of A_2 with pre-image size $\geq r$); 585

$$\frac{w_{e_r}(U) - w_{e_r}(F(X))}{w_{e_r}(U)} = \frac{w_{e_{r-1}}(F(X)) - w_{e_r}(F(X))}{w_{e_{r-1}}(F(X))} = \frac{e_r - e_{r+1}}{e_r} = \frac{\text{Prob}(Y = r-1)}{\text{Prob}(Y \geq r-1)} \tag{4.118}$$

[Case $\gamma \gg 1$] On average each $a_2 \in A_2$ has γ pre-images. Unless r is very small or very large compared to γ, the Central Limit Theorem (with correction factor '± 0.5', see [Geor15], Korollar (5.23)) equation (4.111) implies 586

$$e_r = 1 - \Phi\left(\frac{r - 1 - 0.5 - \gamma}{\sqrt{\gamma}}\right) = \Phi\left(\frac{\gamma - r + 1.5}{\sqrt{\gamma}}\right) \tag{4.119}$$

$$w_{e_r}(F(X)) = 2^m\left(1 - \Phi\left(\frac{r - 0.5 - \gamma}{\sqrt{\gamma}}\right)\right) = 2^m \Phi\left(\frac{\gamma - r + 0.5}{\sqrt{\gamma}}\right) \quad \text{and} \tag{4.120}$$

$$w_{e_r}(U) = 2^m \Phi\left(\frac{\gamma - r + 1.5}{\sqrt{\gamma}}\right) \ . \tag{4.121}$$

[Case $\gamma \gg 1$] From (4.120) and (4.121) we obtain the work factor defect between U and $F(X)$ (on the elements of A_2 with pre-image size $\geq r$) 587

$$w_{e_r}(U) - w_{e_r}(F(X)) = \Phi\left(\frac{\gamma - r + 1.5}{\sqrt{\gamma}}\right) - \Phi\left(\frac{\gamma - r + 0.5}{\sqrt{\gamma}}\right) \tag{4.122}$$

Substituting (4.120) and (4.121) into (4.118) yields the relative work factor defect between U and $F(X)$ (on the elements of A_2 with pre-image size $\geq r$)

$$\frac{w_{e_r}(U) - w_{e_r}(F(X))}{w_{e_r}(U)} = \frac{\Phi\left(\frac{\gamma - r + 1.5}{\sqrt{\gamma}}\right) - \Phi\left(\frac{\gamma - r + 0.5}{\sqrt{\gamma}}\right)}{\Phi\left(\frac{\gamma - r + 1.5}{\sqrt{\gamma}}\right)} \tag{4.123}$$

[Case $\gamma \gg 1$] Differentiating the relative work factor defect (4.123) yields 588

$$\frac{d}{dr}\frac{\Phi\left(\frac{\gamma-r+1.5}{\sqrt{\gamma}}\right) - \Phi\left(\frac{\gamma-r+0.5}{\sqrt{\gamma}}\right)}{\Phi\left(\frac{\gamma-r+1.5}{\sqrt{\gamma}}\right)} = \frac{d}{dr}\left(1 - \frac{\Phi\left(\frac{\gamma-r+0.5}{\sqrt{\gamma}}\right)}{\Phi\left(\frac{\gamma-r+1.5}{\sqrt{\gamma}}\right)}\right) =$$
$$\frac{\Phi\left(\frac{\gamma-r+1.5}{\sqrt{\gamma}}\right)\frac{1}{\sqrt{\gamma}}\phi\left(\frac{\gamma-r+0.5}{\sqrt{\gamma}}\right) - \frac{1}{\sqrt{\gamma}}\phi\left(\frac{\gamma-r+1.5}{\sqrt{\gamma}}\right)\Phi\left(\frac{\gamma-r+0.5}{\sqrt{\gamma}}\right)}{\left(\Phi\left(\frac{\gamma-r+1.5}{\sqrt{\gamma}}\right)\right)^2} \tag{4.124}$$

Here, $\phi(\cdot)$ denotes the density of a standard normal distribution. Applying L'Hopital's rule to (4.123) shows that the relative work factor defect converges to 1 as $r \to \infty$, which matches with intuition. It should be considered that at the same time, the work factor $w_{e_r}(U)$ converges to 0 (and, of course, for fixed γ the model does no longer fit as $r \to \infty$). For the work factor defect we obtain

$$w_{e_r}(U) - w_{e_r}(F(X)) \approx \phi\left(\frac{\gamma - r + 1}{\sqrt{\gamma}}\right). \tag{4.125}$$

Using the approximation (4.125) and applying the inequality (4.36) to the denominator of the right-hand side of (4.123) (with $x = (r - \gamma - 1.5)/\sqrt{\gamma}$) yields the approximation

$$\frac{w_{e_r}(U) - w_{e_r}(F(X))}{w_{e_r}(U)} \approx \frac{\phi\left(\frac{\gamma-r+1}{\sqrt{\gamma}}\right)}{\frac{\sqrt{\gamma}}{r-\gamma-1.5}\phi\left(\frac{r-\gamma-1.5}{\sqrt{\gamma}}\right)} = \frac{r-\gamma-1.5}{\sqrt{\gamma}} e^{\frac{-0.5}{\gamma}\left((\gamma-r+1)^2-(\gamma-r+1.5)^2\right)} =$$
$$\frac{r-\gamma-1.5}{\sqrt{\gamma}} e^{\frac{-0.5}{\gamma}(-(\gamma-r+1)-0.25)} = \frac{r-\gamma-1.5}{\sqrt{\gamma}} e^{\frac{-0.5}{\gamma}(r-\gamma-1.25)} \quad \text{for } r > \gamma + 1.5 \tag{4.126}$$

589 [Case $\gamma \gg 1$] Assume that $e_r \leq \alpha < e_{r-1}$. Linear interpolation in r yields an approximation of the work factor $w_\alpha(F(X))$. More precisely,

$$w_\alpha(F(X)) = b_2 \Phi\left(\frac{\gamma - r_\alpha + 0.5}{\sqrt{\gamma}}\right) \text{ with } r_\alpha = r - 1 + \frac{\alpha - e_{r-1}}{e_r - e_{r-1}} \tag{4.127}$$

while trivially $w_\alpha(U) = b_2\alpha = 2^{-m}\alpha$.

590 [Case $\gamma \gg 1$] Equation (4.123) is the equivalent to (4.118) for arbitrary success probabilities α

$$\frac{w_\alpha(U) - w_\alpha(F(X))}{w_{e_r}(U)} \approx \frac{\alpha - \Phi\left(\frac{\gamma - r_\alpha + 0.5}{\sqrt{\gamma}}\right)}{\alpha}. \tag{4.128}$$

591 [Case $\gamma = 1$, i.e., $n = m$] This corresponds to a cryptographic post-processing for which the input rate equals the output rate. In this case the random variables Y and Y' may be viewed as a Poisson distributed P_τ with parameter $\tau = 1$. In particular, (4.118) simplifies to

$$\frac{w_{e_r}(U) - w_{e_r}(F(X))}{w_{e_r}(U)} \frac{P_1(r-1)}{\sum_{s=r-1}^{\infty} P_1(s)} = \frac{\frac{1}{(r-1)!}}{\sum_{s=r-1}^{\infty} \frac{1}{s!}} \tag{4.129}$$

592 Above, we computed the (average) work factor of a random mapping F, and we compared it to the work factor for the uniform distribution U on A_2. However, in cryptographic applications, usually a particular mapping $f \in \mathcal{F}_{A_1,A_2}$ is selected (e.g., a dedicated hash function), which is permanently applied. This is in particular relevant for cryptographic post-processing algorithms without memory. An important question in this context is how 'typical' such a mapping is with regard to the work factor.

593 The most extreme case clearly is when the selected mapping f maps all $a_1 \in A_1$ onto the same image. Then the entropy of $f(X)$ is 0, and its work factor is 1 for any success probability α.

Of course, it is extremely unlikely that a randomly selected mapping is constant. In particular, dedicated cryptographic algorithms are definitely far away from these extremal cases. The question yet remains how typical the average work factor $f(X)$ of a randomly selected mapping $f \in \mathcal{F}_{A_1,A_2}$ is. In the following we develop a formula for the variance of the work factor of $F(X)$.

With the same strategy as in (4.112), we compute the second moment of the work factor 594
$w_{e_r}(F(X))$.

$$
\begin{aligned}
&E_F\left(w_{e_r}(F(X))^2\right) = E_F\left(\left|\bigcup_{s=r}^{b_1} V_{(F)s}\right|^2\right) = E_F\left(\sum_{s_1,s_2\geq r} \left|V_{(F)s_1}\right| \cdot \left|V_{(F)s_2}\right|\right)\\
&\sum_{s_1,s_2\geq r} E_F\left(\sum_{z_1\in A_2} 1_{\{s_1\}}\left(\left|F^{-1}(z_1)\right|\right) \cdot \sum_{z_2\in A_2} 1_{\{s_2\}}\left(\left|F^{-1}(z_2)\right|\right)\right) =\\
&\sum_{s_1,s_2\geq r} E_F\left(\sum_{z_1,z_2\in A_2} 1_{\{s_1,s_2\}}\left(\left|F^{-1}(z_1)\right|, \left|F^{-1}(z_2)\right|\right)\right) =\\
&\sum_{s_1,s_2\geq r}\sum_{z_1,z_2\in A_2} \text{Prob}\left(\left|F^{-1}(z_1)\right| = s_1, \left|F^{-1}(z_2)\right| = s_2\right) =\\
&\sum_{s_1,s_2\geq r}\sum_{z_1,z_2\in A_2} \text{Prob}\left(\left|F^{-1}(z_1)\right| = s_1\right) \cdot \text{Prob}\left(\left|F^{-1}(z_2)\right| = s_2 :\ \left|F^{-1}(z_1)\right| = s_1\right)
\end{aligned} \tag{4.130}
$$

Similarly as in (4.112), we obtain

$$
\text{Prob}\left(\left|F^{-1}(z_1)\right| = s_1\right) = \binom{b_1}{s_1} p^{s_1}(1-p)^{b_1-s_1} = \text{Prob}(Y' = s_1)\,. \tag{4.131}
$$

The conditional probabilities require a case discrimination. It is

$$
\begin{aligned}
&\text{Prob}\left(\left|F^{-1}(z_2)\right| = s_2 :\ \left|F^{-1}(z_1)\right| = s_1\right) = \binom{b_1-s_1}{s_2} p^{*\,s_2}(1-p^*)^{b_1-s_1-s_2} =\\
&\text{Prob}(Y'_{s_1} = s_2) \quad \text{with } p^* = \frac{1}{b_2-1} \quad \text{and } Y'_{s_1} \sim B(b_1-s_1, p^*) \ \text{ if } z_1 \neq z_2.
\end{aligned} \tag{4.132}
$$

and

$$
\text{Prob}\left(\left|F^{-1}(z_2)\right| = s_2 :\ \left|F^{-1}(z_1)\right| = s_1\right) = 1_{\{s_1\}}(s_2) \quad \text{if } z_1 = z_2. \tag{4.133}
$$

Putting the pieces together and substituting (4.131), (4.132), and (4.133) into (4.130) yields

$$
\begin{aligned}
&E_F\left(w_{e_r}(F(X))^2\right) =\\
&\sum_{s_1,s_2\geq r}\left(\sum_{z_1,z_2\in A_2} \text{Prob}\left(Y' = s_1\right)\cdot \text{Prob}\left(Y'_{s_1} = s_2\right) + \text{Prob}\left(Y' = s_1\right)\sum_{z\in A_2}\left(1_{\{s_1\}}(s_2) - \text{Prob}\left(Y'_{s_1} = s_2\right)\right)\right) =\\
&\sum_{s_1,s_2\geq r} \text{Prob}\left(Y' = s_1\right)\left(\left(b_2^2 - b_2\right)\text{Prob}\left(Y'_{s_1} = s_2\right) + b_2 \cdot 1_{\{s_1\}}(s_2)\right) =\\
&\sum_{s_1=r}^{b_1} \text{Prob}\left(Y' = s_1\right)\left(b_2^2\left(1 - b_2^{-1}\right)\text{Prob}\left(Y'_{s_1} \geq r\right) + b_2 1_{\{\leq b_2/2\}}(s_1)\right)\,.
\end{aligned} \tag{4.134}
$$

If $s_1 > b_2/2$ then in (4.133) $s_2 < s_1$, which verifies the indicator function $1_{\{\leq b_2/2\}}(s_1)$.

595 We first note that $E\left(w_{e_r}(F(X)\right) = w_{e_r}(F(X))$ and

$$\left(w_{e_r}(F(X))\right)^2 = b_2^2 \operatorname{Prob}(Y' \geq r)^2 = b_2^2 \sum_{s_1 \geq r} \operatorname{Prob}(Y' = s_1) \operatorname{Prob}(Y' \geq r)\,. \tag{4.135}$$

Hence,

$$\operatorname{Var}_F\left(w_{e_r}(F(X))\right) = E_F\left(w_{e_r}(F(X))^2\right) - \left(w_{e_r}(F(X))\right)^2 =$$
$$\sum_{s_1=r}^{b_1} \operatorname{Prob}\left(Y' = s_1\right)\left(b_2^2\left[\left(1-b_2^{-1}\right)\operatorname{Prob}\left(Y'_{s_1} \geq r\right) - \operatorname{Prob}\left(Y' \geq r\right)\right] + b_2 1_{\{\leq b_2/2\}}(s_1)\right) \tag{4.136}$$

596 We try to simplify (4.136). Recall that $b_1 = 2^n$ and $b_2 = 2^m$ are large. Since $E(Y') = b_1/b_2 \ll b_1$, the probability $\operatorname{Prob}(Y' > b_2/2)$ is essentially zero. Omitting the indicator function $1_{\{\leq b_2/2\}}(s_1)$ thus does not significantly change the value of (4.136). Similarly, $1 - b_2^{-1} \approx 1$. For $c \geq 1$ the Chernoff inequality implies $\operatorname{Prob}(Y' \geq (1+c)E(Y')) \leq e^{-(c/3)(b_1/b_2)}$. This means that for cryptographic purposes (estimating work factors for reasonable success probabilities) this tail probability can be made sufficiently small for $c \in O(1)$. For $s_1 \leq (1+O(1))E(Y')$ we have $\operatorname{Prob}(Y' \geq r) \approx \operatorname{Prob}(Y'_{s_1} \geq r)$. Altogether, this justifies replacing the bracket $[\cdot]$ in (4.136) by 0. Putting the pieces together (4.136) simplifies to

$$\operatorname{Var}_F\left(w_{e_r}(F(X))\right) \approx b_2 \operatorname{Prob}\left(Y' \geq r\right) = w_{e_r}\left(F\left(X\right)\right)\,. \tag{4.137}$$

The standard deviation of the work factor $w_{e_r}(F(X))$ is $\approx \sqrt{w_{e_r}(F(X))}$. For cryptographically meaningful success probabilities, the standard deviation is small compared to the work factor. Hence, we may assume that a randomly selected mapping in $\mathcal{F}_{A_1,A_2}$ is 'typical' with regard to the work factors.

597 It is $\operatorname{Prob}_X\left(\left|F^{-1}(X)\right| = r\right) = \operatorname{Prob}_X\left(\left|F^{-1}(X)\right| \geq r\right) - \operatorname{Prob}_X\left(\left|F^{-1}(X)\right| \geq r+1\right)$. By (4.111) and (4.112)

$$\operatorname{Prob}_X\left(\left|F^{-1}(X)\right| = r\right) = \operatorname{Prob}\left(Y = r-1\right) \quad \text{and similarly} \tag{4.138}$$
$$E_F\left(\left|\left\{y \in A_2 \mid \left|F^{-1}(y)\right| = r\right\}\right|\right) = E_F\left(\left|\bigcup_{s=r}^{r} V_{(F)s}\right|\right) = b_2 \operatorname{Prob}\left(Y' = r\right) \tag{4.139}$$

where the random variables Y and Y' are binomially $B(b_1 - 1, \frac{1}{b_2})$-distributed and $B(b_1, \frac{1}{b_2})$-distributed, respectively.

598 **Impact on the Shannon entropy**
After this excursion to the work factor, we return to our main goal, the impact of randomly selected mappings on entropy. Our analysis begins with Shannon entropy. As before, $\gamma = 2^{n-m}$ with 'large' parameters n and m, and the random variable X is assumed to be uniformly distributed on A_1.

600 [Shannon entropy] In the context of AIS 20 and AIS 31, we are interested in the entropy $H(f(X))$ for given mappings f. For one-way functions (as SHA-256) it is infeasible to determine this value exactly. Instead, we compute the expected entropy value when the mapping f is selected randomly; cf. pars. 568 to 572; basic considerations were already explained, e.g., in [Schi09b],

Example 3.11. We view f as a realization of the random variable F. Straight-forward considerations yield

$$\begin{aligned} E\left(H\left(F(X)\right)\right) &= -\sum_{f\in\mathcal{F}_{A_1,A_2}} \frac{1}{|\mathcal{F}_{A_1,A_2}|} \sum_{a_2\in A_2} \left(\sum_{a\in f^{-1}(a_2)} \mathrm{Prob}(X=a)\right) \cdot \log_2\left(\sum_{a\in f^{-1}(a_2)} \mathrm{Prob}(X=a)\right) \\ &= -\sum_{f\in\mathcal{F}_{A_1,A_2}} \frac{1}{|\mathcal{F}_{A_1,A_2}|} \sum_{r=1}^{b_1} \left|\{y\in A_2 \mid |f^{-1}(y)|=r\}\right| \cdot \frac{r}{b_1} \cdot \log_2\left(\frac{r}{b_1}\right) \\ &= -\sum_{r=1}^{b_1} E_F\left(\left|\{y\in A_2 \mid |f^{-1}(y)|=r\}\right|\right) \cdot \frac{r}{b_1} \cdot \log_2\left(\frac{r}{b_1}\right) \\ &= -\sum_{r=1}^{b_1} b_2\,\mathrm{Prob}\left(Y'=r\right) \cdot \frac{r}{b_1} \cdot \log_2\left(\frac{r}{b_1}\right) \end{aligned} \tag{4.140}$$

To be precise, $E\left(H\left(F(X)\right)\right)$ denotes the average entropy with regard to F, i.e., $E_F\left(H\left(F(X)\right)\right)$. The first line in (4.140) provides the formula for the general case where X has arbitrary distribution on A_1. For the special case where X is uniformly distributed on A_1, it suffices to consider the size of the pre-images. This simplifies the computations significantly (second line of (4.140)). Since $b_1 = b_2\gamma$ we obtain $\log_2(\frac{r}{b_1}) = \log_2(\frac{r}{\gamma}) - m$, and $E(Y') = \frac{b_1}{b_2}$ (4.140) implies

$$\begin{aligned} E\left(H\left(F(X)\right)\right) &= -b_2\sum_{j=1}^{b_1} \mathrm{Prob}\left(Y'=r\right)\left(-m\frac{r}{b_1} + \frac{r}{b_2\gamma}\log_2\left(\frac{r}{\gamma}\right)\right) = \\ & \frac{b_2}{b_1} mE(Y') - \frac{b_2}{b_2} E\left(\frac{Y'}{\gamma}\log_2\left(\frac{Y'}{\gamma}\right)\right) = m - E\left(\frac{Y'}{\gamma}\log_2\left(\frac{Y'}{\gamma}\right)\right) \end{aligned} \tag{4.141}$$

In (4.140) the random variable X is assumed to be uniformly distributed on A_1. This simplifies the computation as it suffices to count the numbers of pre-images. Of course, the expectation $E\left(H\left(F(X)\right)\right)$ exists for non-uniformly distributed random variables X, too, but the computations become significantly more complicated. 601

[Shannon entropy, $\gamma = 1$] Here $b_1 = b_2 = 2^n$ and $\gamma = 1$. As pointed out in par. 591 the random variable Y' may be viewed as Poisson distributed with parameter 1. Then (4.141) equals 602

$$E\left(H\left(F(X)\right)\right) = n - e^{-1}\sum_{r=0}^{\infty} \frac{1}{r!} r\log_2(r)\,, \tag{4.142}$$

with $0\log_2(0) := 0$ as usual. The second term of (4.142) quantifies the average entropy defect that occurs when a random mapping is applied to a uniformly distributed random variable X. Numerical computations show that

$$e^{-1}\sum_{r=0}^{\infty} \frac{1}{r!} r\log_2(r) \approx e^{-1}\sum_{r=1}^{10} \frac{\log_2(r)}{(r-1)!} \approx 0.827 \tag{4.143}$$

This means that the average Shannon entropy defect per random bit is $0.827/m$. If $n = m = 256$, for example, the average Shannon entropy defect per bit is ≈ 0.003.

603

[Shannon entropy, $\gamma \gg 1$] Since n and m are assumed to be large, Y' can be approximated by a normal distribution $N(\gamma, \gamma)$ with $\gamma = 2^{n-m}$, cf. par. 583. Then

$$\operatorname{Prob}(Y' = r) \approx \frac{1}{\sqrt{2\pi\gamma}} e^{-\frac{(r-\gamma)^2}{2\gamma}}, \tag{4.144}$$

and (4.141) can be approximated by

$$E(H(F(X))) = m - \frac{1}{\sqrt{2\pi\gamma}\ln(2)} \int_{0.5}^{\infty} e^{-\frac{(r-\gamma)^2}{2\gamma}} \cdot \frac{r}{\gamma} \ln\left(\frac{r}{\gamma}\right) dr \tag{4.145}$$

604 [Shannon entropy, $\gamma \gg 1$] Because the exact calculation of the right-hand integral in (4.145) appears to be rather difficult, we apply Jensen's inequality to the expectation in (4.141). Recall that $Y' \sim B(2^n, p)$-distributed with $p = 2^{-m}$ (cf. par. 580), and furthermore $E(Y') = \gamma$ and $E({Y'}^2) = \operatorname{Var}(Y') + E^2(Y') = \gamma(1-p) + \gamma^2$ (par. 583). By this,

$$E\left(\frac{Y'}{\gamma}\log_2\left(\frac{Y'}{\gamma}\right)\right) = \sum_{j=0}^{b_1} \frac{j\operatorname{Prob}(Y'=j)}{\gamma}\log_2\left(\frac{j}{\gamma}\right) = \sum_{j=0}^{b_1} q_j \log_2\left(\frac{j}{\gamma}\right) \tag{4.146}$$

where $q_j := j\operatorname{Prob}(Y'=j)/\gamma$. Since $\gamma = E(Y')$, it is $q_0, \ldots, q_{b_1} \geq 0$ and $q_0 + \cdots + q_{b_1} = 1$. Furthermore, since $x \mapsto \log(x/\gamma)$ is concave on $\mathbb{R}_+$ Jensen's inequality implies

$$\begin{aligned} \sum_{j=1}^{b_1} q_j \log_2\left(\frac{j}{\gamma}\right) \quad &\leq \quad \log_2\left(\sum_{j=1}^{b_1} q_j \frac{j}{\gamma}\right) = \log_2\left(\sum_{j=1}^{b_1}\frac{\operatorname{Prob}(Y'=j)j^2}{\gamma^2}\right) = \log_2\left(\frac{E\left({Y'}^2\right)}{\gamma^2}\right) = \\ &\log_2\left(\frac{\gamma(1-p)+\gamma^2}{\gamma^2}\right) < \log_2\left(1+\frac{1}{\gamma}\right) \leq \frac{1}{\log(2)\gamma}. \end{aligned} \tag{4.147}$$

(Since $q_0 = 0$ and by convention $0 \cdot \log_2(0) = 0$ the sums in (4.147) start with index $j = 1$.) Equation(4.147) matches with our intuition that the entropy defect becomes negligible if the compression difference $n - m$ and thus $\gamma = 2^{n-m}$ increases. Altogether,

$$E(H(F(X))) \geq m - \frac{1}{\ln(4)\gamma}. \tag{4.148}$$

Dividing (4.148) by m provides a lower bound for the average Shannon entropy per internal random number bit:

$$\frac{E(H(F(X)))}{m} \geq 1 - \frac{1}{\ln(4)\gamma m}. \tag{4.149}$$

In particular, the average Shannon entropy per internal random number bit decreases exponentially in $n - m$.
Note: Of course, Jensen's inequality applies to the case $\gamma = 1$, too, but there (4.143) is more suitable as it quantifies the entropy defect more precisely than (4.147).

605 **Impact on min-entropy**
Finally, we consider the impact of randomly selected mappings on the min-entropy. As before, $\gamma = 2^{n-m}$ and the random variable X is assumed to be uniformly distributed on A_1. The following results have the highest relevance for functionality classes PTG.3 and NTG.1. This in particular applies to the results on non-uniformly distributed random variables.

606

[Min-entropy] Thus, for each mapping $f\colon A_1 \to A_2$ we have $\mathrm{Prob}(f(X) = y') = \frac{|f^{-1}(y')|}{2^n}$ for each $y' \in A_2$, and furthermore,

$$\left(\max_{y' \in A_2} \left|f^{-1}(y')\right| > \gamma(1+\tau)\right) \Longleftrightarrow$$
$$\left(H_{min}(f(X)) < -\log_2\left(\frac{\gamma(1+\tau)}{2^n}\right) = m - \log_2(1+\tau)\right) \quad \text{for } \tau > 0 \tag{4.150}$$

Below, we analyze the distribution of the size of the largest pre-image if the mapping f is selected randomly.

Let $f \in \mathcal{F}_{A_1,A_2}$ be fixed. For the remainder of this section the term $M(f) := \max_{y' \in A_2} \left|f^{-1}(y')\right|$ denotes the maximal pre-image size for f (a.k.a. maximal occupancy). In pars. 608 to 610 we state three known limiting distributions of $M(F)$; see [KoSC78] for details. Interestingly, the limiting behavior depends on the scale parameter 607

$$\frac{|A_1|}{|A_2|\log(|A_2|)} = \frac{2^n}{2^m \log(2^m)} = \frac{\gamma}{m\log(2)}\,. \tag{4.151}$$

In the pars. 608 to 610 we assume that $n, m \to \infty$.

[Case $\frac{\gamma}{m\log(2)} \longrightarrow 0$] ([KoSC78], Sect. II 6, Theorem 1, p. 96) This case covers data expansion ($n < m$), the case $\gamma = 1$, and small compression $\gamma > 1$. Interestingly, the asymptotic distribution is concentrated at most two values. More precisely, assume that $r = r(m)$ is chosen so that $r > \gamma$ and $2^m \nu_{P_\gamma}(r)$ converges to $\lambda > 0$. Here, ν_{P_γ} denotes the Poisson distribution with parameter γ. Then 608

$$\mathrm{Prob}(M(F) = r-1) \quad \longrightarrow e^{-\lambda} \tag{4.152}$$
$$\mathrm{Prob}(M(F) = r) \quad \longrightarrow 1 - e^{-\lambda} \tag{4.153}$$

[Case $\frac{\gamma}{m\log(2)} \longrightarrow x > 0$] ([KoSC78], Sect. II 6, Theorem 2, p. 96) In this case the asymptotic distribution is discrete with infinite range. Assume that $r = r(m)$ is chosen so that $r > \gamma$ and $2^m \nu_{Poi,\gamma}(r)$ converges to $\lambda > 0$. Then 609

$$\mathrm{Prob}(M(F) \le r + j) \longrightarrow_{r\to\infty} \exp\left(-\frac{\lambda\eta^{j+1}}{1-\eta}\right) \quad \text{for } j \in \mathbb{Z} \tag{4.154}$$

where η is the zero of the equation

$$\eta + x(\log(\eta) - \eta + 1) = 0 \tag{4.155}$$

in $(0, 1)$.
Note: In (4.155) x denotes the value specified in the Case condition.

[Case $\frac{\gamma}{m\log(2)} \longrightarrow \infty$] ([KoSC78], Sect. II 6, Theorem 3, p. 96) This is the most relevant case for us since it treats the case of large compressions. Here, the asymptotic distribution is continuous. It is an extreme value distribution of the double exponential type, often named 'Gumbel distribution'. 610

More precisely,

$$\mathrm{Prob}\left(\frac{M(F)-\gamma-\gamma\, u\left(\frac{1}{\gamma}\left(\log(2^m)-\frac{\log\log(2^m)}{2}\right)\right)}{\sqrt{\frac{\gamma}{2}\log(2^m)}}+\frac{1}{2}\log(4\pi)\le z\right)=$$

$$\mathrm{Prob}\left(\frac{M(F)-\gamma-\gamma\, u\left(\frac{1}{\gamma}\left(m\log(2)-\frac{\log(m\log(2))}{2}\right)\right)}{\sqrt{\frac{\gamma}{2}\, m\log(2)}}+\frac{1}{2}\log(4\pi)\le z\right)\longrightarrow e^{-e^{-z}} \quad (4.156)$$

The function $u\colon (0,\infty)\to(0,\infty)$ is implicitly defined by

$$-u(t)+(1+u(t))\log(1+u(t))=t\,. \quad (4.157)$$

For small $t>0$ the function $u(t)$ has the expansion

$$u(t)=\sqrt{2t}+\frac{1}{3}t-\frac{\sqrt{2}}{36}t^{3/2}+\dots \quad (4.158)$$

In (4.156) it is $t=\frac{1}{\gamma}\left(m\log(2)-\frac{\log(m\log(2))}{2}\right)$. The expansion (4.158) applies to large compression rates γ.

611 [$\gamma=1$] The Case $\frac{\gamma}{m\log(2)}\longrightarrow 0$ from par. 608 applies to $\gamma=$ (no data compression). On the other hand, for a randomly selected element $a_2\in A_2=\{0,1\}^m$ the pre-image size $|f^{-1}(a_2)|$ is Poisson distributed with parameter $\tau=1$. It should be easier to apply this result.

612 [numerical example, $\gamma\gg 1$] The largest pre-image determines the min-entropy. Formula (4.156) in par. 610 gives an upper bound for the maximal occupancy $M(F)$, which in turn provides a lower bound for the min-entropy. In Tab. 4 we used the approximation $u(t)\approx\sqrt{2t}+\frac{1}{3}t-\frac{\sqrt{2}}{36}t^{3/2}$. (Cutting off $u(t)$ after the first term, for example, yields numerical values that differ a little.) The expansion of $u(t)$ until the third term should be enough by far because its argument in (4.156) is in the order of $1/\gamma$. Furthermore, the min-entropy bound depends on the choice of z. (Of course, in (4.156) probability 1 cannot be achieved because constant mappings map A_1 onto a single element.) Solving the term Prob$(\dots)$ in (4.156) for $M(F)$ gives

$$M(F)\le\sqrt{\frac{\gamma}{2}\, m\log(2)}\left(z-\frac{1}{2}\log(4\pi)\right)+\gamma+\gamma\, u\left(\frac{1}{\gamma}\left(m\log(2)+\frac{\log(m\log(2))}{2}\right)\right) \quad (4.159)$$

Then

$$M_z^*(F):=\sqrt{\frac{\gamma}{2}\, m\log(2)}\left(z-\frac{1}{2}\log(4\pi)\right)+\gamma+\gamma\, u\left(\frac{1}{\gamma}\left(m\log(2)+\frac{\log(m\log(2))}{2}\right)\right) \quad (4.160)$$

defines an upper bound for the maximal pre-image size that is not exceeded with probability $e^{-e^{-z}}$. Therefrom, by (4.150) with $M_z^*(F)=\gamma(1+\tau)$, we conclude that the min-entropy per m-bit output block does not undercut

$$-\log_2\left(\frac{M_z^*(F)}{2^n}\right)=m-\log_2\left(\frac{M_z^*(F)}{\gamma}\right)\,. \quad (4.161)$$

with probability $e^{-e^{-z}}$. Consequently,

$$h_{min,(n,m,z)}:=1-\frac{\log_2\left(\frac{M_z^*(F)}{\gamma}\right)}{m} \quad (4.162)$$

the *min-entropy defect per output bit*, is exceeded only with a probability of $1 - e^{-e^{-z}}$. (As before, X and F are uniformly distributed on $\{0,1\}^n$ and on $\mathcal{F}_{\{0,1\}^n,\{0,1\}^m}$, respectively.)

613 Table 4 provides the *min-entropy defect per output bit* (4.162) for several parameter sets (n, m, z). There, z_τ is selected such that the right-hand side in (4.156) equals $1 - 2^{-\tau}$. Entry 3 in row 2, for example, says that the min-entropy defect exceeds $2^{-64.17}$ only with probability $< 2^{-16}$. The numerical values indicate that the min-entropy defect per output bit essentially depends on the difference $n - m$.

Table 4: Case $\frac{\gamma}{m \log(2)} \longrightarrow \infty$ (cf. par. 610): Min-entropy defect per output bit for different parameters, $h_{min,(n,m,z)}$, computed with formula (4.162)

(n,m)	z_8	z_{12}	z_{16}
$(320, 256)$	$2^{-33.59}$	$2^{-33.05}$	$2^{-32.67}$
$(256, 128)$	$2^{-65.09}$	$2^{-64.56}$	$2^{-64.17}$
$(192, 128)$	$2^{-33.09}$	$2^{-32.56}$	$2^{-32.17}$

614 [non-uniformly distributed pre-images] So far, in this section we have assumed that X is uniformly distributed on $\{0,1\}^n$. For PTG.3-compliant PTRNGs the cryptographic post-processing algorithm may be modeled as a randomly selected mapping, and an output sequence of length n of a PTG.2-compliant PTRNGs (here: intermediate random numbers) may be interpreted as a realization of a random vector X; cf. par. 566 to 571. Of course, we cannot assume that X is uniformly distributed on $\{0,1\}^n$ because the intermediate random numbers usually are (to some degree) biased and correlated. One might expect that pre-images with small probabilities and those with large probabilities cancel each other out to a large extent. The question is, however, to which degree deviations from the uniform distribution influence the above results. In the following we focus on min-entropy.

615 [non-uniformly distributed pre-images] The ideal situation, of course, would be if the random variable X would (for each $a_2 \in A_2$) assume a value in its pre-image with probability 2^{-m}. Two things usually prevent this: the pre-images $f^{-1}(\cdot)$ of randomly selected mappings do not have identical size, and X is not uniformly distributed on A_1. Principally, one might try to adapt the strategies for the uniform distribution but then the computations became too complicated; it would no longer suffice to count the number of pre-images. Below, we show that the deviations of X from the uniform distribution can be compensated by a moderate increase of the input length from n to n^*.

616 [non-uniformly distributed pre-images] As above, we assume that a vector of intermediate random numbers (generated by a PTG.2-compliant PTRNG) is a realization of a random variable X'. Let

$$n^* = \min\{n' \geq n \mid H_{min}(X') \geq n\} \quad \text{and } \Delta n := n^* - n \tag{4.163}$$

In the following we assume that $n > m$ and that (n, m) are sufficiently large such that the asymptotic formula (4.156) applies. Furthermore, $A_1^* = \{0,1\}^{n^*}$, and as before $A_1 = \{0,1\}^n$ and $A_2 = \{0,1\}^m$.
In the following we compare the case where an n^*-bit random vector X' is mapped (by a randomly selected mapping $\in \mathcal{F}_{A_1^*,A_2}$) to m-bit output vectors with the case where uniformly distributed

n-bit random vector X is mapped (by a randomly selected mapping $\in \mathcal{F}_{A_1,A_2}$) to m-bit output vectors. The second case has already been studied above.

617 [non-uniformly distributed pre-images, Example] Assume that the output of the PTG.2-compliant PTRNG (intermediate random numbers) can be modeled by a sequence of binary-valued iid random variables $B_1, B_2, \ldots$ with $\text{Prob}(B_j = 1) = 0.5 + \epsilon$ for some $\epsilon \geq 0$. Then $H_{min}(B_j) = -\log_2(0.5 + \epsilon)$, and $n^* = \left\lceil \frac{n}{-\log_2(0.5+\epsilon)} \right\rceil$.
Numerical example: $(n, \epsilon, n^*, \Delta n) = (256, 0.01, 264, 8)$, $(256, 0.007, 262, 6)$, $(320, 0.007, 327, 7)$.

618 [non-uniformly distributed pre-images, $\gamma \gg 1$] Let $\gamma^* = 2^{n^*-m}$ and $a_2 \in A_2$. We assume that $f \in \mathcal{F}_{A_1^*,A_2}$ is selected randomly. The pre-image size $v := |f^{-1}(a_2)|$ can be interpreted as a realization of a random variable $V_{a_2} \sim B(2^{n^*}, 2^{-m}) \approx N(2^{n^*-m}, 2^{n^*} \cdot 2^{-m}(1-2^{-m})) \approx N(\gamma^*, \gamma^*)$ (CLT). We may assume that the pre-image $f^{-1}(a_2) = \{a'_1, \ldots, a'_v\}$ is a randomly selected subset of A_1 of size v. Hence, we may assume that $a'_j \in f^{-1}(\{a_2\})$ has been selected in A_1 with probability 2^{-n^*}. For $j = 1, \ldots, v$ we define the random variable $T_j := \text{Prob}(X' = a'_j)$. Then

$$E(T_j) = \sum_{a \in A_1} 2^{-n^*} \text{Prob}(X' = a) = 2^{-n^*} . \tag{4.164}$$

and, similarly,

$$E\left(T_j{}^2\right) = \sum_{a \in A_1} 2^{-n^*} \text{Prob}(X' = a)^2 = 2^{-n^*} 2^{-H_2(X')} . \tag{4.165}$$

Recall that $H_2(X')$ denotes the collision entropy of X'. Since $V_{a_2} \sim N(\gamma^*, \gamma^*)$, it is $|v| = \gamma^* + O(\sqrt{\gamma^*}) \ll 2^{n*}$. This justifies the assumption that the random variables $T_1, T_2, \ldots, T_v$ are iid.

619 [non-uniformly distributed pre-images, $\gamma \gg 1$] We set $T_{a_2} := \text{Prob}(X' \in f^{-1}(a_2))$ ($= \text{Prob}(X' \in f^{-1}(a_2))$), or equivalently, $T_{a_2} = T_1 + \cdots + T_v$. Wald's Theorem implies

$$E\left(T_{a_2}\right) = E\left(\sum_{j=1}^{V_{a_2}} T_j\right) = E(V_{a_2})E(T_j) = \gamma^* 2^{-n^*} = 2^{-m} . \tag{4.166}$$

Similarly, although with greater effort, (4.167) follows. Concerning the random variable V_{a_2} we, 'switch' between the normal distribution and the discreteness of possible pre-image sizes (binomial distribution). From (4.165) we obtain

$$\begin{aligned} E\left(T_{a_2}^2\right) &= E\left(\left(\sum_{j=1}^{V_{a_2}} T_j\right)^2\right) = \sum_{v=0}^{2^{n^*}} \mathrm{Prob}(V_{a_2}=v) E\left(T_{a_2}^2 \mid V_{a_2}=v\right) = \\ & \sum_{v=0}^{2^{n^*}} \mathrm{Prob}(V_{a_2}=v) E\left((T_1+\cdots+T_v)^2\right) = \\ & \sum_{v=0}^{2^{n^*}} \mathrm{Prob}(V_{a_2}=v)\left(vE\left(T_1^2\right)+(v^2-v)\left(E(T_1)\right)^2\right) = \\ & \sum_{v=0}^{2^{n^*}} \mathrm{Prob}(V_{a_2}=v)\left(v\cdot 2^{-n^*}2^{-H_2(X')}+(v^2-v)2^{-2n^*}\right) = \\ & E(V_{a_2})\cdot 2^{-n^*}2^{-H_2(X')}+\left(E(V_{a_2}^2)-E(V_{a_2})\right)2^{-2n^*} = \\ & \gamma^*\cdot 2^{-n^*}2^{-H_2(X')}+\gamma^{*2}2^{-2n^*} = 2^{-m}\cdot 2^{-H_2(X')}+2^{-2m} \end{aligned} \tag{4.167}$$

Finally, from (4.166) and (4.167) we obtain

$$\mathrm{Var}\left(T_{a_2}\right) = \mathrm{Var}\left(\sum_{j=1}^{V_{a_2}} T_j\right) = 2^{-m}\cdot 2^{-H_2(X')}+2^{-2m}-2^{-2m} = 2^{-m}\cdot 2^{-H_2(X')} \quad \text{and} \tag{4.168}$$

$$\begin{aligned} \sigma_{T_{a_2}} &= \sqrt{\mathrm{Var}\left(\sum_{j=1}^{V_{a_2}} T_j\right)} = \sqrt{2^{-m}\cdot 2^{-H_2(X')}} = \\ & 2^{-0.5m}\cdot 2^{-0.5\cdot H_2(X')} = 2^{-0.5(n+m)}\cdot 2^{-0.5(H_2(X')-n)} \end{aligned} \tag{4.169}$$

Note: If $X' = (B_1,\ldots,B_{n^*})$ with iid $B(1, 0.5+\epsilon)$-distributed random variables $B_1,\ldots,B_{n^*}$ (cf. par. 617) then $H_2(X') = -n^*\log_2(0.5+2\epsilon^2)$ and $H_{min}(X') = -n^*\log_2(0.5+\epsilon)$.

[non-uniformly distributed pre-images, $\gamma \gg 1$, comparison to the uniform case] If the PTG.2-compliant PTRNG would generate iid unbiased intermediate random numbers, the random vector X' would be uniformly distributed so that $n^* = n$, $H_2(X') = H_{min}(X') = n$, and (4.169) simplifies to $\sigma_{T_{a_2}} = 2^{-0.5(n+m)}$. As in par. 616 we assume that the random variable X is uniformly distributed on $\{0,1\}^n$ while X' assumes values in $\{0,1\}^{n^*}$. Then $E(f(X) = a_2) = 2^{-m} = E(f'(X') = a_2)$ if $a_2 \in A_2$ and the mappings $f \in \mathcal{F}_{A_1,A_2}$ and $f' \in \mathcal{F}_{A_1^*,A_2}$ are randomly selected. The factor $2^{-0.5(H_2(X')-n)}$ quantifies the ratio of the average standard deviations of the probabilities $\mathrm{Prob}(f(X) = a_2)$ and $\mathrm{Prob}(f(X' = a_2)$. By (4.163) we have $H_2(X') - n \geq H_{min}(X') - n \geq n - n = 0$, and thus $2^{-0.5(H_2(X')-n)} \leq 1$. This is an indicator that for X' the situation is even more favorable than for the ideal case at the cost of Δn additional input bits. 620

[non-uniformly distributed pre-images, $\gamma \gg 1$] In the derivation of formula (4.169), we applied the assumption that the random variables $T_1, T_2, \ldots, T_v$ are iid (cf. par. 619). Although it may not be true in a strict sense, two features justify this assumption. At first, $|f^{-1}(\{a_2\})|/|A_1| \approx 2^{-m}$. In other words: for parameters n, m that are relevant for functionality class PTG.3 (and NTG.1), the size of $f^{-1}(\{a_2\})$ is very small compared to the number of elements in A_1. Secondly, by assumption the intermediate random numbers have been generated by a PTG.2-compliant PTRNG 621

(cf. par. 616), which means that their distribution is not very far from the uniform distribution. For very extreme input distributions, the independence assumption might be invalid.
Note: We allow applying this formula for functionality class NTG.1, too, because in this case, Δ is usually very large.

4.5 Stochastic model, online test, total failure test, start-up test

622 TRNGs such as PTRNGs and NPTRNGs should provide information-theoretic security. The degree of randomness can be quantified by the entropy of the generated random numbers.

623 The evaluation processes for PTRNGs and NPTRNGs are, however, very different.

624 The main reason is that PTRNGs use physical noise sources. A physical noise source exploits physical phenomena (thermal noise, shot noise, jitter, metastability, radioactive decay, etc.) from dedicated hardware designs (using diodes, ring oscillators, etc.) or physical experiments to produce digitized random data. Dedicated hardware designs can use general-purpose components (like diodes, logic gates, etc.) if the designer is able to understand, describe and quantify the characteristics of the design that are relevant for the generation of random numbers.

625 In contrast, NPTRNGs exploit non-physical noise sources. Non-physical noise sources typically exploit system data (RAM data, system time, etc.) and / or user interaction (e.g., mouse movement, key strokes) to produce digitized random data.

626 Different copies of physical noise sources (e.g., within a chip series) are identically designed, and therefore their stochastic behaviour may not be identical but essentially similar. In contrast, non-physical noise sources are not under the control of the designer. For NPTRNGs running on different platforms, the behavior might be very different.

627 Finally, the central task of PTRNG and NPTRNG evaluations is to provide a lower entropy bound per internal random number bit. For PTRNGs, AIS 31 demands a so-called stochastic model. In most Common Criteria evaluations the evaluated PTRNGs are based on electronic circuits.

628 The stochastic model is the core of any PTRNG evaluation according to AIS 31. In Subsection 4.5.1 the concept is motivated and explained. In Subsection 4.5.2 the theoretical explanations are illustrated by an elementary example.

629 Subsections 4.5.3, 4.5.4, and 4.5.5 treat online tests, total failure tests, and start-up tests. AIS 31 requires that the online test is tailored to the stochastic model. The start-up tests should also consider the stochastic model, while the total failure tests should be based on a failure analysis of the physical noise source.

630 In the literature stochastic models of many real-world PTRNG designs have been studied. In Section 5.4 several stochastic models of real-world physical noise sources and two generic stochastic models are are discussed, and references are provided.

4.5.1 Stochastic model: motivation and definition

Finally, the random numbers delivered by a PTRNG to the consuming cryptographic application (external random numbers) shall be suitable. That is, they must meet the security requirements or the assumptions of the consuming application, which usually means being iid and uniformly distributed (i.e., unbiased). In the terminology of AIS 31, internal random numbers are the finalized random numbers ready for output that are still inside the RNG security boundary. The external random numbers are subsets (usually subsequences) of the generated internal random numbers that are passed to the requesting application outside the security boundary of the RNG. 631

The goal is to guarantee a lower entropy bound per bit of the internal random numbers. 632

Unfortunately, there do not exist effective (reliable) estimators or black box tests for the entropy of a given, arbitrary sequence of random numbers without further information, i.e., without stochastic assumptions on its distribution. 633

This is because entropy is a property of random variables but not of their realizations (here: random numbers); see Sect. 4.3, for example. For this purpose (verification of a lower entropy bound), the functionality classes PTG.2 and PTG.3 of AIS 31 require a stochastic model. 634

We interpret the raw random numbers $r_1, r_2, \ldots$ and the internal random numbers $y_1, y_2, \ldots$ as realizations of random variables $R_1, R_2, \ldots$ and $Y_1, Y_2, \ldots$, respectively. Analogously, we interpret observable and measurement values of the physical noise source (if relevant for the development, justification, and verification of the stochastic model) as realizations of random variables, too. The random variables R_j and Y_j are discrete. The random variables R_j assume values in $\{0,1\}$, $\{0,1\}^k$, or $\mathbb{Z}$, while the Y_j are $\{0,1\}^m$-valued. Here, k and m denote suitable integers. The random variables that quantify the stochastic behaviour of observables usually are real-valued. 635

[use of language] If there is no risk of misunderstanding, we loosely speak of the entropy per raw random number, per raw random number bit, per internal random number, etc. A more precise but more clumsy formulation, of course, would be 'the entropy per corresponding random variable' or even better 'the average gain of entropy per corresponding random variable' if dependent random variables are concerned.
Note: This applies to Shannon entropy and min-entropy. 636

A stochastic model provides a partial mathematical description (of the relevant properties) of a physical noise source using random variables. The stochastic model shall allow the verification of a lower entropy bound for the internal random numbers (or for intermediate random numbers) during the lifetime of the PTRNG, even if the quality of the raw random numbers goes down. 637

If there is no post-processing, the raw random numbers and the internal random numbers coincide. Formally, the post-processing equals the identity mapping. 638

Of course, a precise analysis of the impact of a (DRG.3-compliant) cryptographic post-processing on the entropy per random bit is infeasible. Instead, the cryptographic post-processing may be interpreted as a random mapping (with particular properties). Sect. 4.4 provides many results on random mappings, which may be useful for this purpose. 639

Note: This scenario is relevant for functionaliy class PTG.3. The input values to the cryptographic post-processing algorithm are called intermediate random numbers.

640 Due to pars. 639 and 637 the first part of the evaluation (stochastic model) is identical for evaluations with regard to both functionality classes PTG.2 and PTG.3.

641 For 'real-world' PTRNGs, the distribution of the underlying random variables $R_1, R_2, \ldots$ and $Y_1, Y_2, \ldots$ (cf. par. 635) is usually unknown. To some degree the distribution may vary over time, e.g., due to aging effects, changing environmental conditions, etc.; cf. par. 668 to 671.

642 [stochastic model, optimal case] Ideally, a stochastic model is a family of probability distributions that contains the true distribution of the raw random numbers or of suitably defined auxiliary random variables during the lifetime of the PTRNG, even if the quality of the digitized data goes down. This family of distributions may depend on one or several parameters (typically, one to three). In Subsection 4.5.2, and in Subsections 5.4.2 to 5.4.6 such scenarios are discussed.

643 The ambitious goal described in par. 642, namely defining a family of probability distributions that contains the true distribution of the raw random numbers or of suitably defined auxiliary random variables, may not always be feasible because in certain scenarious it would be too complicated. Anyway, we interpret the raw random numbers or suitably defined auxiliary random numbers as realizations of random variables. It suffices if the stochastic model focuses and quantifies particular features of the unknown distributions *provided that this allows the quantification of a lower entropy bound for the raw random numbers or auxiliary random variables (and, finally, for the internal random numbers)*. Other sources of randomness are not credited.
Example: Assume that for a particular physical noise source several sources of randomness have impact on the distribution of the raw random numbers or on certain auxiliary random variables. It is permitted that the stochastic model focuses on one source of randomness (e.g., on thermal noise), if evidence can be given the other sources of randomness (e.g., flicker noise) are independent from the selected one.

644 In the best case, the stochastic model would contain the true (but unknown) distribution of the internal random numbers, or more precisely, of the corresponding random variables $Y_1, Y_2, \ldots$.

645 However, in many cases algorithmic post-processing (in combination with the distribution of the raw random numbers) is too complicated for an explicit formulation of the stochastic model for the internal random numbers and, moreover, for a sound and reliable verification and thorough mathematical analysis of the stochastic model, which is even more important.

646 For these reasons AIS 20/31 does not require a stochastic model of the internal random numbers. Instead, AIS 31 demands a stochastic model of the raw random numbers (or of suitably defined auxiliary random variables), and on the basis of this stochastic model, a lower entropy bound for the internal random numbers shall be derived.

647 Of course, from an abstract point of view, the algorithmic post-processing transforms a stochastic model of the raw random numbers into a stochastic model of the internal random numbers. Under favorable circumstances, e.g., if the algorithmic post-processing is not too complicated, it may be possible to explicitly formulate and to analyze the 'transformed' stochastic model.

This is trivially the case, of course, in the absence of an algorithmic post-processing algorithm. Non-trivial positive examples are given in Subsections 5.1.1, 5.1.2 and 5.1.3 (XORing independent raw random numbers, von Neumann transformation, thinning out of homogeneous Markov chains). However, this is not always the case. Par. 740 in Section 5.1 provides an elementary counterexample where the raw random number bits are XORed to the feedback value of an LFSR. 648

However, it is not necessary to analyze the transformed stochastic model. As stated above it suffices to derive a lower entropy bound for the internal random numbers on the basis of the stochastic model. In the LFSR example from par. 740 (depending on the initial state of the LFSR), the raw random numbers are mapped 1-1 to the internal random numbers. Hence, the (average) entropy per bit is trivially the same for both the raw random numbers and the internal random numbers. On the other hand, even for iid $B(1,p)$-distributed raw random numbers, it is hardly feasible to provide an explicit description of the distribution of the internal random numbers unless $p = 0.5$. 649

In some PTRNG designs the stage where the raw random numbers occur first may not be uniquely identifiable. In such cases different interpretations are permitted, but we strongly recommend selecting an early stage because this usually simplifies the justification of the stochastic model, see pars. 661, 662, 663, and 664. 650

As an alternative to a stochastic model for the raw random numbers, in some scenarios it can be favorable to consider a stochastic model for suitably defined 'auxiliary' random variables. If this stochastic model finally allows the derivation of a lower entropy bound for the internal random numbers, this approach is permitted. An example is discussed in Subsect. 5.4.2. 651

Different instances of a PTRNG design (e.g., PTRNGs on chips of some series) can, to some degree, behave differently. Even the distribution of a single PTRNG changes to some degree during its lifetime. This may be caused by tolerances of components (of the physical noise source), variations of the environmental conditions (temperature or voltage, for example), and aging effects, for example. 652

The stochastic model shall contain all distributions that can occur in any possible scenario for any copy of the PTRNG using the design under consideration (usually running on essentially the same hardware). Different parameters correspond to different distributions. 653

Assume that $\mathcal{S}_1$ denotes a stochastic model for some PTRNG, and that $\mathcal{S}_2$ is a superset of $\mathcal{S}_1$, i.e., that each distribution of $\mathcal{S}_1$ is also contained in $\mathcal{S}_2$. Then $\mathcal{S}_2$ is a stochastic model for this PTRNG, too. 654

The use of a 'large' stochastic model (depending on many parameters) has both advantages and disadvantages. The advantage is that the verification of the stochastic model can become easier, and thus the proof that the true distribution(s) of the raw random numbers (of all copies, under all conditions of use) are contained in the admissible set of distributions (Example: $\mathcal{S}_2 \cong$ Markovian model vs. $\mathcal{S}_1 \cong$ iid model). When estimating the parameters those parameter components from which the true distribution does not (significantly) depend on, should be rather small (partly 655 caused by statistical noise). For the verification of the entropy bound, the larger stochastic model usually should not cause serious additional problems apart from the fact that the entropy estimation formula becomes more difficult, see par. 685. An obvious disadvantage of this approach is that the online test must cover a wider range of admissible distributions. This possibly reduces the effectiveness of the online test. In pars. 685 to 687 an example is discussed.

656 [Advantages of a stochastic model] It is a notable advantage of stochastic models that they (ideally) comprise (parametric) families of distributions. First of all, the justification / verification that a whole class of distributions contains the true distribution is easier than showing that it matches with a particular single distribution. Moreover, as already pointed out in par. 641, even the distribution of a single PTRNG varies to some degree while the PTRNG is in operation. Finally, the distributions contained in a stochastic model usually allow a unified analysis since they only differ by their parameters.

657 Of course, the stochastic model shall also contain distributions that correspond to defective states of the physical noise source that yield non-tolerable weaknesses of the internal random numbers (too large entropy defects). When the PTRNG is in operation, non-tolerable defective behavior must be detected. Therefore, suitable online tests and total failure tests are required. Online tests and total failure tests are explained in Subsections 4.5.3 and 4.5.4.

658 The last feature of the stochastic models addressed in par. 656 supports the estimation of entropy. In a first step the parameters of the true distribution are estimated on the basis of observed raw random numbers (or auxiliary random numbers). This parameter estimate is substituted into an entropy formula that fits the distribution of the stochastic model. This yields an estimate for the entropy per random number (or more precisely, per random variable). In Subsection 4.5.2 this procedure is illustrated by an elementary example (coin tossing). Par. 527 provides the entropy formula for homogeneous Markov chains, for example.

659 The experiments and the entropy estimations shall be performed under different environmental conditions. The evaluated prototypes shall meet the requirements of functionality class PTG.2 or PTG.3 under all admissible environmental conditions.

660 When the PTRNG is in operation, the online test shall guarantee that non-tolerable weaknesses of the random numbers lead to a noise alarm (see Subsect. 4.5.2).

661 [verification of the stochastic model] Finally, a stochastic model is a claim that the random values (usually, raw random numbers) produced by some physical experiment or an electronic circuit follow a probability distribution that is contained in a specified family of distributions. As already mentioned the correctness of the stochastic model has to be justified and verified.

662 [verification of the stochastic model] The stochastic model shall be supported by technical arguments based on the design of the physical noise source and findings in the literature. This requires at least a qualitative understanding of the physical noise source.

663 [verification of the stochastic model] Empirical data gained from the physical noise source (analog data like voltage or timing variations, raw random numbers, etc.) shall be used to develop, confirm, and adjust the claimed stochastic model or subclaims thereof. Different environmental

conditions (temperature, voltage, etc.) shall be considered. This may be done by statistical tests that are tailored to the physical noise source and the stochastic model. (These statistical tests are applied *in addition* to the evaluator black box test suites T_{rrn} and T_{irn} that are described in Subsects. 4.6.2 and 4.6.3.) This should also increase an understanding of the physical noise source that is exploited by the RNG and support the confidence in the stochastic model. For very simple and theoretically well-understood designs (e.g., for the coin tossing example or if the PTRNG exploits certain physical experiments), the evaluator might leave out or at least reduce such experimental investigations.

[verification of the stochastic model] An interesting question is to what extent the raw random numbers depend on variations of the environmental conditions (e.g., temperature, voltage) and of the characteristics of the physical noise source. Such dependencies may be very different (and difficult to quantify). A high-resolution measurement of the power consumption, for example, might reveal correlations to the raw random numbers. The analysis of the TOE should consider the question of whether variations of some parameters can cause significant changes of relevant statistical properties of the raw random numbers because such a behavior might be exploited by an adversary. 664

The developer has to specify the allowed ranges of the environmental conditions. The evaluation shall verify that the entropy of the internal random numbers remains large enough as long as the parameters stay in the permitted ranges. 665

[stage of the stochastic model] Considering the stochastic model at an early stage of the random bit generation usually has the advantage that random data can still be clearly distinguished from ideal output in the case of a significant entropy defect. Furthermore, the supporting technical rationale usually allows the evaluator to confirm that technical arguments predict the general shape of the distribution of random output. 666

[stage of the stochastic model] In contrast, assume that a PTRNG is analyzed at a stage where the random output is indistinguishable from ideally distributed output (e.g., after cryptographic post-processing) almost irrespective of the amount of true entropy contained therein. For example, because these are pseudorandom. This cannot lead to a successful evaluation. 667

For non-stationary stochastic processes the sound and trustworthy verification of a stochastic model and the estimation of the parameters is rather difficult and can be practically infeasible. Stationarity facilitates the tasks in an evaluation considerably, in particular since many transformations maintain stationarity; cf. Sect. 4.1, par. 506, and Sect. 5.4, for real-world examples. Stationarity means that the process behaves time-invariantly throughout the entire time. Hence, demanding stationarity in a strict mathematical sense would be too restrictive since the parameters of the true distributions may vary to some degree over time (due to aging effects, varying environmental conditions, etc). Nevertheless, the property that *observing a physical noise source at one point in time is representative for other points in time* is an important prerequisite for a meaningful PTRNG evaluation. Therefore, this document requires that the raw random numbers (resp., auxiliary random numbers) belong to a 'time-locally stationary' stochastic process; cf. par. 669). 668

'Time-local stationarity' is a AIS 31-specific term. It means that the raw random numbers (resp., the auxiliary random numbers), or more precisely, the corresponding random variables may be 669 viewed as stationarily distributed over 'short' time-scales that are 'large' compared to the sample size of the online tests and the evaluator tests (e.g., to estimate parameters). Within such periods the relevant distribution parameters shall change at most marginally.

670 This approach takes advantage of the properties of (mathematical) stationarity but also takes into consideration that, for real-world PTRNGs, stationarity in a strict mathematical sense may not exist due to reasons which have already been discussed above. PTRNGs can generate hundreds of kBits or even MBits of raw random numbers per second so that within a few seconds a very large amount of random numbers are generated.

671 The aging effects of the analog components may slowly change the distribution of the parameters. However, such effects are not relevant over short time scales. Transient effects on the parameters during the start-up of the physical noise source may be ignored provided that no internal random numbers are output. These raw random numbers can still be used to seed the internal state of the algorithmic post-processing algorithm (if with memory) and the cryptographic post-processing algorithm but no entropy is credited for them.

672 As already mentioned in par. 659, the parameters of the underlying distribution and, based on that, the entropy of the raw random numbers (or, the auxiliary random numbers) shall be estimated under different environmental conditions. Minor variations of the estimated parameters under changing environmental conditions are expected and tolerable as long as the entropy remains large enough. Even then, if the parameter estimates vary 'significantly', this might be a starting point for a fault attack on the physical noise source and thus should be considered in the overall evaluation of the TOE (cf. Sect. 2.1).

673 In the analysis of PTRNGs, we assume that the adversary knows the design details but does not have knowledge of any 'internal state' of the physical noise source. The adversary, for instance, does not know the current analog state of a Zener diode.

4.5.2 Example: Stochastic model for coin tossing

674 In this subsection we illustrate the concept of a stochastic model by using an elementary example. A single coin is tossed repeatedly by a human operator. For simplicity we assume that this operator neither has the opportunity to cheat (i.e., to precisely influence the outcome of the coin tossings), e.g., due to a minimum throw height of each valid coin toss nor that he is interested in cheating.

675 It should be noted that this experiment would not be viewed as a PTRNG in the sense of AIS 31 because of the significant impact of the human operator. However, it provides an appropriate example to illustrate the concept of stochastic models.

676 We identify the outcomes 'head' and 'tail' with '1' and '0', respectively. We assume that the outcomes $x_1, x_2, \ldots$ are realizations of binary-valued random variables $X_1, X_2, \ldots$. In this example, these are our raw random numbers.

677 Since a coin has no memory and since the physical structure of the coin remains identical (at

least during reasonable time periods), we may assume that the random variables $X_1, X_2, \ldots$ are iid $B(1,p)$-distributed with unknown parameter p. In this example the stochastic model is given by a one-parameter family of distributions.
Note: An alternative, more formal description of this stochastic model is given by $\{B(1,p)^n \mid p \in [0,1]\}$. The n-fold product measure $B(1,p)^n$ describes the distribution of the random vector $(X_1, \ldots, X_n)$.)

This stochastic model does not only apply to a single coin but to any coin (even though for different parameters). 678

For real-world PTRNGs the verification of the stochastic model is more difficult. In Sect. 5.4 several examples are discussed. 679

When tossing the coin N times, the mean value provides an estimate $\widetilde{p}$ for the unknown parameter p 680

$$\widetilde{p} := \frac{1}{N} \sum_{j=1}^{N} x_j \,. \tag{4.170}$$

The strong law of large numbers guarantees that the right-hand side converges 'almost surely' to the parameter p as N tends to infinity.

By formula (4.37) the estimation error in (4.170) is $\leq \epsilon$ with probability $1 - 2\Phi(-2\epsilon\sqrt{N})$. 681
Numerical example: For $(\epsilon, N) = (0.01, 10000)$, we have $-2\epsilon\sqrt{N} = -2.0$, and $2\Phi(-2.0) \approx 0.046$. Increasing N from 10.000 to 100.000 reduces this probability to 10^{-9}.

Since the random variables $X_1, X_2, \ldots, X_N$ are iid, $H(X_1, X_2, \ldots, X_N) = NH(X_1)$. Substituting $\widetilde{p}$ from (4.170) into the one-dimensional Shannon entropy formula yields the entropy estimate for $H(X_1)$ 682

$$\widetilde{H}(X_1) := -\left(\widetilde{p}\log_2(\widetilde{p}) + (1-\widetilde{p})\log_2(1-\widetilde{p})\right) \,. \tag{4.171}$$

The linear Taylor expansion gives an approximation of the estimation error 683

$$\widetilde{H}(X_1) - H(X_1) \approx \Delta p \left(-\log_2(p) + \log_2(1-p)\right) \tag{4.172}$$

where $\Delta p := \widetilde{p} - p$.

Analogously to par. 682 684

$$\widetilde{H}_{min}(X_1) := -\log_2\left(\max\{\widetilde{p}, 1-\widetilde{p}\}\right) \,. \tag{4.173}$$

provides an estimator for the min-entropy.

Now consider another stochastic model that assumes that the random variables $X_1, X_2, \ldots$ form a homogeneous Markov chain with transition matrix $P = (p_{ij})_{0 \leq i,j \leq 1}$. This stochastic model depends on two parameters $p_{01} := \text{Prob}(X_{n+1} = 1 \mid X_n = 0)$ and $p_{10} := \text{Prob}(X_{n+1} = 0 \mid X_n = 1) \in [0,1]$. The transition matrix P reads 685

$$P = \begin{pmatrix} 1 - p_{01} & p_{01} \\ p_{10} & 1 - p_{10} \end{pmatrix} \,. \tag{4.174}$$

If $p_{10} = 1 - p_{01}$, then both rows of P are identical, which means that the random variables $X_1, X_2, \ldots$ are iid. In particular, the Markovian stochastic model is a superset of the iid stochastic model discussed above and thus is a valid stochastic model for the coin tossing experiment, too (cf. par. 654). Here, the stochastic model depends on two parameters. The set $\{(p_{01}, p_{10}) \mid 0 \leq p_{01}, p_{10} \leq 1\}$ specifies the admissible parameters.
Note: The iid model is described by the subset $\{(p_{01}, p_{10}) \mid p_{10} = 1 - p_{01}, 0 \leq p_{01} \leq 1\}$.

686 [Continuation of par. 685] If $0 < p_{01} + p_{10} < 2$, the Markov chain is ergodic, and the distributions $\nu_1, \nu_2, \ldots$ converge to the limiting distribution $\nu = (\frac{p_{10}}{p_{01}+p_{10}}, \frac{p_{01}}{p_{01}+p_{10}})$ (par. 502). The special cases $p_{01} = p_{10} = 0$ and $p_{01} = p_{10} = 1$ correspond to noise sources that generate constant raw random number bit sequences or alternating raw random number bit sequences $\ldots, 0, 1, 0, 1, \ldots$, respectively. By (4.67) we obtain the conditional entropy

$$\begin{aligned} H(X_{m+1} \mid X_m) &= -\frac{p_{10}}{p_{01}+p_{10}}\left(p_{01}\log_2(p_{01}) + (1-p_{01})\log_2(1-p_{01})\right) - \\ &\quad \frac{p_{01}}{p_{01}+p_{10}}\left((1-p_{10})\log_2(1-p_{10}) + p_{10}\log_2(p_{10})\right) . \end{aligned} \tag{4.175}$$

The conditional entropy (4.175) quantifies the average increase of entropy per raw random number bit.
Note 1: The gain of entropy by the next raw random number bit depends on the current random raw random number bit, thus on the first or on the second row of the transition matrix P.
Note 2: If min-entropy is claimed, pars. 531 to 539 in Sect. 4.3 can be useful.

687 [Continuation of par. 686] For the Markovian stochastic model, not only one parameter p (as in the iid model) but two parameters p_{01}, p_{10} have to be estimated and then substituted into the entropy formula (4.175).

4.5.3 Online test

688 The task of the online test (of a PTRNG) is to detect sufficiently soon when the quality of the random numbers becomes too low (compared to the requirements of functionality classes PTG.2 or PTG.3) while the PTRNG is in operation. An effective online test is mandatory for the functionality classes PTG.2 and PTG.3.

689 The stochastic model shall contain all possible distributions of the raw random numbers (or, alternatively, of auxiliary random variables; cf. Subsect. 5.4.2) that may occur during the lifetime of the PTRNGs. As already pointed out in Subsect. 4.5.2, even for a given PTRNG during its lifetime, some variation of the parameters is normal, and different PTRNGs of the same type can differ to some extent. The online test shall detect if the true distribution has left the subset of appropriate parameters A_{good} (see pars. 690 and 691). All parameters in A_{good} provide enough entropy (cf. PTG.2.2, resp. PTG.3.5). This may be done directly by guessing the parameters, or indirectly by statistical tests that fail if the true parameters leave the set A_{good}.
Note: Depending on the entropy claim, entropy means Shannon entropy, min-entropy, or both.

690 Ideally, the online test would never fail if the true parameter(s) belong to A_{good} but always fail whenever the parameter(s) lie outside of A_{good}, i.e., if they lie in its complement A_{bad}. Fig. 7

illustrates the failure probabilities of an ideal test. Of course, this aim cannot be achieved because the discriminatory power of statistical tests (with finite sample size) is not infinite. Fig. 8 shows a more realistic picture.

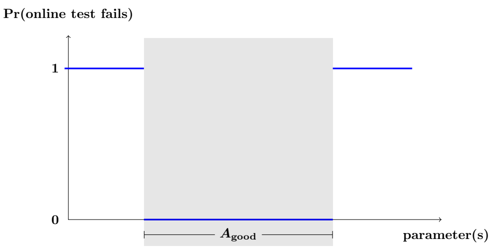

Figure 7: ideal online test: never fails if the true parameter(s) is in A_{good} but always fails otherwise

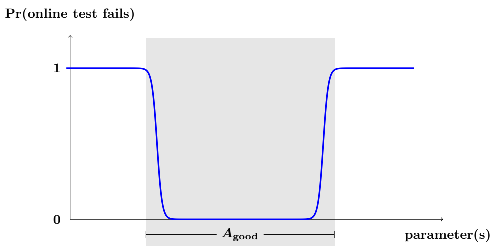

Figure 8: more realistic online test: significant failure probability near the border between A_{good} and A_{bad}, and large failure probability outside A_{good}

In Fig. 8 the failure probability for the parameter set A_{bad} is appropriate but also for certain parameters inside A_{good} (implicitly defined by PTG.2.2 or PTG.3.5), namely for the 'border region', the failure probability is rather large. This is not a security problem but affects the availability of the PTRNG. 691

For PTRNG evaluations only the behavior of the online test on the set A_{bad} is relevant. In particular, the applicant has to give evidence that Requirement PTG.2.4 or PTG.3.7 (as appropriate), is fulfilled. 692

From a security point of view, the behavior of the online test on the set A_{good} is irrelevant. However, availability is an important feature of IT products. In pars. 694 to pars. 698 we formulate some thoughts about how to combine security (effectiveness of the online test) with availability (not too many 'false' failures of the online test). In Section 5.5.1 concrete examples are discussed. 693

Often, the entropy of the raw random numbers is larger than required to fulfill the entropy requirements specified by the functionality classes PTG.2 and PTG.3. Assume for the moment that during operation, the true parameters of all (properly working) copies of the PTRNG design under evaluation are contained in a subset $A_{real} \subseteq A_{good}$, which contains 'very good' parameters that parametrize 'very good' distributions. From a security point of view, failures of the online test on the difference set $A_{good} \setminus A_{real}$ are neither necessary nor harmful, and the availability is not affected because these parameters should never occur for properly working PTRNGs. 694

If the designer assumes that for all (properly working) PTRNGs under consideration, the true parameters indeed always stay in such a subset $A_{real} \subseteq A_{good}$, he can utilize this property to design an effective online test. The applicant does not need to provide evidence that the true parameters of the PTRNG copies are indeed always contained in the set A_{real}. Overly optimistic assumptions, however, may limit the availability of the PTRNG, but this is primarily an issue for the applicant to consider. 695

696

Of course, the smaller A_{real} is (i.e., the larger the difference set $A_{good} \setminus A_{real}$), the easier it is to design a suitable (efficient) online test, which on the one hand detects sufficiently soon when the true parameter(s) leave A_{good} and on the other side hardly limits the availability of appropriate PTRNGs. Fig. 9 shows an example where the online test rarely fails if the true parameters are in A_{real}, while the failure probability is large for A_{bad}. Fig. 10 illustrates the relation between these subsets for a stochastic model that depends on two parameters (e.g., a Markovian model). Note: A_{good} includes the green and the yellow area.

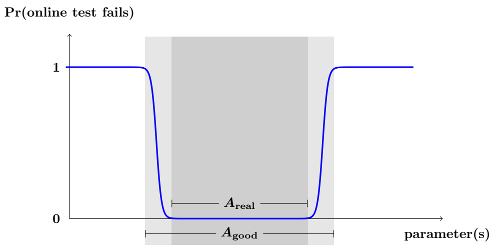

Figure 9: appropriate online test: low failure probability on A_{real} and large failure probability on A_{bad}

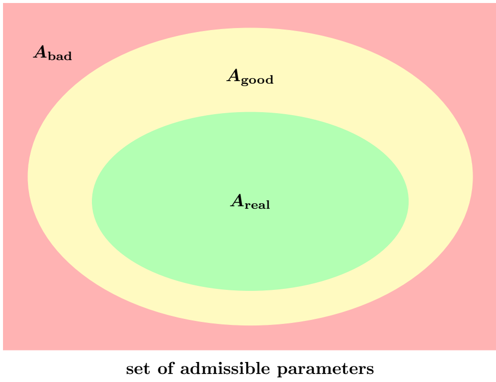

Figure 10: Relation between A_{real}, A_{good}, and A_{bad}

697 Of course, assuming a smaller subset A_{real} increases the technical requirements on the PTRNG design, including aging effects, tolerances of components, and the dependence on environmental conditions.
Note: As already mentioned in Chapter 3, the lower entropy bound defined in PTG.2.2 and PTG.3.5 could have been set larger. One reason for omitting this option is to enable effective and efficient online tests.

698 There is a 1-1 correspondence between the admissible parameters of the stochastic model and the possible distributions of the raw random numbers (or, alternatively, to distributions of the auxiliary random variables). This allows the identification of parameters with distributions. We may view A_{real} as the (composite) null hypothesis of the online test, and the inappropriate parameters A_{bad} as the alternative hypothesis.

699 [terminology] The online test, or more precisely, the online test scheme (synonymously, the online test procedure), consists of one or several statistical tests (applied simultaneously or consecutively when the PTRNG is in operation), evaluation rules, a calling scheme (cf. par. 712), and a specification of what happens if the online test fails ('consequences of a noise alarm'; cf. par. 715).

700 [terminology] If it is unambiguous we alternatively use the term 'online test' for the applied statistical test(s) but also in place of 'online test scheme' or 'online test procedure'.

701 The developer shall provide evidence that the online test (i.e., the online test scheme) is appro-

priate, i.e., that it fulfills Requirement PTG.2.4 or PTG.3.7, respectively.

Which random numbers should be tested by the online test? This is a natural question. The general advice is to apply the online test to the raw random numbers even if the stochastic model considers 'auxiliary' random variables. 702

If the PTRNG applies an algorithmic post-processing algorithm, AIS 31 principally allows online tests on the internal random numbers. However, usually the design of the online test and the verification of its effectiveness, if possible at all, are at least significantly more difficult if the online test is applied to the internal random numbers. Exceptions are possible if the transformed stochastic model can be described explicitly (cf. pars. 647 and 648) or if the post-processing can be reversed (cf. par. 705). However, in the first scenario it should be more favorable to test the raw random numbers, cf. par. 710. It is easy to see that, at least if the post-processing algorithm is stateless, for each online test on the internal random numbers, an online test on the raw random numbers exists, that is at least as effective. 703
Note: From a logical point of view, the online test on the internal random numbers can then be interpreted as an online test on the raw random numbers.

Online tests on the output of a cryptographic post-processing algorithm (as substitute for an online test on the raw random numbers) are meaningless and therefore not permitted. 704

Example: Consider the PTRNG design described in par. 740 where the raw random number bits are XORed to the feedback value of an LFSR. Even if the physical noise source would fail completely and generate constant sequences of 0s or 1s (total failure; cf. Subsect. 4.5.4), the statistical properties of the internal random numbers should pass common statistical tests (apart from linear complexity tests, of course) unless the LFSR is too short. Effective online tests on the internal random numbers have to reverse the effect of the LFSR, which means that finally the raw random numbers would be tested anyway. 705

The example in par. 705 underlines the general advice from par. 702 that the raw random numbers should be tested. 706

The online test shall be tailored to the stochastic model. 707

Example: A monobit test counts the number of 0's and 1's. A monobit test is appropriate for the stochastic model from par. 677 (coin tossing, iid model). The monobit test shall detect when the parameter p moves or lies outside the permitted area. Of course, a monobit test is not appropriate for the Markovian stochastic model that was discussed in pars. 685 to 687. Assume, for example, that $p_{01} = p_{10} = 0.1$. By (4.175) the conditional entropy $H(X_{m+1} \mid X_m) = 0.468$ is far too low. On the other hand, by par. 686, the limiting distribution is $\nu = (0.5, 0.5)$. Hence, a monobit test would not detect even this dramatic entropy defect. In a Markovian stochastic model an appropriate online test must consider the transition probabilities. 708
Note: The other direction, selecting an online test that tests properties beyond the given stochastic model, is permitted. For an iid model a poker test can be a suitable choice. Apart from the bias it would detect any (small) dependencies that are not covered by the stochastic model. Large dependencies should not occur, since otherwise the stochastic model would be inappropriate.

709

Generally, the more comprehensive a stochastic model, i.e., the more parameters it includes, the easier it should be to verify. On the negative side, the specification of an effective and efficient online test may become more difficult.

710 Assume that a physical noise source generates iid $B(1,p)$-distributed raw random number bits for which $|p-0.5|$ is too large so that the entropy per bit is insufficient. To increase the entropy per random bit, non-overlapping pairs of raw random number bits are XORed (algorithmic post-processing). In this scenario it would also be easy to formulate a stochastic model for the internal random numbers (cf. par. 648 and Subsect. 5.1.1), because the transformed stochastic model is again iid. But, even in this scenario it is more favorable to test the raw random numbers in place of the internal random numbers: First, XORing non-overlapping pairs of raw random numbers reduces the sample size of the online test to 50%, and the algorithmic post-processing reduces the 'distance' between A_{real} and A_{bad}, thereby reducing the discriminatory power of the online test.
Note: An example is discussed in Subsect. 5.5.2.

711 The sample size of an online test is usually much smaller than the sample size of a typical evaluator test that is applied to some prototypes of the PTRNG. Consequently, the discriminatory power of a single online test is much smaller. In Subsection 5.5.2 an online test (i.e., an online test scheme) is discussed. In particular, a 'history variable' compensates for this effect to some degree as it captures long-term effects that result from deviations of the expectation of the test value.

712 [calling schemes] Apart from the choice of an appropriate online test, the calling scheme is important for the effectiveness of the online test scheme. The online test might, for example, be applied to all raw random numbers, to the raw random numbers that are used to generate the output data of the current request, or to the raw random numbers that precede those raw random numbers that are used to generate the current request. The developer has to justify that his choice is appropriate in the given scenario.

713 [calling schemes] A bad (i.e., not acceptable) solution would be to apply an online test to a sample of random numbers and output (non-tested) random numbers that were generated much later because after the online test, the behavior of the physical noise source might have changed considerably.

714 [calling schemes] The situation would change if the raw random number are tested with a suitable online test, and the internal random number are securely stored. (Of course, the respective memory had to be protected against manipulation and readout all the time. This is not an aspect of AIS 31 itself but of the overall evaluation; cf. Sect. 2.1.)

715 [noise alarm and false positives] Due to the nature of statistical tests, an online test could fail even if the entropy per bit is sufficiently large. In fact, failures would also occur for ideal RNGs. If the online test fails, this causes a noise alarm. Because erroneous (or accidental) noise alarms may occur, it is not obvious what should happen after a noise alarm.

716 [consequences of a noise alarm] Depending on the concrete application, different reactions to a noise alarm (triggered by the online test) can be appropriate. For example:

a) The most rigorous measure clearly is to stop the output of random numbers forever.

b) The device could be subjected to an 'emergency test' (without outputting internal random numbers). The emergency test would be used to determine whether the noise alarm was accidental or justified. In the first case the RNG again outputs random numbers, while in the second case, further output of random numbers is permanently prohibited.

c) A human operator checks the quality of the RNG (typically by appropriate statistical tests) before further output is allowed.

d) Noise alarms may be logged, but operation continues.

e) etc.

Whether certain options are possible (and reasonable) depends on the concrete application scenario.

[consequences of a noise alarm] The developer has to justify the suitability of the specified consequences of a noise alarm. This belongs to the evaluation of the online test. 717

[consequences of a noise alarm] It is not a valid option, of course, to just perform online tests until one online test is (accidentally) passed and then to continue as before. 718

4.5.4 Total failure test

The total failure test shall detect failures of the physical noise source that imply that the entropy per raw random number bit has decreased to (essentially) 0. A failed total failure test causes a total failure alarm. 719

The total failure test shall detect naturally occurring (usually permanent) total failures of the physical noise source. That is, the aim of a total failure test is testing for possible ways for the device to fail rather than detecting targeted attacks such as fault-injection attacks that cause a (usually transient) failure mode. Total failure tests might be able to detect targeted attacks, but in general additional countermeasures are required. If targeted attacks are relevant for the TOE in the intended usage, then potential attacks and the implemented countermeasures shall be analyzed within the overall evaluation of the TOE, cf. Sect. 2.1. 720

After a total failure the entropy per raw random number bit is essentially 0. Hence, a total failure must be quickly detected, cf. pars. 727 ff. In particular, no weak internal random numbers shall be output when a total failure has been detected. 721

The developer shall give evidence that the implemented total failure test is appropriate. 722

A thorough failure analysis of the physical noise source is indispensable. This analysis shall clarify which failures are technically plausible, and their impact on the raw random numbers should be described. 723

724

Technically, the total failure test can be realized by sensors or by statistical tests. AIS 31 does not specify any other requirements besides its effectiveness.

725 In the simplest scenario a total failure of the physical noise source implies constant sequences of raw random numbers (0's or 1's), e.g., due to a stuck flip flop. Of course, this behavior could be detected by the online test or by a statistical test that fails if the last (for example) 40 raw random number bits were constant. The choice of the threshold (e.g., 40) should consider the throughput of the RNG within its lifetime to prevent erroneous total failure alarms. A little bit more general is the repetition count test defined in [SP800-90B], Subsection 4.4.1, which can handle bit strings in place of bits.

726 Depending on the physical noise source and its digitization mechanism, it is possible that a total failure may lead to more complicated error patterns, e.g., alternating sequences $\ldots, 0, 1, 0, 1, \ldots$ or even sequences containing some remaining noise. For example, if a Zener diode fails, the analog-to-digital converter may yet receive some noise from an amplification circuit. Despite some remaining entropy, a failure of the Zener diode still constitutes a total failure of the noise source.

727 Immediately after a total failure has been detected, no 'weak' internal random numbers (containing only little entropy) shall be output. In particular, if the PTRNG design belongs to one of the following three design types

(i) no post-processing

(ii) simple algorithmic post-processing (with memory or without)

(iii) stateless post-processing based on cryptographic constructions

this means that no internal random numbers shall be output that have been generated after a total failure was detected.

728 Usually, immediate detection of a total failure can most effectively be attained by sensors or similar technical measures but not by statistical tests. However, FIFO buffers (par. 729) and cryptographic post-processing (par. 732) allow a delayed detection, thereby relaxing the requirements on the total failure test.

729 Assume that (i) the internal random numbers are stored in a FIFO buffer before they are output, and that (ii), by design, always at least t internal random number bits are stored in the FIFO buffer. In this case, it suffices if further output of internal random numbers is prevented (at the latest) after t internal random number bits have been output after the total failure has occurred. Compared to par. 727 this relaxes the requirement that the total failure has to be detected immediately.

730 When a post-processing algorithm is applied, the permitted delayed detection time in par. 729 can be translated into a requirement on the raw random numbers. If the PTRNG applies no post-processing or injective post-processing (i.e., one-to-one post-processing as the LFSR design in par. 740), par. 729 implies that further output of internal random numbers has to be prevented

at the latest after t raw random number bits (permitted by the FIFO) have been generated after the total failure has occurred. To be precise, if the post-processing algorithm generates k-bit internal random numbers, the threshold of t raw random number bits decreases to $t - k + 1$ because, in the worst-case, the total failure occurs when the last raw random number bit for an internal random number was generated. Similarly, if the PTRNG XORs non-overlapping pairs of raw random number bits, the permitted delayed detection time increases from t to $2t - 1$ raw random number bits.

It is a natural aim to keep t as small as possible. A dedicated statistical total failure test (e.g., 731
checking whether the last t raw random number bits have been identical) should be considered instead of using the online test (applied with an specific rejection area that is adjusted to the total failure case) because the sample size of the online tests is usually much larger than the sample size of a dedicated total failure test. The size of t is not a security feature and thus not prescribed by AIS 31.

[PTG.2.6, PTG.3.9] Assume that the PTRNG applies a cryptographic post-processing algorithm 732
that, viewed as a DRNG, belongs to functionality class DRG.2 or DRG.3. Furthermore, assume that the effective internal state comprises v bits, while k denotes the bit length of the internal random numbers. (This scenario applies to PTG.3-compliant PTRNGs.) After a total failure has occurred, at most $\lfloor v/k \rfloor$ further internal random numbers may be output that have been generated after the total failure has occurred. The justification of this relaxation is that the internal state should have accumulated v bits of entropy since the start of the PTRNG.

Permitted delayed detection times resulting from cryptographic post-processing and from FIFO 733
buffers can be combined.

4.5.5 Start-up test

When the PTRNG has been started, a so-called start-up test (a.k.a. self test) is performed. The 734
start-up test shall test for total failures and severe statistical weaknesses.

For these reasons the start-up test shall be tailored to the PTRNG. Usually, a reasonable choice 735
is to apply the online test (possibly with different evaluation rules).

4.6 Evaluator Black box Test Suites

[This subsection is under construction!]

Note: The (old) requirements (PTG.2.2) and (PTG.3.2) from draft v.2.35 have been cancelled. This has influence on the statistical black box test suites.

4.6.1 Specification of Statistical Tests

4.6.2 The T_{rrn} test suite

4.6.3 The T_{irn} test suite

5 Examples

This chapter discusses examples from several areas and illustrates general concepts that have been introduced in the previous chapters. We begin with algorithmic post-processing algorithms (Sect. 5.1), and then we discuss exemplary verifications of algorithmic requirements of functionality classes DRG.2, DRG.3, and DRG.4 (Sect. 5.2). In Sect. 5.3 the conformity of the Hash_DRBG [SP800-90A] to the algorithmic requirements of class DRG.3 is verified. Section 5.4 investigates stochastic models for real-world designs of physical noise sources. In Sect. 5.5 strategies for online tests are discussed, and Sect. 5.6 deals with Linux /dev/random and /dev/urandom. Applicants, designers, and evaluators can refer to the discussed examples and to the results that are provided in this chapter. 736

5.1 Examples of Algorithmic Post-processing

In this section we discuss several examples of algorithmic post-processing. For further expositions, we refer the interested reader, e.g., to [Schi09b], section 3.5, or to [DiBi07; Lach08], for example. 737

By an algorithmic post-processing algorithm, we mean a relatively simple mapping, which (ideally) has been selected with regard to the admissible distributions of the input data, usually the raw random numbers. In other words: the algorithmic post-processing should be tailored to the stochastic model. It shall be possible to determine the impact of the algorithmic post-processing on the entropy per random bit. 738

The entropy per bit cannot be increased by injective mappings. 739

An example of this type this type of post-processing would be, for example, a noise source that outputs one raw random number bit per clock cycle. An LFSR is clocked synchronously, and the raw random number bit is XORed to the feedback value of the LFSR. The output of the LFSR are the internal random numbers. Ignoring the initial state of the LFSR, this post-processing algorithm is injective. It thus cannot increase the entropy per bit but transforms weaknesses of the raw random number bits into others statistical defects, e.g., a bias into dependencies (see, e.g., [Schi09b], Example 3.7)., In particular, if the binary-valued random variables $R_1, R_2, \ldots$ and $Y_1, Y_2, \ldots$ model the raw random numbers and the internal random numbers, respectively, for any distribution of $R_1, R_2, \ldots$ (at least in average) we have 740

$$H(Y_{n+1} \mid Y_1, \ldots, Y_n) \geq H(R_{n+1} \mid R_1, \ldots, R_n)\,. \tag{5.1}$$

Ignoring the initial internal state of the LFSR (or assuming that this initial state is known), then the previous formula is an equality.

If the raw random numbers already have enough entropy, it suffices to prove that $H(Y_{n+1} \mid Y_1, \ldots, Y_n) \geq H(R_{n+1} \mid R_1, \ldots, R_n)$. Otherwise, the gain of entropy per bit has to be verified, which is usually more difficult. 741

To increase the entropy per bit, one has to compress the input stream, resulting in a lower output rate. 742

743 The examples discussed below do not have an internal state, which means that they have no memory. Of course, designs with memory are also possible; cf. Par: 740, for example.

5.1.1 Fixed compression rate

744 In this subsection we treat several examples of algorithmic post-processing algorithms with fixed compression rates.

745 [Xoring non-overlapping k-bit subsequences, iid] If the random variables $R_1, R_2, \ldots$ are iid $B(1, 0.5+\epsilon_0)$-distributed for some ϵ_0, the random variables $Y_j := R_{k(j-1)+1} + \cdots + R_{kj} (\text{mod } 2)$ are iid, too. Setting $\epsilon := 2\epsilon_0$ (equivalently, $\epsilon_0 = 0.5\epsilon$) we obtain

$$\text{Prob}\,(Y_j = 0) = \sum_{i=0;i \text{ even}}^{k} \binom{k}{i} (0.5 + 0.5\epsilon)^i \,(0.5 - 0.5\epsilon)^{k-i}$$

$$\text{Prob}\,(Y_j = 1) = \sum_{i=0;i \text{ odd}}^{k} \binom{k}{i} (0.5 + 0.5\epsilon)^i \,(0.5 - 0.5\epsilon)^{k-i} \quad \text{and thus}$$

$$\epsilon_k := \text{Prob}\,(Y_j = 1) - \text{Prob}\,(Y_j = 0) = -\sum_{i=0}^{k} (-1)^i \binom{k}{i} (0.5 + 0.5\epsilon)^i \,(0.5 - 0.5\epsilon)^{k-i}$$
$$= -\left(-(0.5 + 0.5\epsilon) + (0.5 - 0.5\epsilon)\right)^k = (-1)^{k+1}\epsilon^k\,. \tag{5.2}$$

Formula (5.2) says that the bias vanishes exponentially fast in the number of XORed bits. On the negative side, the output rate reduces by factor k. The greatest practical significance has the case $k = 2$. In particular,

$$|\text{Prob}\,(Y_j = 1) - 0.5| = 2^{k-1}\epsilon_0^k\,. \tag{5.3}$$

746 [Xoring k non-overlapping bits of a Markov chain] If the random variables $R_1, R_2, \ldots$ form a homogeneous binary-valued ergodic Markov chain, the random variables $Y_1, Y_2, \ldots$ are usually no longer Markovian ($Y_j := R_{k(j-1)+1} + \cdots + R_{kj} (\text{ mod } 2)$ as in par. 745). On the other hand, the random vectors $\vec{R}_1 := (R_1, \ldots, R_k), \vec{R}_2 := (R_{k+1}, \ldots, R_{2k}), \ldots$ are Markovian with a $(2^k \times 2^k)$-transition matrix Q. As for the special case $k = 2$ in [Schi09b], Example 3.31, we obtain a lower entropy bound

$$H(Y_{n+1} \mid Y_n, \ldots, Y_1) \geq H(Y_{n+1} \mid \vec{R}_n, \ldots, \vec{R}_1) = H(Y_{n+1} \mid \vec{R}_n) = H(Y_{n+1} \mid R_{nk}). \tag{5.4}$$

The inequality follows from the fact that Y_j is a function of $\vec{R}_j$, and the Markov property of $\vec{R}_1, \vec{R}_2, \ldots$ and $R_1, R_2, \ldots$ implies the equation signs. As in par. 745 the case $k = 2$ has the greatest practical significance.

747 [Xoring k non-overlapping bits of a Markov chain, ctd.] If all transition probabilities of P are positive, this is the case for Q, too, and the Markov chain $\vec{R}_1, \vec{R}_2, \ldots$ is ergodic with invariant distribution $\nu_{(k)}$. If we assume that the random variables $R_1, R_2, \ldots$ are in an equilibrium state,

then $\vec{R}_1, \vec{R}_2, \ldots$ are in the equilibrium state, too. Furthermore, the random vectors $Y_1, Y_2, \ldots$ are stationarily distributed with distribution η. More precisely,

$$\eta_0 = \sum_{i=0}^{1} \nu_i \sum_{\substack{j_1,\ldots,j_k \\ j_1+\ldots+j_k \equiv 0 \bmod 2}} p_{ij_1} p_{j_1 j_2} \cdots p_{j_{k-1} j_k}\,, \quad \eta_1 = 1 - \eta_0 \tag{5.5}$$

In particular, by (5.4)

$$H(Y_{n+1} \mid Y_n, \ldots, Y_1) \geq H(Y_{n+1} \mid R_{nk}) = \sum_{i=0}^{1} \nu_i H(Y_{n+1} \mid R_{nk} = i)\,. \tag{5.6}$$

[Xoring 2 non-overlapping bits of a Markov chain] In pars. 748 to 749 we take a closer look at the distribution of the random variables $Y_1, \ldots, Y_n$ for $k = 2$. Again, we assume that the random variables $R_1, R_2, \ldots$ form a homogeneous Markov chain with 2×2-transition matrix $P = (p_{ij})_{0 \leq i,j \leq 1}$ with positive transition probabilities. The limiting distribution is given by $\nu = (\nu_0, \nu_1) = \left(\frac{p_{10}}{p_{01}+p_{10}}, \frac{p_{01}}{p_{01}+p_{10}}\right)$. Under the assumption that the Markov chain $R_0, R_1, \ldots$ is in an equilibrium state, we conclude 748

$$\text{Prob}\,(Y_1 = y_1, \ldots, Y_n = y_n) = \sum_{j=0}^{1} \nu_j \,\text{Prob}\,(Y_1 = y_1, \ldots, Y_n = y_n \mid R_0 = j) = \tag{5.7}$$

$$\sum_{j=0}^{1} \nu_j \,\text{Prob}\,(Y_n = y_n \mid Y_1 = y_1, \ldots, Y_{n-1} = y_{n-1}, R_0 = j)\,\text{Prob}\,(Y_1 = y_1, \ldots, Y_{n-1} = y_{n-1} \mid R_0 = j)$$

Exploiting the Markov property of $R_0, R_1, \ldots$, the last but one conditional probability in (5.7) can be expressed as follows:

$$\begin{aligned}
&\text{Prob}\,(Y_n = y_n \mid Y_1 = y_1, \ldots, Y_{n-1} = y_{n-1}, R_0 = j) = \\
&\sum_{i=0}^{1} \text{Prob}\,(Y_n = y_n, R_{2n-2} = i \mid Y_1 = y_1, \ldots, Y_{n-1} = y_{n-1}, R_0 = j) = \\
&\sum_{i=0}^{1} \text{Prob}\,(Y_n = y_n \mid R_{2n-2} = i) \cdot \text{Prob}\,(R_{2n-2} = i \mid Y_1 = y_1, \ldots, Y_{n-1} = y_{n-1}, R_0 = j) = \\
&\sum_{i=0}^{1} (p_{i0} p_{0,y_n} + p_{i1} p_{1,1-y_n}) \cdot \text{Prob}\,(R_{2n-2} = i \mid Y_1 = y_1, \ldots, Y_{n-1} = y_{n-1}, R_0 = j)
\end{aligned} \tag{5.8}$$

[Xoring 2 non-overlapping bits of a Markov chain, special cases] 749
(i) The first special case is given when $p_{00} = p_{10}$ because then the random variables R_j are iid. This special case has already been covered by par. 745.
(ii) Another special case is given when $p_{01} = p_{10}$. Then $\nu = (0.5, 0.5)$, i.e., the random variables $R_0, R_1, \ldots$ are unbiased but dependent. The equality $p_{01} = p_{10}$ implies $p_{00} = p_{11}$, and thus $p_{ik} = p_{1-i,1-k}$. Thus, the term $(p_{i0} p_{0,y_n} + p_{i1} p_{1,1-y_n})$ does not depend on i, simplifying (5.8) to

$(p_{00}p_{0,y_n} + p_{01}p_{0,y_n}) = p_{0,y_n}$. By induction, we obtain from (5.7) and (5.8)

$$\mathrm{Prob}\left(Y_1 = y_1, \ldots, Y_n = y_n\right) = \sum_{j=0}^{1} \nu_j \,\mathrm{Prob}\left(Y_1 = y_1, \ldots, Y_n = y_n \mid R_0 = j\right) =$$

$$p_{0,y_n} \sum_{j=0}^{1} \nu_j \,\mathrm{Prob}\left(Y_1 = y_1, \ldots, Y_{n-1} = y_{n-1} \mid R_0 = j\right) = p_{0,y_n} \,\mathrm{Prob}\left(Y_1 = y_1, \ldots, Y_{n-1} = y_{n-1}\right) =$$

$$p_{0,y_n} p_{0,y_{n-1}} \cdots p_{0,y_1} \tag{5.9}$$

Hence, the random variables $Y_1, Y_2, \ldots$ are iid with $\mathrm{Prob}(Y_j = i) = p_{0i}$. Interestingly, unlike in special case (i) where XORing non-overlapping pairs of raw random number bits works very well, in special case (ii) it does not. Compared to the Markov chain $R_0, R_1, \ldots$, the entropy per bit does not increase; this refers to both Shannon entropy and min-entropy; cf. pars. 686 and 537. Note: It should be noted that the special case (ii) can be verified directly without considering the complicated formulae in par. 748. In fact, the probability that $R_{2n} = R_{2n-1}$, or equivalently, that $Y_n = 0$, is $p_{00} = p_{11}$, regardless of $y_1, \ldots, y_{n-1}$. This in turn implies

$$\mathrm{Prob}\left(Y_1 = y_1, \ldots, Y_n = y_n\right) = p_{0,y_n} \,\mathrm{Prob}\left(Y_1 = y_1, \ldots, Y_{n-1} = y_{n-1}\right). \tag{5.10}$$

The rest follows by induction.

750 Of course, the compression functions are not limited to XORing single bits. Another approach is to group the sequence $R_1, R_2, \ldots$ into non-overlapping blocks of t bits; $\vec{R}_1 := (R_1, \ldots, R_t), \vec{R}_2 := (R_{t+1}, \ldots, R_{2t}), \ldots$ and to interpret their realizations as values in a finite group G with 2^t elements and group operation. For example, algorithmic post-processing could given by $\vec{Y}_j := \vec{R}_{2j-1} + \vec{R}_{2j} (\mathrm{mod}\, 2^t)$. Applying this group operation provides a stronger mixture of the particular components than the bitwise XOR operation.

751 A straight-forward example is $G = \mathbb{Z}_{2^t}$ together with the addition modulo 2^t. For the special cases $t = 4$ and $t = 8$, also $G = Z_{2^t}^*$ together with the multiplication modulo 2^t as group operation is an example, because 17 and 257 are prime. The value is identified with 2^t.

752 The pars. 753 to 758 may in particular be useful when the noise source generates k-bit raw random number vectors $\vec{r}_1, \vec{r}_2, \ldots$.

753 Assume that $\Omega_1 = \{x_1, \ldots, x_n\}$, $\Omega_2 = \{y_1, \ldots, y_n\}$, and $\Omega = \{z_1, \ldots, z_n\}$. The random variables X and Y are independent and take on values in Ω_1 and Ω_2, respectively, with probabilities $\mathrm{Prob}(X = x_j) = p_j$ and $\mathrm{Prob}(Y = y_j) = q_j$ for $j = 1, \ldots, n$. Without loss of generality we may assume that $p_1 \leq \ldots \leq p_n$ and $q_1 \leq \ldots \leq q_n$. (Otherwise, relabel the elements of Ω_1 and Ω_2.) Furthermore, $f\colon \Omega_1 \times \Omega_2 \to \Omega$ and $Z = f(X, Y)$.

754 Assume further that the mapping $f\colon \Omega_1 \times \Omega_2 \to \Omega$ is invertible in the second argument (i.e., for each fixed first argument). Hence, for each pair $(i, j) \in \{0,1\}^n \times \{0,1\}^n$, there exists a unique index k such that $z_i = f(x_j, y_k)$. In other words: For each $i \leq n$ the function f generates a permutation π_i of $\{1, \ldots, n\}$ that is given by $z_i = f\left(x_j, y_{\pi_i(j)}\right)$. Since X and Y are independent

$$\mathrm{Prob}\left(Z = z_i\right) = \sum_{j=1}^{n} \mathrm{Prob}\left(X = x_j, Y = y_{\pi_i(j)}\right) = \sum_{j=1}^{n} p_j q_{\pi_i(j)} \quad \text{for } 1 \leq i \leq n\,. \tag{5.11}$$

Applying the re-arrangement inequality [HaLP34] to the right-hand side of (5.11) provides the inequality 755

$$\sum_{j=1}^{n} p_j q_{n-j+1} \leq \text{Prob}\,(Z = z_i) = \sum_{j=1}^{n} p_j q_{\pi_i(j)} \leq \sum_{j=1}^{n} p_j q_j. \tag{5.12}$$

In pars. 756 to 758 we additionally assume that $\Omega_1 = \Omega_2$ and that $p_j = q_j$ for $j = 1, \ldots, n$, i.e., that X and Y are identically distributed. Inequality (5.12) implies 756

$$H_{\min}(Z) = -\log_2\left(\max\{\text{Prob}\,(Z = z_i) \mid i = 1, \ldots, n\}\right) \leq -\log_2\left(\sum_{j=1}^{n} p_j^2\right) = H_2(X) \tag{5.13}$$

$$\text{with equality if } \max\{\text{Prob}\,(Z = z_i) \mid i = 1, \ldots, n\} = \sum_{j=1}^{n} p_j^2 \tag{5.14}$$

which ties the collision entropy of X to the min-entropy of $Z = f(X, Y)$.

Assume that $\Omega_1 = \Omega_2 = \{0,1\}^k$ and $f(x, y) = x \oplus y$ (bitwise XOR operation). Then $z = f(x, y) = \vec{0}$ iff $x = y$, and thus 757

$$\max\left\{\text{Prob}\,(Z = z_i) \mid i = 1, \ldots, 2^k\right\} = \text{Prob}(Z = \vec{0}) = \sum_{j=1}^{2^k} p_j^2 \tag{5.15}$$

Finally, (5.13) and (5.14) imply

$$H_{\min}(Z) = -\log_2\left(\sum_{j=1}^{2^k} p_j^2\right) = -\log_2\left(\text{Prob}\left(Z = \vec{0}\right)\right) = H_2(X)\,. \tag{5.16}$$

Equation (5.16) simplifies the estimation of the min-entropy of Z to the estimation of the probability $\text{Prob}\left(Z = \vec{0}\right)$. This may be interesting for noise sources that output independent k-bit raw random number vectors. Alternatively, portions of k bits may be taken from stationary binary-valued raw random numbers such that consecutive vectors may be assumed to be independent. In particular, (5.16) suggests a simple online test that checks the proportion of pairs of k-bit input vectors that are identical. (This is equivalent to counting the number of 0's of the output sequence.) 758

5.1.2 Von Neumann unbiasing

Von Neumann unbiasing works asynchronously, i.e., it receives pairs of binary-valued raw random numbers $\vec{r}_k = (r_{2k}, r_{2k+1})$ as input but does not generate internal random number bit for all 759 input pairs. More precisely, let

$$r'_k := \begin{cases} 0 & \text{if } \vec{r}_k = (0,1) \\ 1 & \text{if } \vec{r}_k = (1,0) \\ o & \text{if } \vec{r}_k = (0,0) \\ o & \text{if } \vec{r}_k = (1,1) \end{cases} \tag{5.17}$$

The internal random number bits $y_1, y_2, \ldots$ are given by the concatenation of all $r'_k \in \{0,1\}$.

760 It is well-known (and easy to prove) that the internal random numbers $Y_1, Y_2, \ldots$ are iid $B(1, 0.5)$-distributed if the random variables $R_1, R_2, \ldots$ are iid $B(1,p)$ distributed. This means that the von Neumann unbiasing algorithm removes the bias completely.

761 The main problem in the context of PTRNG evaluation is the verification that the random variables $R_1, R_2, \ldots$ are (at least almost) iid.

762 However, there are further disadvantages: The output rate drops down to $p(1-p) \leq 0.25$ of the input rate ($p = \text{Prob}(R_1 = 1)$), and from a technical point of view, it can cause problems because it is impossible to guarantee the generation of an internal random number bit within a fixed time interval.

763 [Generalized von Neumann unbiasing] The technique described above can be generalized to transform pairs of integer values $\vec{r}_k = (r_{2k}, r_{2k+1})$ into internal random number bits by the following rule.

$$r'_k := \begin{cases} 0 & \text{if } \vec{r}_k = (r_{2k} < r_{2k+1}) \\ 1 & \text{if } \vec{r}_k = (r_{2k} > r_{2k+1}) \\ o & \text{if } \vec{r}_k = (r_{2k} = r_{2k+1}) \end{cases} \tag{5.18}$$

5.1.3 Thinning out

764 Assume that the sequence $R_1, R_2, \ldots$ has only a small bias but non-negligible dependencies. A straight-forward strategy is to use only each t^{th} raw random number, i.e., $Y_n := R_{nt}$. This should reduce the dependencies.

765 Assume that $R_1, R_2, \ldots$ form a homogeneous ergodic Markov chain on the finite state space $\Omega_R := \{\omega_1, \ldots, \omega_k\}$ with state transition matrix P (typically, $k = 2$). Then $Y_1, Y_2, \ldots$ also forms a homogeneous ergodic Markov chain but with state transition matrix P^t. The rows of the powers $P, P^2, \ldots$ converge exponentially fast to the limiting distribution ν. Thus $H(\nu)$ (or $H_{min}(\nu)$, respectively) provide upper entropy bounds.

5.2 Evaluation of DRNGs: Miscellaneous aspects

766

In this section we discuss several pure DRNG and hybrid DRNG designs. The focus lies on the exemplary verification of requirements of functionality classes DRG.2, DRG.3, and DRG.4. Furthermore, we also illustrate some pitfalls that can occur when a DRNG was designed carelessly. These may serve as a warning to evaluators.

In this section we focus on algorithmical aspects. We do not cover entropy issues associated with 767
seeding procedures and reseeding procedures.

The Hash_DRBG and the HMAC_DRBG from the NIST document [SP800-90A] are analyzed 768
in Sect. 5.3, which is a section of its own.

5.2.1 AES in OFB mode

In this subsection we analyze a simple pure DRNG design. We illustrate how typical proofs 769
can be organized. We show that this DRNG is compliant to functionality class DRG.2 but not to DRG.3. In particular, the DRNG provides backward secrecy and forward secrecy but not enhanced backward secrecy.

The 'core' is the block cipher AES-256. The DRNG calls the AES-256 cipher once during each 770
iteration (full OFB mode). Its plaintext and ciphertext space are given by $S_B := \{0,1\}^{128}$ while $S_K = \{0,1\}^{256}$ denotes the key space. For simplicity, we further assume that this DRNG only accepts requests of at most 128 bits, the bit length of a single internal random number. Below, we formulate the describing 9-tuple $(S, S_{req}, R, A, I, \phi, \phi_{req}, \phi_0, \psi)$; cf. (3.1).

[describing 9-tuple] The components of the 9-tuple are as follows: $S = S_B \times S_K$, $S_{req} = S$, 771
$R = S_B$, $A = \{o\}$ (no external input, pure DRNG), $I = \{1, \ldots, 128\}$ (requests have length ≤ 128 bits), $\phi\colon S \to S, \phi(r,k) := (\text{AES-256}(r,k), k)$ (state transition function), $\phi_{req}\colon S \to S_{req}, \phi_{req}(s) = s$ (identity mapping), $\phi_0\colon S_{req} \to S_{req}$ (the definition of ϕ_0 is irrelevant because a request comprises only one internal random number; cf. par. 141), and $\psi\colon S \to R, \psi(r,k) := r$ (output function).

We assume that a seed material string of $128 + 256 = 384$ bits is generated by a PTRNG that is 772
compliant with PTG.2 or PTG.3, or by an NPTRNG compliant with class NTG.1. The seeding procedure and the reseeding procedure are rather simple.

The seed material string equals the first internal state $s_1 := (r_1, k)$ of the DRNG. In terms of 773
the seeding procedure describing 4-tuple (SM, PS, S, ϕ_{seed}) (cf. (3.3)), the seeding procedure reads as follows. $SM = \{0,1\}^{384}$, $PS = \{o\}$, $\phi_{seed}\colon SM \times PS \to S, \phi_{seed}(s', o) = s'$ (seeding procedure, projection onto the first component).

For the reseeding procedure a seed material string of 384 bits is generated by a PTRNG that is 774
compliant with PTG.2 or PTG.3, or by an NTRNG compliant with class NTG.1. In terms of the seeding procedure describing 4-tuple (SM, PS, S, ϕ_{seed}) (cf. (3.4)), the reseeding procedure reads as follows: $SM = \{0,1\}^{384}$, $PS = \{o\}$, $\phi_{reseed}\colon S \times SM \times PS \to S, \phi_{seed}(s, s', o) = s + s' \bmod 2$ (reseeding procedure, XORing the reseed string onto the internal state).

775

The second component of the internal state $S = S_B \times S_K$ remains constant and serves as a long-term key for AES-256. The output function is the projection onto the first component of the internal state. Since each random number reveals the current first component, the first 128 bits of the internal state ($= S_B$) are potentially public. Thus, the unknown part of the internal state comprises 256 bits; cf. par. 781.

776 Now assume that an adversary knows the random numbers $r_i, \ldots, r_j$. The task is to determine or to guess the successor r_{j+1} or the predecessor r_{i-1} of this subsequence.

777 [forward secrecy] The subsequence $r_i, \ldots, r_j$ can be written in the following form: $r_{i+1} = \text{AES-256}(r_i, k), r_{i+2} = \text{AES-256}(r_{i+1}, k), \ldots, r_j = \text{AES-256}(r_{j-1}, k)$. If this information would suffice to determine $r_{j+1} = \text{AES-256}(r_j, k)$, this would mean that a chosen plaintext attack on AES-256 (for the (plaintext / ciphertext) pairs $(r_i, r_{i+1}), \ldots, (r_{j-1}, r_j)$) would be feasible. However, the cryptographic community has analyzed the AES cipher for more than two decades, and no such cryptanalytic attack has been found. This common knowledge about AES-256 can be used to conclude that the DRNG has forward secrecy.

778 [forward secrecy] Par. 777 provides a typical security proof for DRNGs. The desired security property of the DRNG is traced back to established properties of the cryptographic primitives.

779 [backward secrecy] The proof of backward secrecy is analogous to the proof in par. 777. We express the known subsequence $r_i, \ldots, r_j$ as follows: $r_{j-1} = (\text{AES-256})^{-1}(r_j, k)$, $r_{j-2} = (\text{AES-256})^{-1}(r_{j-1}, k), \ldots, r_i = (\text{AES-256})^{-1}(r_{i+1}, k)$. A successful attack on $r_{i-1} = (\text{AES-256})^{-1}(r_i, k)$ would imply a chosen ciphertext attack on $(\text{AES-256})^{-1}$, the decryption function of AES-256. Since no such attack is known, we conclude that the DRNG has backward secrecy.

780 [enhanced backward secrecy] Obviously, the DRNG does not have enhanced backward secrecy. If an adversary learns the internal state $s_n = (r_n, k)$, he obtains the preceding internal states $s_{n-1}, s_{n-2} \ldots$ (and the preceding random numbers $r_{n-1}, r_{n-2} \ldots$).

781 As a by-product of the security proofs in pars. 777 (forward security) and 779 (backward security), we conclude that the set of effective internal states equals the key space $S_K = \{0,1\}^{256}$, due to the generally accepted properties of AES.

782 In the previous paragraphs we have proved that the DRNG fulfills several requirements of functionality class DRG.2. In particular, this refers to requirements DRG.2.1 (pars. 772, 773, 774), DRG.2.2 (par. 770), DRG.2.3 (par. 781), DRG.2.4 (pars. 772, 773, 774, 781), DRG.2.5 (par. 777), and DRG.2.6 (par 779). Moreover, DRG.2.7 does not apply because the DRNG is a pure DRNG, and thus, DRG.2.6 is also fulfilled. The state transition function ϕ is cryptographic, and thus DRG.2.8 is fulfilled, too. Since no statistical weaknesses of AES-256 are known, the evaluator might argue that requirement DRG.2.9 is fulfilled on the basis of theoretical considerations.

783 By par. 782 the DRNG is compliant with functionality class DRG.2. Yet the DRNG is not compliant with functionality class DRG.3 because of the missing enhanced backward secrecy (par. 780).

5.2.2 Pure and hybrid DRNGs and a (too) simple state transition function

In this subsection several simple DRNG designs are considered. In pars. 785 to 796 the DRG.2-compliance of a pure DRNG design is verified, and then different extensions to hybrid DRNG designs are discussed. Pars. 797 to 799 underline that a (too) simple state transition function may ruin the security if the adversary is able to control a single additional input value. We offer bug fixes but also sketch an instructive pitfall. 784

For simplicity, we assume that all DRNGs discussed in this subsection accept only requests, whose bit length is $\leq$ the bit length of an internal random number. 785

In par. 787 below, we formulate the describing 9-tuple $(S, S_{req}, R, A, I, \phi, \phi_{req}, \phi_0, \psi)$ for the 'pure' version of the DRNG; cf. (3.1). 786

[pure DRNG, describing 9-tuple] For the pure DRNG the components of the 9-tuple are as follows: $S = S_{req} = Z_{2^{512}}$, $R = \{0,1\}^{256}$, $A = \{o\}$ (no external input, pure DRNG), $I = \{1,\ldots,256\}$ (requests have length ≤ 256 bits), $\phi\colon S \to S, \phi(s) := (s+1 \bmod 2^{512})$ (state transition function, modular incrementation by 1), $\phi_{req}\colon S \to S_{req}, \phi_{req}(s) = s$, $\phi_0\colon S_{req} \to S_{req}$ (the definition of ϕ_0 is irrelevant because a request requires only one internal random number), and $\psi\colon S_{req} \to R, \psi(s_{req}) := \text{SHA-256}(s_{req})$ (output function). 787

We assume that a seed material string of 512 bits is generated by a TRNG that is compliant with functionality class PTG.2, PTG.3, or NTG.1. The seeding procedure and the reseeding procedure are rather simple. 788

The seed material string equals the first internal state s_1 of the DRNG. In terms of the seeding procedure describing 4-tuple (SM, PS, S, ϕ_{seed}) (cf. (3.3)), the seeding procedure can be described as follows. $SM = \{0,1\}^{512}$, $PS = \{o\}$, $\phi_{seed}\colon SM \times PS \to S, \phi_{seed}(s', o) = s'$ (seeding procedure, projection onto the first component). 789

For the reseeding procedure a seed material string of 512 bits is generated by a TRNG that is compliant with PTG.2, PTG.3, or NTG.1. In terms of the seeding procedure describing 4-tuple (SM, PS, S, ϕ_{seed}) (cf. (3.4)), the reseeding procedure reads as follows: $SM = \{0,1\}^{512}$, $PS = \{o\}$, $\phi_{reseed}\colon S \times SM \times PS \to S, \phi_{seed}(s, s', o) = s \,\text{XOR}\, s'$ (reseeding procedure, bitwise addition mod 2). 790

Due to the one-way property of SHA-256, we may assume that the internal state S equals the effective internal state. Thus, the effective internal state comprises 512 bits. 791

[backward secrecy and forward secrecy] The subsequence $r_i, \ldots, r_j$ can be expressed as $r_i = \text{SHA-256}(s_i), r_{i+1} = \text{SHA-256}(s_i + 1 \bmod 2^{512}), \ldots, r_j = \text{SHA-256}(s_i + j - i \bmod 2^{512})$. The task of an adversary would be to exploit this information to determine $r_{j+1} = \text{SHA-256}(s_i + j - i + 1 \bmod 2^{512})$ (forward secrecy) or $r_{i-1} = \text{SHA-256}(s_i - 1 \bmod 2^{512})$ (backward secrecy). 792

[backward secrecy and forward secrecy] If an adversary could determine any internal state, this would violate the one-way property of SHA-256. Similarly, the assumption that an adversary would be able to determine r_{i-1} or r_{j+i+1} only on the basis of $r_i, \ldots, r_{i+j}$ and relations between 793 pre-images would also contradict the common knowledge about SHA-256. In particular, it can be assumed that the DRNG has forward secrecy and backward secrecy.

794 Yet this DRNG does not provide enhanced backward secrecy. If an adversary would learn the internal state s_n, he could easily obtain the preceding internal states $s_{n-1} \equiv s_n - 1 \bmod 2^{512}$, $s_{n-2} \equiv s_n - 2 \bmod 2^{512}, \ldots$ (and by this, the preceding random numbers $r_{n-1}, r_{n-2} \ldots$).

795 [pure DRNG] In the previous paragraphs we have proved that the pure DRNG fulfills several requirements of functionality class DRG.2. In particular, this refers to requirements DRG.2.1 (pars. 788, 789, 790), DRG.2.2 (par. 785), DRG.2.3 (par. 791), DRG.2.4 (pars. 788, 789, 790, 791), DRG.2.5 (par. 793), and DRG.2.6 (par. 793). Moreover, DRG.2.7 does not apply because the DRNG is a pure DRNG, and thus DRG.2.7 is also fulfilled. The output function ψ is cryptographic, and thus DRG.2.8 is fulfilled, too. Since no statistical weaknesses of the SHA-256 are known, the evaluator can argue that requirement DRG.2.9 is fulfilled on the basis of theoretical considerations.

796 [pure DRNG] By par. 795 the pure DRNG is compliant with functionality class DRG.2. The DRNG is not compliant with functionality class DRG.3 because of the missing enhanced backward secrecy (par. 794).

797 [hybrid DRNG, inadequate design] In this paragraph the pure DRNG is extended to a hybrid DRNG design that allows additional input. For this, we set $A = \{0,1\}^{512} \cup \{o\}$ and replace the state transition function ϕ and the function ϕ_{req} by

$$\phi_{req}(s,a) = \begin{cases} s \bmod 2^{512} & \text{if } a = o \\ s + a \bmod 2^{512} & \text{if } a \neq o\,, \end{cases} \tag{5.19}$$

$$\phi(s,a) = \begin{cases} s + 1 \bmod 2^{512} & \text{if } a = o \\ s + a + 1 \bmod 2^{512} & \text{if } a \neq o\,, \end{cases} \tag{5.20}$$

798 [successful attack] If the additional input values are generated by a strong TRNG, no problems should occur (512-bit strings are interpreted as binary representations of 512-bit integers). However, if the adversary is able to control a single additional input value, he is able to set the future random numbers. More precisely: Assume that $a_{j-k}, \ldots, a_{j-1}$ describe the additional inputs at time $j-k, \ldots, j-1$. If at time j the adversary inputs $a_j = 2^{512} - a_{j-k} - \cdots - a_{j-k-1} - k$ for some $k < j$, then $r_j = r_{j-k}$ and $s_{j+1} = s_{j-k+1}$. If $a_{j-k+1} = a_{j+1}, \ldots, a_{j-1} = a_{j+k-1}$ (e.g., all $= o$), then $r_j = r_{j-k}, r_{j+1} = r_{j-k+1}, \ldots, r_{j+k-1} = r_{j-1}$, which means that the DRNG repeats the last k internal random numbers.

799 The hybrid design from par. 797 violates requirement DRG.2.7. Thus, this hybrid DRNG is not DRG.2-compliant. This is an example where a hybrid DRNG is weaker than its pure DRNG version. In particular, this observation justifies requirement DRG.2.7 (resp. DRG.3.8, resp. DRG.4.8).

800 [hybrid DRNG, healed design] In this paragraph we fix the buggy design from par. 797. We

modify ϕ_{req} and ϕ to

$$\phi_{req(H)}(s,a) = \begin{cases} s & \text{if } a = o \\ s + a \bmod 2^{512} & \text{if } a \neq o\,, \end{cases} \tag{5.21}$$

$$\phi_{(H)}(s,a) = \begin{cases} \text{SHA-512}(s) & \text{if } a = o \\ \text{SHA-512}(s + a \bmod 2^{512}) & \text{if } a \neq o\,, \end{cases} \tag{5.22}$$

where '(H)' stands for 'healed' to avoid confusion. The (generally accepted) security properties of SHA-512 prevent even an adversary with full control over the additional input data from selecting values that affect the internal state S in a targeted way. By the same argument, due to the properties of the hash function SHA-256, an adversary is not able to influence the internal random number of the current request in a targeted way. Using the generally accepted security properties of SHA-512 and SHA-256, one can show that the DRNG has backward secrecy, forward secrecy, and enhanced backward secrecy. In particular, this hybrid DRNG is compliant with functionality class DRG.3.
Note: For an evaluation the argument should be more detailed.

[instructive pitfall] Depending on the device the implementation of two different hash functions may be too expensive. This problem could be solved by replacing $R = \{0,1\}^{256}$ by $R = \{0,1\}^{512}$ and the output function $\psi = \text{SHA-256}$ by $\psi_* = \text{SHA-512}$. However, this design is terribly weak since $r_n = \psi_*(\phi_{req(H)}(s_n,a))) = \text{SHA-512}(s_n + a \bmod 2^{512}) = \phi_{(H)}(s_n,a) = s_{n+1}$. Knowledge of a single internal random number reveals the next internal state, and thus (provided that an adversary knows the future additional input data) all future internal random numbers. 801

Par. 801 emphasizes that it does not suffice that the functions ϕ and ψ are individually strong. Their interaction must be secure, too; cf. par. 803. 802

[hybrid DRNG, another design] We set $S = \{0,1\}^{256}$ and $A = \{0,1\}^*$, and we identify both the state space $S = Z_{2^{256}}$ and S_{req} with $\{0,1\}^{256}$. Moreover, 803

$$\phi_{req(H2)}(s,a) = (s \| 00 \| a) \tag{5.23}$$
$$\psi_{(H2)}(s_{req}) = \text{SHA-256}(s_{req}) \tag{5.24}$$
$$\phi_{(H2)}(s,a) = \text{SHA-256}(s \| 11 \| a) \tag{5.25}$$

The strings '00' and '11' ensure that the arguments of the state transition function ϕ and output function ψ are different.

The describing 9-tuple reads as follows: $S = \{0,1\}^{256}$, $S_{req} = \{0,1\}^{256}$, $R = \{0,1\}^{256}$, $A = \{0,1\}^*$, $I = \{1,\ldots,256\}$, $\phi_{(H2)}$ (state transition function, cf. (5.25)), $\phi_{req(H2)}$ (cf. (5.23)), $\phi_0\colon S_{req} \to S_{req}, \phi_0(s_{req}) = s_{req}$ (the definition of ϕ_0 is irrelevant because a request requires only one internal random number), and $\psi_{(H2)}\colon S \to R$ (output function, cf. (5.24)). 804

[DRG.3-compliant DRNG with bijective output function] First, $p \in \{0,1\}^{128}$ is a constant. The additional input $a \in A$ is a bit string of length $\ell \in \{0,\ldots,128\}$, and $\iota(a) := (a, 0\ldots,0) \in \{0,1\}^{128}$, i.e., $\iota(\cdot)$ extends a to a 128-vector by appending 0's to the right. In particular, if $a = \emptyset$ then $\iota(a) = (0,\ldots,0)$. The output function $\psi_{(b)}\colon \{0,1\}^{256} \times A \to \{0,1\}^{128}$ is given by $\psi_{(b)}(s,a) := \text{AES-256}(p \oplus \iota(a), s)$. The key s is the value of current internal state. The state transition function $\phi_{(b)}\colon \{0,1\}^{256} \times A \to \{0,1\}^{256}$, $\phi_{(b)}(s,a) := \text{SHA-256}(s \| p \oplus \iota(a))$. Following 805 par. 785 we assume that requests are limited to 128 bits, which is the bit length of a single internal random number).

806 The describing 9-tuple reads as follows: $S = \{0,1\}^{256}$, $S_{req} = \{0,1\}^{256} \times \{0,1\}^{128}$, $R = \{0,1\}^{128}$, $A = \{a \in \{0,1\}^* \mid 0 \le |a| \le 128\}$, $\phi_{(b)}$ (state transition function), $\phi_{req(b)} \colon S \times A \to S_{req}$, $\phi_{req(b)}(s,a) := (s, \iota(a))$, $\phi_0 \colon S_{req} \to S_{req}$ (the definition of ϕ_0 is irrelevant because a request requires only one internal random number), and $\psi_{(b)} \colon S_{req} \to R$ (output function).

807 [DRG.3-compliant DRNG with bijective output function, ctd.] We first note that the DRNG described in par. 805 is compliant to class DRG.3: Both the state transition function $\phi_{(b)}$ and the output function $\psi_{(b)}$ are cryptographic. Finding predecessors or successors to a given sequence of internal random numbers $r_i, \ldots, r_j$ would require that an adversary was able to mount a successful chosen-plaintext attack or chosen-ciphertext attack on AES-256, which is not considered practically feasible; cf. Subsect. 5.2.1. The one-way property of the state transition function $\phi_{(b)}$ ensures enhanced backward secrecy.

808 Interestingly, for each fixed internal state s the mapping $\chi_s \colon \{0,1\}^{128} \to \{0,1\}^{128}$, $\chi_s(a) :=$ AES-256$(p \oplus \iota(a), s)$ is bijective. In the context of functionality class PTG.3 assume that a PTRNG supplies intermediate random numbers of length 128 bits. Then $\iota(a) = a$, and the output function is bijective for each value of the internal state, by this maintaining the entropy of the intermediate random numbers.

5.2.3 One-way functions derived from the AES block cipher

809 Assume that $\mathrm{Enc}(\cdot,\cdot) \colon \{0,1\}^t \times \{0,1\}^t \to \{0,1\}^t$ denotes an (ideal) block cipher for which the block length and key length are t bits. Then $\chi \colon \{0,1\}^m \times \{0,1\}^m \to \{0,1\}^m$, $\chi(m,k) := \mathrm{Enc}(m,k) \oplus m$ defines a one-way compression function. In one form or another, this idea is used in well-known constructions such as Davies–Meyer, Matyas–Meyer–Oseas, and Miyaguchi–Preneel.

810 For resource-constrained devices such as smart cards, designs of one-way compression functions that use AES (or more generally, a widely recognized block cipher) can be an alternative to the use of dedicated hash functions. Such constructs are allowed, in priciple. The applicant has to give evidence that the class requirements are fulfilled.

5.3 NIST Approved Designs [SP800-90A]: Conformity analysis with regard to DRG.3 and DRG.4

811 The document [SP800-90A] specifies three NIST-approved DRNG designs, the Hash_DRBG, the HMAC_DRBG, and the CTR_DRBG. These DRNGs are based on hash functions (Hash_DRBG and HMAC_DRBG) and block ciphers (CTR_DRBG).

812 Figure 11 illustrates the generic design of these DRBGs. The meaning of the components will become clear in the subsections below.

813

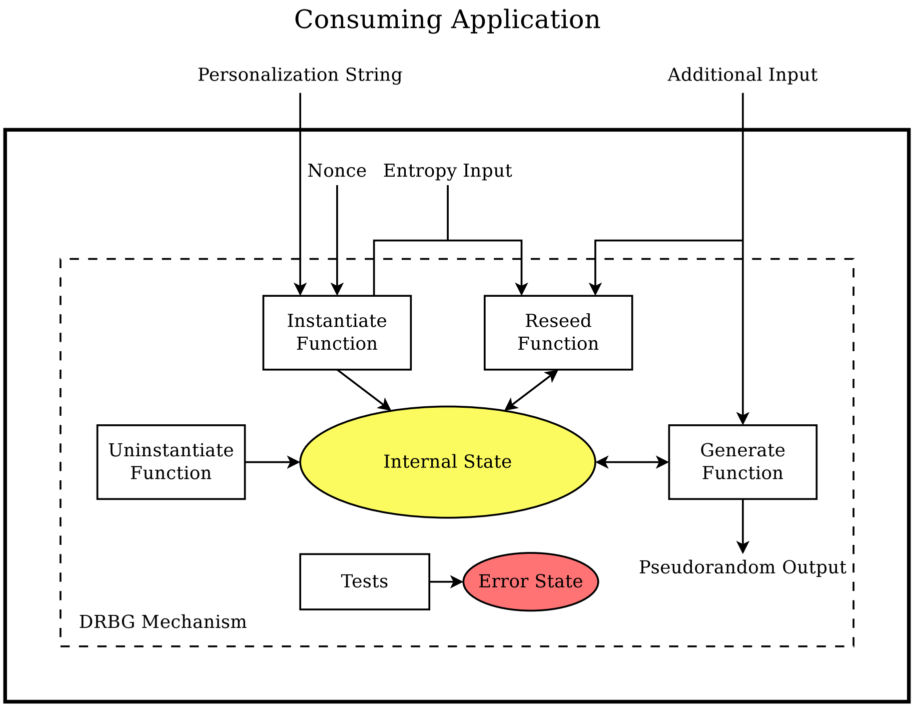

Figure 11: DRBG functional model of the NIST-approved DRBGs; source: [SP800-90A], Sect. 7, Figure 1

Subsect. 5.3.1 provides a conformity proof for the Hash_DRBG to the algorithmic requirements of functionality class DRG.3. The permitted hash algorithms are listed in par. 820.

Subsect. 5.3.2 provides a conformity proof for the HMAC_DRBG to the algorithmic requirements of functionality class DRG.3. The permitted hash algorithms are listed in par. 897. In central parts this conformity proof refers to [Kels23]. 814

An applicant for a certificate can refer to Subsect. 5.3.1, or to relevant paragraphs. No further proof needs to be supplied that (correct) implementations of the NIST-approved designs will conform to the functionality classes hereinafter indicated. The resistance of the implementation against attacks, for instance, is part of the overall evaluation of the TOE; cf. Sect. 2.1. 815

[Hash_DRBG: Conformity to DRG.3] Par. 820 lists the permitted hash functions. By par. 883 the Hash_DRBG fulfills the algorithmic requirements DRG.3.2, DRG.3.3, DRG.3.5, DRG.3.6, DRG.3.7, DRG.3.8, DRG.3.9 and DRG.3.10. Par. 885 formulates (easy to check) sufficient conditions that DRG.3.1 and DRG.3.4 are fulfilled if the entropy_input is generated by a TRNG that is compliant with class PTG.2, PTG.3 or NTG.1. 816

[Conformity to DRG.4] In addition to the requirements of functionality class DRG.3, compliance to class DRG.4 demands an appropriate calling scheme for high-entropy additional input, for the seeding procedure and / or for the reseeding procedure (DRG.4.10). According to requirements DRG.4.1 and DRG.4.10, the seed material for the seeding procedure and the reseeding procedure 817 and (if applicable) the high-entropy additional input (that shall ensure enhanced forward secrecy) shall be generated by a PTRNG. If the PTRNG is compliant with class PTG.2 or PTG.3, this simplifies the verification of requirements DRG.4.4 and DRG.4.10. Par. 887 provides further information.

5.3.1 Security Evaluation of the Hash_ DRBG [SP800-90A]

818 In this subsection we analyze the conformity of the Hash_DRBG to the requirements of functionality class DRG.3. Pars. 883 to 885 summarize the results.

819 For a detailed description of the Hash_DRBG, we refer to [SP800-90A], Subsubsect. 10.1.1.

820 In the following we assume

$$Hash \in \{\mathrm{SHA\text{-}224}, \mathrm{SHA\text{-}512/224}, \mathrm{SHA\text{-}256}, \mathrm{SHA\text{-}512/256}, \mathrm{SHA\text{-}384}, \mathrm{SHA\text{-}512}, \mathrm{SHA3\text{-}224}, \mathrm{SHA3\text{-}256}, \mathrm{SHA3\text{-}384}, \mathrm{SHA3\text{-}512}\}\,. \tag{5.26}$$

The hash function $Hash$ outputs strings of length $outlen \in \{224, 256, 384, 512\}$ as indicated by its name; if the name includes two numbers, the output length is indicated by the second number.

821 If $Hash \in \{\mathrm{SHA\text{-}224}, \mathrm{SHA\text{-}512/224}, \mathrm{SHA\text{-}256}, \mathrm{SHA\text{-}512/256}\}$, then $seedlen = 440$. If $Hash \in \{\mathrm{SHA3\text{-}224}, \mathrm{SHA3\text{-}256}, \mathrm{SHA3\text{-}384}, \mathrm{SHA3\text{-}512}\}$, then $seedlen = 512$. Finally, if $Hash \in \{\mathrm{SHA\text{-}384}, \mathrm{SHA\text{-}512}\}$, then $seedlen = 888$.

822 Remark: In addition to (5.26) [SP800-90A] allows the hash functions SHA-1.

823 Our first goal is to describe the Hash_DRBG by the 9-tuple (par. 139) and by the 4-tuples for the seeding procedure (par. 158) and reseeding procedure (par. 162).

824 In this subsection $S' := \{0,1\}^{seedlen}$. If additions modulo $2^{seedlen}$ are concerned, we tacitly identify S' with $Z_{2^{seedlen}}$.

825 The set of internal states of the Hash_DRBG (denoted as working state in [SP800-90A]) is given by the cartesian product

$$S := S' \times S' \times \mathbb{Z}_{2^{48}}\,. \tag{5.27}$$

Its elements are triples (v, c, rc). The values v and $c = c(v_1)$ are kept secret, while the value of the reseed_counter rc is publicly known. The reseed_counter is initialized by 1 and incremented by 1 after each request. The reseeding procedure is required after at most 2^{48} requests.
Note: Since $c(v_1)$ remains constant within a request, for the sake of readability we briefly write c instead of $c(v_1)$.

826 The product space $S' \times S'$ (cf. (5.27)) is the set of all effective internal states; cf. par. 878.

827

Furthermore,

$$
\begin{aligned}
S_{req} &:= S' && (5.28)\\
A &:= \{0,1\}^* && (5.29)\\
R &:= \{0,1\}^{outlen} && (5.30)\\
I &:= Z_{2^{19}} && (5.31)
\end{aligned}
$$

The bit length of the additional input $a \in A$ is $\leq 2^{35}$. Empty strings are possible.

After each request the internal state S has been updated by the state transition function ϕ. 828

$$
\begin{aligned}
&\phi := \phi_B \circ (\phi_A \times \mathrm{id}) : S \times A \times I \to S \quad \text{with} && (5.32)\\
&\phi_A \colon S \times A \times I \to S, \quad \phi_A(v,c,rc,a,p) := (v + f(v,a) \bmod 2^{seedlen}, c, rc) && (5.33)\\
&\phi_B \colon S \times A \times I \to S, \quad \phi_B(v,c,rc,a,p) := (v + g(v) + c + rc \bmod 2^{seedlen}, c, rc+1) && (5.34)\\
&\text{with } f \colon \{0,1\}^{seedlen} \times \{0,1\}^* \to \{0,1\}^{outlen}, \quad f(v,a) := \begin{cases} Hash(0x02\|v\|a) \text{ if } a \neq \emptyset \\ 0 \text{ if } a = \emptyset \end{cases} && (5.35)\\
&\text{and} \quad g \colon \{0,1\}^{seedlen} \to \{0,1\}^{outlen}, \quad g(v) := Hash(0x03\|v)\,. && (5.36)
\end{aligned}
$$

Actually, in the Hash_DRBG_Generate Process ([SP800-90A], Subsect. 10.1.1.4) the internal state S is processed in three steps, in Step 2 (by ϕ_A), and in Step 5 and Step 6 (by ϕ_B). Step 2 is carried out before the random numbers are generated (Step 3), while Step 5 and Step 6 are performed after the random numbers have been generated. This means that during the request, the internal state assumes an intermediate value $s := \phi_A(s_{old}, a)$. Note that the state transition function $\phi \colon S \times A \times I \to S$ does not depend on the bit length of the request.

From a logical point of view, the internal state (v, c, rc) is updated per request by the state transition function $\phi = \phi_B \circ \phi_A$. Its first component v is updated within each request (by ϕ_A and ϕ_B). The value c is a function of the first internal state after the seeding procedure or the reseeding procedure (cf. pars. 839 and 843). It remains constant until the next reseeding procedure. The request counter rc is initialized by 1 and is increased by 1 after each request. 829

The temporary internal state during a request is generated and updated by 830

$$
\begin{aligned}
&\phi_{req} :=: S \times A \to S_{req} \quad \text{with}\\
&\phi_{req}(v,c,rc,a) := (v + f(v,a) \bmod 2^{seedlen}) && (5.37)\\
&\phi_0 \colon S_{req} \to S_{req}, \quad \phi_0(s_{req}) := s_{req} + 1 \bmod 2^{seedlen} && (5.38)
\end{aligned}
$$

The value s_{req} corresponds to 'data' in the Hashgen process; cf. [SP800-90A], Subsect. 10.1.1.4. Furthermore, s_{req} equals the first component of the current internal state S (after ϕ_A has been applied).

Finally, the output function ψ is defined by 831

$$
\psi \colon S_{req} \to R, \quad \psi(s') := Hash(s') \qquad (5.39)
$$

This completes the specification of the describing 9-tuple.

832

Next, we provide formal descriptions of the seeding procedure and the reseeding procedure.

833 In [SP800-90A], Sect. 10.3.1, the derivation function Hash_{df} is defined. The function Hash_{df} is the 'core' of both the seeding procedure and the reseeding procedure.

834 The derivation function Hash_{df} concatenates the *Hash* values of different input values. In the following we assume that Hash_{df} is a one-way function; see par. 850.

835 The initial internal state is computed from the seed material (denoted by seed_material in [SP800-90A]). For the Hash_DRBG

$$\text{seed\_material} \; = (\text{ entropy\_input } || \text{ personalization\_string }) \tag{5.40}$$

The maximum bit length of both the entropy_input and the personalization_string is 2^{35} [SP800-90A], Table 2. The personalization_string belongs to the set PS. It may contain secret parts but need not.
Note: The definition of seed_material in (5.40) refers to the upcoming version of [SP800-90A]. For the current version of [SP800-90A] the definition reads as follows

$$\text{seed\_material} \; = (\text{ entropy\_input } || \text{ nonce } || \text{ personalization\_string }) \tag{5.41}$$

The nonce may contain entropy but need not. Both nonce and personalization_string belong to the set PS.

836 The security shall be assured by the entropy of the string entropy_input (denoted as 'seed material' in the seed describing 4-tuple from par. 158).

837 In the notation of the 4-tuple which describes the seeding procedure (par. 158)

$$SM = PS = \{0,1\}^* \tag{5.42}$$

838 The nonce and the personalization_string are constructed from the current value of PS.

839 The first initial state (v_1, c, rc) is computed from seed_material via

$$\begin{aligned} &\phi_{seed}\colon \{0,1\}^* \to \{0,1\}^{seedlen} \times \{0,1\}^{seedlen} \times \mathbb{Z}_{2^{48}}, \quad \phi_{seed}(\text{seed\_material}) \\ &:= (v_1 := \text{Hash}_{\text{df}}(\text{seed\_material}, seedlen), c := \text{Hash}_{\text{df}}(0x00 \| v_1, seedlen), 1)\,, \end{aligned} \tag{5.43}$$

i.e., $c = c(v_1)$. The parameter *seedlen* depends on *Hash*.
Note: In the seeding procedure the bit string seed_material (5.40) is the concatenation of values in SM (entropy_input) and PS (nonce and personalization_string).

840 For the reseeding procedure

$$\text{seed\_material} \; = (0x01 \| v \| \text{ entropy\_input } || \text{ additional\_input}) \tag{5.44}$$

The letter v denotes the first component of the internal state before the reseeding procedure. The maximum bit length of both the entropy_input and the additional_input is 2^{35} [SP800-90A], Table 2. The additional_input may be empty. The additional_input belongs to the set PS.

841

The security shall be assured by the entropy of entropy_input.

In the notation of the 4-tuple which describes the reseeding procedure (par. 162) 842

$$SM = PS = \{0,1\}^* \tag{5.45}$$

The first initial state (v_1, c, rc) after the reseeding procedure is computed from seed_material via 843

$$\begin{aligned} &\phi_{reseed}\colon \{0,1\}^* \to \{0,1\}^{seedlen} \times \{0,1\}^{seedlen} \times \mathbb{Z}_{2^{48}}, \quad \phi_{reseed}(\text{seed\_material}) \\ &:= (v_1 := \text{Hash}_{\text{df}}(\text{seed\_material}, seedlen), c := \text{Hash}_{\text{df}}(0x00\|v_1, seedlen), 1) \end{aligned} \tag{5.46}$$

i.e., $c = c(v_1)$. The parameter *seedlen* depends on *Hash*.
Note: In the reseeding procedure the bit string seed_material (5.44) is the concatenation of the first component of the internal state (v) and of values in SM (entropy_input) and PS (nonce and personalization_string).

Below we analyze the conformity of the Hash_DRBG to the requirements of functionality class DRG.3. 844

[Notation] If the request y demands a bit string of length reqbits (by specification reqbits $\leq 2^{19}$, cf. (5.31)) we set 845

$$m_y := \left\lceil \frac{\text{reqbits}}{outlen} \right\rceil \text{ and } u := \text{reqbits} - (m_y - 1)outlen\,. \tag{5.47}$$

The output is

$$\psi(\widetilde{v}_y)\|\psi(\widetilde{v}_y + 1)\| \ldots \|\psi(\widetilde{v}_y + m_y - 2)\|pr_u(\psi(\widetilde{v}_y + m_y - 1))\,. \tag{5.48}$$

It is $\widetilde{v}_y = \phi_{req}(v_y) = s_{req}$, the value of S_{req} after being initialized by ϕ_{req} (or, equivalently, the first component of the internal state S after ϕ_A has been applied). Furthermore, $pr_u(\cdot)$ denotes the projection onto the leftmost u bits. In particular, *Hash* is applied m_y times.

First we investigate forward secrecy (DRG.3.5) and backward secrecy (DRG.3.6). 846

[Notation] To simplify the notation we denote the j^{th} random number (hash value or truncated hash value) of request y by $w_{(y)j}$ (cf. (5.48)). Assume that an adversary knows the random numbers 847

$$w_{(y_1)j}, \ldots, w_{(y_1)m_{y_1}}, w_{(y_1+1)1}, \ldots, w_{(y_2)i}\,. \tag{5.49}$$

His task would be to compute or guess the next random number (forward secrecy) or the random number that precedes this sequence (backward secrecy).

[state transition] Assume that the triple (v_y, c, y) denotes the internal state at the beginning of request y. By (5.32), (5.33), and (5.34) the next internal state (after request y has been completed) equals 848

$$\begin{aligned} &(v_{y+1}, c, y+1) = \phi(v_y, c, y, a_y) = \\ &(v_y + g^*(v_y, a_y) + c + y + f(v_y, a_y) \bmod 2^{seedlen}, c, y+1) \quad \text{where} \\ &g^*\colon \{0,1\}^{seedlen} \times \{0,1\}^* \to \{0,1\}^{outlen}, \quad g^*(v_y, a_y) := g(v_y + f(v_y, a_y) \bmod 2^{seedlen}) \end{aligned} \tag{5.50}$$

849 [Cryptographic assumptions] In pars. 850 to 856 several cryptographic assumptions are formulated and justified, which will be needed below to verify the backward secrecy, forward secrecy and enhanced backward secrecy properties. These assumptions concern $\psi = Hash$ but also the mappings $\text{Hash}_{\text{df}}(\cdot, seedlen), g, f, g^*$, which are derived from $Hash$ and are closely related.

850 [Cryptographic assumptions] The 'core' of the following cryptographic assumptions is (5.51). Since $Hash$ is a (worldwide) recognized hash function this justifies the following assumption

$$\begin{aligned} &\psi(\cdot) = Hash(\cdot) \text{ has the pre-image resistance property.} \\ &\text{It can be modeled by a random mapping.} \end{aligned} \tag{5.51}$$

Pre-image resistance means that it is practically infeasible to determine a pre-image under $Hash$ to a given image value y, i.e., to find any x with $Hash(x) = y$. The second assumption in (5.51) refers to the modeling of $Hash(\cdot)$ in the random oracle model. This means that for given x, the value $Hash(x)$ can be viewed as a realization of a random variable that is uniformly distributed on $\{0,1\}^{outlen}$. Furthermore, the output values of $Hash$ for different input values can be viewed as independent.
Note: If the output length of a function is too small, modeling by a random mapping does not imply the pre-image resistance property.
Note: The justification of the cryptographic assumptions below are in a way redundant as they use 'cryptographic' arguments and the modeling by random mappings.

851 [Cryptographic assumptions] By definition the function $g(\cdot) = \psi(0x03\|\cdot)$ appends the argument to the fixed string '$0x03$' and then applies $Hash$. Thus, by (5.51) we may also assume

$$\begin{aligned} &g\colon \{0,1\}^{seedlen} \to \{0,1\}^{outlen}, \quad g(v) = \psi(0x03\|v) \\ &\text{has the pre-image resistance property. It can be modeled by a random mapping} \end{aligned} \tag{5.52}$$

Rationale: Otherwise, finding a pre-image of $y \in \{0,1\}^{outlen}$ under $Hash$ would not be hard, if there exists a pre-image x of y which starts with the pre-fix byte '$0x03$'. If the bit length of the pre-images was not limited, the pre-image resistance of $g(\cdot)$ would directly follow from the pre-image resistance of $Hash$. In the context of the Hash_DRBG the variable part of the pre-images under $g(\cdot)$ has fixed length $seedlen$, and this restriction does not simplify the problem because $2^{seedlen}$ is large. The second claim follows from the restriction of $Hash$ to the domain $\{0x03\} \times \{0,1\}^{seedlen}$.

852 [Cryptographic assumptions] This assumption considers the restriction of g to a domain interval I_b of length 2^{outlen}. Based on (5.52) we also assume

$$\begin{aligned} &g\colon \{0,1\}^{outlen} \to \{0,1\}^{outlen}, \quad g(v') = \psi(0x03\|v' + b \bmod 2^{seedlen}) \quad \text{for known } b \in \mathbb{Z}_{2^{seedlen}} \\ &\text{has the pre-image resistance property. It can be modeled by a random mapping.} \end{aligned} \tag{5.53}$$

Rationale: For a given $c \in \mathbb{Z}_{2^{outlen}}$, a randomly selected interval $I_b := [b, b + 2^{outlen} - 1]$ contains one pre-image x^* of c (i.e., $g(x^*) = c$) on average. Thus, if finding a pre-image of an restriction $g_{|I_b}$ was easy, an adversary could also find a pre-image of g in (5.52). In fact, he could randomly select an integer $b \in \mathbb{Z}_{2^{seedlen}}$, and with probability $\approx 1 - e^{-1} \approx 0.63$, the interval I_b contains a pre-image of c. In this case the adversary could solve (5.53). The second claim follows from restricting the domain of g to $\{0x03\} \times \{0,1\}^{outlen}$.

853

[Cryptographic assumptions] If $a \neq \emptyset$ then $f(v,a) = \psi(0x02\|v\|a)$. An argument similar to as in par. 851 justifies that

$$\text{If } a \neq \emptyset \text{ then} \quad f\colon \{0,1\}^{seedlen} \times \{0,1\}^{*} \to \{0,1\}^{outlen},\, f(v,a) := \begin{cases} Hash(0x02\|v\|a) \text{ if } a \neq \emptyset \\ 0 \text{ if } a = \emptyset \end{cases}$$

has the pre-image resistance property. It can be modeled by a random mapping. (5.54)

Note: In (5.54) we assume that v is unknown.
(Of course, Assumption (5.54) is not valid for $a = \emptyset$.)

[Cryptographic assumptions] The derivation function $\mathrm{Hash_{df}}(\cdot,\cdot)$ is given by the concatenation of one or several hash values (possibly truncated) whose pre-images only differ in the first byte; cf. [SP800-90A], Subsect. 10.3.1. In both the seeding procedure and the reseeding procedure $\mathrm{Hash_{df}}(\cdot, seedlen)$ is applied twice to compute the first and the second component of the internal state S; cf. (5.40) and (5.44). In these cases at most three hash values are concatenated, depending on *Hash*. Thus, we assume 854

[seeding procedure, reseeding procedure] $\quad \mathrm{Hash_{df}}\colon \{0,1\}^{*} \to \{0,1\}^{seedlen}$
has the pre-image resistance property. It can be modeled by a random mapping(5.55)

Rationale: The following task is not more difficult than finding a pre-image of $\mathrm{Hash_{df}}(\cdot, seedlen)$: An adversary knows three hash values $Hash(x), Hash(x'), Hash(x'')$ where x, x', x'' are in some way related (here: differences in the first input byte). The task is to find any pre-image x^{*} with $Hash(x^{*}) = Hash(x)$. If this was possible this would point to an exploitable correlation of the hash function *Hash* for related input values. In particular, this would exclude the modeling of *Hash* by a random mapping in the random oracle model.

[Cryptographic assumptions] Assume that an adversary knows $k < 2^{60}$ hash values 855

$$\psi(x), \psi(x+\delta_1 (\bmod 2^{seedlen})), \ldots, \psi(x+\delta_{k-1} (\bmod 2^{seedlen})) \quad \text{with } k < 2^{60} \tag{5.56}$$

and the differences δ_j for $1 \leq j \leq k-1$ but not $x \in \mathbb{Z}_{2^{seedlen}}$ (resp., $x \in \{0,1\}^{seedlen}$). In the remainder of this subsection we assume that

To a given $\delta \in \mathbb{Z}_{2^{seedlen}}$ an adversary is not able to calculate $\psi(x+\delta (\bmod 2^{seedlen}))$
unless $x+\delta (\bmod 2^{seedlen}) \in \{x, x+\delta_1 (\bmod 2^{seedlen}), \ldots, x+\delta_{k-1} (\bmod 2^{seedlen})\}$
In particular, the knowledge of (5.56) does not allow finding x. (5.57)

Rationale: This is not a 'standard assumption' on hash functions (recall that $\psi = Hash$), but it is closely related to its pre-image resistance. Assumption (5.57) is similar to Assumption (5.55), although only three calls of *Hash* were considered rather than 2^{60}. On the other hand, $2^{60} \ll 2^{outlen}$ so that a violation of (5.57) would point to a hidden weakness of *Hash*, namely to correlations of *Hash* values for different input values. In particular, this would exclude the modeling of *Hash* by a random mapping. In fact, (5.57) follows when modeling of *Hash* by a random mapping. Altogether, these arguments make Assumption (5.57) rather plausible.

[Cryptographic assumptions] Based on (5.53) we conclude 856

$$\{0,1\}^{outlen} \to \{0,1\}^{outlen}, \quad v' \mapsto v' + b + g(v'+b) \bmod 2^{outlen} \quad \text{for known } b \in \mathbb{Z}_{2^{outlen}}$$

has the pre-image resistance property. It can be modeled by a random mapping. (5.58)

Rationale: Assumption (5.58) is reasonable because the modular addition of the identity mapping should be 'incompatible' with $g(\cdot)$, or more precisely, to its restriction to I_b (cf. 5.53). When modeling as a random mapping, the modular addition of v just means that the values of $g(\cdot)$ are pointwise shifted $\mod 2^{outlen}$, transforming the uniform distribution to the uniform distribution.
Note: Assume that $v + g(v) = c \bmod 2^{seedlen}$. Since $0 \leq g(v) < 2^{outlen}$, we have $v \in [c \bmod 2^{seedlen}, c + 2^{outlen} - 1 \bmod 2^{seedlen}]$ (modular interval). Thus, it suffices to determine $v \bmod 2^{outlen}$, which leads to (5.58).

857 By definition,

$$\text{The output values of } \psi = Hash, g, g^*, f \text{ consist of } outlen \text{ bits.} \tag{5.59}$$

858 When proving the backward secrecy and the forward secrecy properties, we assume that an adversary knows a sequence of internal random numbers, while the (intermediate) internal states (v_y, c, y) are unknown for $y_1 \leq y \leq y_2$ apart from the request counter y. A priori, the values $g^*(v_y, a_y)$ and $f(v_y, a_y)$ are unknown, too. An adversary would gain additional information if he knew the modular differences

$$b_{y+1} := v_{y+1} - v_y \equiv g^*(v_y, a_y) + c + y + f(v_y, a_y) \bmod 2^{seedlen} \quad \text{for all } y = y_1 + 1, \ldots, y_2\,. \tag{5.60}$$

In this case the adversary would know the differences modulo $2^{seedlen}$ between the first components of all (relevant) internal states. (Note that the knowledge of any internal state would allow an easy computation of all successors.) Within each request no more than 2^{19} random bits can be output, which means that no more than $\lceil 2^{19}/outlen \rceil \leq \lceil 2^{19}/224 \rceil < 2^{12}$ hash values are computed.

859 [backward secrecy and forward secrecy, simpler problem] Now consider the following problem: The adversary knows the hash values

$$\begin{aligned} &\psi(\tilde{v}_{y_1} + j), \ldots, \psi(\tilde{v}_{y_1} + 2^{12} - 1), \psi(\tilde{v}_{y_1+1}), \ldots, \psi(\tilde{v}_{y_2} + i) \\ &\text{and the differences } b_{y_1+1}, \ldots, b_{y_2} \quad \text{as defined in (5.60)} \end{aligned} \tag{5.61}$$

but not the pre-images of these hash values. Furthermore, the adversary knows which random numbers belong to which request and that within one request the pre-images form an interval (to be precise, an interval $\bmod 2^{seedlen}$) of length $< 2^{12}$. His task is to determine the successor (forward secrecy), resp. the predecessor (backward secrecy), of the sequence (5.61).

860 [backward secrecy and forward secrecy, simpler problem] Par. 847 formulates the task that an adversary has to solve in order to violate forward secrecy or backward secrecy. If he additionally knows the modular differences b_{y+1} (5.60), his task does not become more difficult. In (5.61) we extended the requests $y_1 + 1$ to $y_2 - 1$ to 2^{12} random numbers per request, which is more information than in par. 847. Furthermore, the random numbers are not truncated, and the adversary knows the differences $b_{y_1+1}, \ldots, b_{y_2}$. Extending the request lengths to their maximum does not affect the following internal states and thus, does not affect future random numbers.

861 [backward secrecy and forward secrecy, simpler problem] Altogether, the tasks in par. 859 cannot be more difficult than the tasks of par. 847 because more information is available. The idea in

analyzing the simpler problem is to get rid of complicated design features and to trace the problem back to the properties of *Hash*.

[backward secrecy and forward secrecy, simpler problem] In pars. 863 to 874 we show that it is not practically feasible to determine the successor or the predecessor of (5.61) or to guess these values with non-negligibly greater probability than without knowledge of the sequence (5.61). This shows that the Hash_DRBG fulfills requirements DRG.3.5 and DRG.3.6. 862

[forward secrecy] We begin with the proof of forward secrecy. The successor of (5.61) either is $\psi(\widetilde{v}_{y_2}+i+1)$ if $i \leq 2^{12}-2$, resp. $\psi(\widetilde{v}_{y_2+1})$ if $i = 2^{12}-1$. We distinguish two cases: 863

Case i) The pre-image of the searched random number, $\psi(\widetilde{v}_{y_2}+i+1)$ or $\psi(\widetilde{v}_{y_2+1})$, is not contained in the set of the Hash pre-images in (5.61). By par. 855, Assumption (5.57), an adversary is not able to exploit the knowledge of the hash values (5.61) and the modular differences between the pre-images to determine the next random number.

Case ii) The pre-image of the requested random number is contained in the set of pre-images of (5.61), i.e., a 'pre-image' hit occurs. Then the adversary's task is easy because both succeeding random numbers coincide. (We neglect exceptional cases where one value of a 'pre-image hit pair' marks the end of a request while the other does not. In particular, we even overestimate the probability of a pre-image hit in the following.) Before we derive an upper bound for the probability that Case ii) occurs, we point out two facts.

[forward secrecy, to par. 863, Case i)] In Case i) by Assumption (5.57) the successor of the subsequence (5.61) cannot be determined even if all the b_y are known. Furthermore, Assumption (5.58) prevents an adversary from determining v_y from the knowledge of $b_{y+1} = b_{y+1}(v_y, a_y)$ (pre-image resistance of g^*). Since the value v_y is unknown, the adversary cannot increase his success rate by a chosen additional input. Furthermore, the adversary cannot determine $v_{y+1} = b_{j+1} + v_y$. 864

[forward secrecy, to par. 863, Case ii)] It is $3 \cdot outlen > seedlen$ for all admissible hash functions. Thus, a collision of triplets $(\psi(\widetilde{v}_{y_2}+i-2), \psi(\widetilde{v}_{y_2}+i-1), \psi(\widetilde{v}_{y_2}+i)) = (\psi(\widetilde{v}_{y_a}+j-2), \psi(\widetilde{v}_{y_a}+j-1), \psi(\widetilde{v}_{y_a}+j))$ for some request y_a is a strong indicator that $\widetilde{v}_{y_2}+i = \widetilde{v}_{y_a}+j$. Except for Hash = SHA-384, even a collision of 2-tuples should suffice. 865

[forward secrecy, to par. 863, Case ii)] Next, we determine an upper bound for the probability that Case ii) occurs. By induction on y equation (5.60) implies 866

$$
\begin{aligned}
v_y \equiv \quad & v_1 + b_2 + \cdots + b_y \equiv \\
& v_1 + \sum_{s=1}^{y-1} g^*(v_s, a_s) + (y-1)c + \frac{y(y-1)}{2} + \sum_{s=1}^{y-1} f(v_s, a_s) (\bmod 2^{seedlen}) \quad (5.62) \\
& \text{for } 1 \leq y \leq 2^{48} \quad (5.63)
\end{aligned}
$$

[forward secrecy, to par. 863, Case ii)] If the pre-image $\widetilde{v}_{y_2}+i+1$ (or analogously, the pre-image $\widetilde{v}_{y_2+1}$) is contained in the set of pre-images in (5.61), briefly denoted as 'pre-image hit' in the 867 following — i.e., Case ii) applies, then $\widetilde{v}_{y_2} + i + 1 \in \widetilde{v}_y + \{0, \ldots, 2^{12} - 1\} \bmod 2^{seedlen}$ for some request $y < y_2$ (necessary condition). This is equivalent to $v_{y_2} - v_y + i + 1 \bmod 2^{seedlen} \in \{0, \ldots, 2^{12} - 1\}$. Substituting v_{y_2} and v_y by (5.62) yields

$$\sum_{s=y}^{y_2-1} g^*(v_s, a_s) + (y_2 - y)c + \frac{y_2(y_2-1)}{2} - \frac{y(y-1)}{2} + \sum_{s=y}^{y_2-1} f(v_s, a_s) + i + 1$$
$$\equiv j \bmod 2^{seedlen} \quad \text{for some } 1 \le y \le y_2 \le 2^{48}, 1 \le j \le 2^{12}\,. \tag{5.64}$$

868 [forward secrecy, to par. 863, Case ii)] A quick look at (5.64) shows that $g^*(v_s, a_s), f(v_s, a_s) < 2^{outlen}$, $y \le 2^{48}$ while $i, j \le 2^{12}$. Consequently,

$$\sum_{s=y}^{y_2-1} g^*(v_s, a_s) + \frac{y_2(y_2-1)}{2} - \frac{y(y-1)}{2} + \sum_{s=y}^{y_2-1} f(v_s, a_s) + i + 1 < 2^{outlen+49} + 2^{95}\,. \tag{5.65}$$

Equation (5.65) says that all terms in (5.64) except $c(y_2 - y)$ are $< 2^{outlen+50}$, and thus only affect the least $outlen + 50$ bits if we neglect carries. The most significant bits of (5.64) are determined by the term $(y_2 - y)c$.

869 [forward secrecy] Of course, for a pre-image hit all $seedlen$ bits must coincide. It is not easy to determine the exact probability for a pre-image hit. Instead, motivated by the observations in par. 868, we determine an upper bound for this probability by considering 'partial' pre-image hits in the least significant $outlen$ bits and in the most significant $seedlen - outlen - 50$ bits.

870 [forward secrecy] In the following we assume that in (5.64) the sum $\sum_{s=y}^{y_2-1}(g^*(v_s, a_s) + f(v_s, a_s)) \bmod 2^{outlen}$ behaves (stochastically) like a realization of a uniformly distributed random variable Z_y on $\{0, 1\}^{outlen}$. This is a mild assumption if we assume that the summands $(g^*(v_s, a_s) + f(v_s, a_s))$ at least approximately behave like realizations of independent random variables (recall that g^* and f assume values in $Z_{2^{outlen}}$).

871 [forward secrecy, pre-image hit mod 2^{outlen}] In the following $A_{y,\ell}$ denotes the event that a pre-image hit in the $outlen$ least significant bits of $y_2 + i + 1$ occurs with some random number, which has been generated in the y^{th} request. Then $\text{Prob}(A_{y,\ell}) < 2^{12-outlen}$ for each $y < y_2$.

872 [forward secrecy, pre-image hit on the most significant bits] A pre-image hit does not only imply a hit of the $outlen$ least significant bits but of all bits, in particular, of the $(seedlen - t)$ most significant bits. To simplify our notation we set $t := outlen + 50$, and $A_{y,m}$ denotes a hit in the $(seedlen - t)$ most significant bits. A pre-image hit for some random number within request y implies

$$\sum_{s=y}^{y_2-1} g^*(v_s, a_s) + (y_2 - y)c + \frac{y_2(y_2-1)}{2} - \frac{y(y-1)}{2} + \sum_{s=y}^{y_2-1} f(v_s, a_s) + i + 1$$
$$\bmod 2^{seedlen} \in [0, 2^{12})\,. \tag{5.66}$$

Furthermore, let $c = c_1 \cdot 2^{outlen} + c_0$ with $c_0 = c \bmod 2^{outlen}$. By par. 868

$$0 \le T := \sum_{s=y}^{y_2-1} g^*(v_s, a_s) + (y_2 - y)c_0 + \sum_{s=y}^{y_2-1} f(v_s, a_s) + \\ + \frac{y_2(y_2-1)}{2} - \frac{y(y-1)}{2} + i < 3 \cdot 2^{outlen+48} < 2^{outlen+50} = 2^t\,, \tag{5.67}$$

and since the left-hand side of (5.66) equals $(y_2 - y)c_1 2^{outlen} + T$, we conclude that

$$(y_2 - y)c_1 \cdot 2^{outlen} \bmod 2^{seedlen} \in [(2^{seedlen-t} - 1)2^t, 2^{seedlen}) \cup [0, 2^{12})\,. \tag{5.68}$$

Equation (5.68) is a necessary condition for a pre-image hit of the most significant $seedlen - t$ bits.

[forward secrecy] We may assume that c_1 is the realization of a random variable C_1 that is uniformly distributed on $Z_{2^{seedlen-outlen}}$. Then the random variable $X := C_1/2^{seedlen-outlen}$ may be modeled as uniformly distributed in the unit interval $[0, 1)$, because we are interested in the probability that X is contained in a 'large' interval whose length is 2^t times $2^{-seedlen+outlen}$. Moreover, since $y_2 - y$ is an integer, the random variable $X_y := X(y_2 - y) \bmod 1$ may be viewed as uniformly distributed on $[0, 1)$, too. The term $A'_{y,m}$ denotes the event that the random variable C_1 fulfills (5.68). Dividing the second line below by $2^{seedlen}$ yields 873

$$\begin{aligned} &\mathrm{Prob}(A_{y,m}) < \mathrm{Prob}(A'_{y,m}) = \\ &\mathrm{Prob}((y_2 - y)C_1 \cdot 2^{outlen} \bmod 2^{seedlen} \in [(2^{seedlen-t} - 1)2^t, 2^{seedlen}) \cup [0, 2^{12})) \approx \\ &\mathrm{Prob}(X_y \in [1 - 2^{-seedlen+t}, 1)) \approx 2^{-seedlen+t}\,. \end{aligned} \tag{5.69}$$

since $2^{12} \ll 2^t$. Putting the pieces together gives

$$\begin{aligned} &\mathrm{Prob}(\text{pre-image hit of } y_2 + i + 1) < \sum_{y=1}^{y_2-1} \mathrm{Prob}(A_{y,m}, A_{y,\ell}) < \sum_{y=1}^{y_2-1} \mathrm{Prob}(A'_{y,m}, A_{y,\ell}) = \\ &\sum_{y=1}^{y_2-1} \mathrm{Prob}(A'_{y,m}) \cdot \mathrm{Prob}(A_{y,\ell}) \le 2^{48} \cdot 2^{-seedlen+t} \cdot 2^{-outlen+12} = \\ &2^{-seedlen+110} \le 2^{-330}\,. \end{aligned} \tag{5.70}$$

(Since C_1 does not affect $\mathrm{Prob}(A_{y,\ell})$ this implies that $\mathrm{Prob}(A'_{y,m}, A_{y,\ell}) = \mathrm{Prob}(A'_{y,m}) \cdot \mathrm{Prob}(A_{y,\ell})$.) To be precise, for the SHA-384 and SHA-512 hash functions, the probability for a pre-image hit of $y_2 + i + 1$ is $\le 2^{-778}$. This means that the Hash_ DRBG provides forward secrecy, i.e., fulfills requirement DRG.3.5.

[backward secrecy] The proof of the backward secrecy property can be organized analogously. The only difference is that in place of the pre-images $v_{y_2} + j + 1$ or v_{y_2+1}, the pre-images $v_{y_1} + i - 1$ or $v_{y_1-1} + m_{y_1-1}$ (if $i = 0$), respectively, have to be considered. In particular, the Hash_DRNG fulfills the backward secrecy requirement, i.e., fulfills requirement DRG.3.6. 874

[enhanced backward secrecy] Next, we verify that the Hash_DRBG has enhanced backward secrecy. We assume that an adversary knows the internal state of the Hash_DRBG after request y, namely $(v_{y+1}, c, y + 1)$. Assume further that $\widetilde{v}_y = \phi_{req}(v_y, a_y)$. To violate the enhanced backward secrecy property, an adversary has to determine any of the random numbers 875

$\psi(\widetilde{v}_y), \psi(\widetilde{v}_y+1), \ldots, pr_u(\psi(\widetilde{v}_y+m_y-1))$ of request y or to guess them with significantly larger probability than without knowledge of the internal state. (Of course, knowledge of $\widetilde{v}_y$ or (v_y, a_y) would solve this problem.) In particular,

$$(v_{y+1}, c, y+1) = \phi_B(\widetilde{v}_y, c, y) = (\widetilde{v}_y + g(\widetilde{v}_y) + c + y \bmod 2^{seedlen}, c, y+1) \quad \text{and} \tag{5.71}$$

$$\widetilde{v}_y = v_y + f(v_y, a_y) \bmod 2^{seedlen} \tag{5.72}$$

If $a_y = \emptyset$, then $v_y = \widetilde{v}_y$. Assume that $a_y \neq \emptyset$. By (5.50) it would not be easier to determine v_y first instead of $\widetilde{v}_y$ even if an adversary knew the value $f(v_y, a_y)$. In both cases an adversary had to solve an equation of the following type: $v + g(v) \equiv c \bmod 2^{seedlen}$ with known right-had side c.

876 [enhanced backward secrecy] By the assumption in par. 875, the values c and y are known. Hence, (5.71) allows the adversary to determine the sum $\widetilde{v}_y + g(\widetilde{v}_y)$, or more precisely, $\widetilde{v}_y + g(\widetilde{v}_y) \bmod 2^{seedlen}$. (In very rare cases $\widetilde{v}_y + g(\widetilde{v}_y)$ may exceed the modulus $2^{seedlen}$.) As pointed out in the note of par. 856, this allows finding an interval I_b of length 2^{outlen}, which contains the pre-image $\widetilde{v}_y$. By (5.58) an adversary cannot solve this pre-image problem practically.

877 [enhanced backward secrecy] Recall that the adversary searches the image $\psi(\widetilde{v}_y + i)$ for some $i \leq 2^{12}$. By par. 876 the knowledge of $\widetilde{v}_y + g(\widetilde{v}_y)$ does not suffice to determine $\widetilde{v}_y$. Thus, the adversary is not able to guess any of the random numbers $\psi(\widetilde{v}_y + i)$ with significantly greater probability than without knowledge of the current internal state $(v_{y+1}, c, y+1)$. Moreover, by (5.55) $\mathrm{Hash}_{\mathrm{df}}(\cdot, seedlen)$ is a one-way function. Hence, it is not feasible to determine the first internal state v_1 from c or to guess it with significantly greater probability than without knowledge of c (cf. par. 843). (Of course, the knowledge of v_1 (together with the knowledge of the additional input data $a_1, \ldots, a_y$) would allow recovering all the previous random numbers.) Hence, the Hash_DRBG fulfills the enhanced backward secrecy requirement, i.e., requirement DRG.3.7.

878 [effective internal state] In this paragraph we consider the backward secrecy and forward secrecy properties. The set of all internal states is $S = S' \times S' \times \mathbb{Z}_{2^{48}}$. The third component equals the number of the next request and may be publicly known. Since $\psi(\cdot)$ is a one-way function, it is not possible to determine any intermediate value $\widetilde{v}_y$ from the random numbers in (5.49) or (in a more favorable scenario for the adversary) from (5.61), even if $y_1 = 1$, $y_2 = 2^{48}$, and if for each v_y the subsequence of random numbers has maximum length. In particular, it is not possible to get $v_{y+1} = pr_{(S')}(\phi_B(\widetilde{v}_y, c, y))$ or $v_y \in \phi_{req}^{-1}(\widetilde{v}_y)$. Furthermore, c is the image of v_1 under a one-way function (cf. (5.55), and pars. 839 and 843). Thus, it would require knowledge of v_1 to compute c. The set of effective internal states is given the cartesian product $S' \times S'$.

879 [request requirement] By specification the Hash_DRBG fulfills requirement DRG.3.2.

880 [additional input] Since $f(v, a)$ (if $a \neq \emptyset$) (5.35) and $g(v)$ (5.36) are closely related to *Hash* targeted chosen input attack are not feasible because v_y is unknown. Hence, the Hash_DRBG fulfills requirement DRG.3.8.

881 [cryptographic functions] Both the state transition function ϕ and the output function ψ are cryptographic and composed of summarized cryptographic primitives. Thus, the Hash_DRBG fulfills requirement DRG.3.9.

882

[statistical tests] The first components of the internal states, $v_1, v_2, \ldots$, and also the intermediate values $\widetilde{v}_1, \widetilde{v}_2, \ldots$, are images of v_1 under the repeated application of two one-way functions (possibly affected by additional input values $a_1, a_2, \ldots$). For each intermediate value $\widetilde{v}_y$, less than 2^{12} random numbers $\psi(\widetilde{v}_y), \psi(\widetilde{v}_y + 1), \ldots$ are generated. The random numbers are the (possibly truncated) hash values of input values that are mutually distinct with overwhelming probability (cf. par. 873 ff.). If the sequence of random numbers (interpreted as a binary sequence) within a seedlife would fail, *fair* statistical tests (i.e., which do not exploit knowledge of the internal state) significantly often, this would point to inherent weaknesses of the hash function. Since, for the permitted hash functions (5.26), no statistical weaknesses are known, we may assume that the Hash_DRBG fulfills requirement DRG.3.10.

[summary] The Hash_DRBG fulfills the algorithmic requirements DRG.3.2 (par. 879), DRG.3.3 (pars. 826, 878, **??**), DRG.3.5 (pars. 863 to 873), DRG.3.6 (par. 874), DRG.3.7 (pars. 875 to 877), DRG.3.8 (par. 880), DRG.3.9 (par. 881), and DRG.3.10 (par. 882). 883

[(re-)seeding] A verification of requirements DRG.3.1 and DRG.3.4 is still needed. These requirements are concerned with the true RNG that is used for the seeding procedure / reseeding procedure and thus require case-by-case considerations. Par. 885 formulates sufficient conditions that are easy to check. 884

[(re-)seeding] Assume that the entropy_input string consists of ≥ 256 bits that have been generated by a TRNG that is compliant with class PTG.2, PTG.3 or NTG.1. Then requirement DRG.3.1 is fulfilled. Furthermore, requirement DRG.3.4 is fulfilled, too. We first note that the Shannon entropy of the entropy_input is > 255. If the length of the entropy_input string is ≈ 256 bits (with a given (fixed) nonce and personalization_string, resp. with given additional_input) $\phi_{seed}(\cdot,\cdot)$, resp. $\phi_{reseed}(\cdot,\cdot)$, should be 'almost' injective, thus losing only marginal entropy. Note that even if the entropy_input string consists of *seedlen* bits, the entropy loss is marginal; see Sect. 4.4. Thus, if the length of the entropy_input increases, the defect caused by collisions increases much slower than the overall entropy of the entropy_input string. 885
As an alternative to a TRNG that is compliant with PTG.2, PTG.3 or NTG.1, one can use a TRNG with some guaranteed entropy bound (DRG.3-compliance; cf. DRG.3.1). To achieve compliance to class DRG.4, the TRNG has to be a PTRNG (cf. DRG.4.1). The minimum number of bits needed from this TRNG (resp. PTRNG depends on the guaranteed entropy bound.

[(re-)seeding] The second component of the internal state, $c = c(v_1)$, does not increase the entropy of the effective internal state because c is a function of v_1. 886

[enhanced forward secrecy] The Hash_DRBG can be 'upgraded' to class DRG.4 if (parts of) the entropy_input-string (for the seeding procedure and the reseeding procedure) and / or the high-entropy additional input data are generated by a physical RNG, provided that a suitable calling scheme is used (see DRG.4.10). The explanations from pars. 884 and 885 can be transferred to high-entropy additional input (if applicable) if we replace 'entropy_input' by '(high-entropy) additional input'. 887

[additional input, PTG.3] If the fresh entropy is introduced by additional input, the amount of entropy is bounded by the output length of the function $f(\cdot,\cdot)$, or equivalently, by the output length of the applied hash function *Hash*, resp. by the bit length of a single internal random 888 number. This is an important observation when the Hash_ DRBG is considered as cryptographic post-processing for a PTG.3-compliant PTRNG. Introducing fresh entropy by the seeding procedure or reseeding procedure allows the introduction of (almost) *seedlen* bits of entropy.

5.3.2 Security Evaluation of the HMAC_ DRBG [SP800-90A]

889 In this subsection we analyze the conformity of the HMAC_ DRBG to the requirements of functionality class DRG.3. The relevant parts of the analysis are based on [Kels23]; cf. pars. 912, 913, 914, and 923.

890 For a detailed description of the HMAC_ DRBG we refer to [SP800-90A], Subsubsect. 10.1.2.

891 In the following we assume

$$\begin{aligned} Hash \in \{&\text{SHA-256}, \text{SHA-512/256}, \text{SHA-384}, \text{SHA-512}, \\ &\text{SHA3-256}, \text{SHA3-384}, \text{SHA3-512}\}\,. \end{aligned} \tag{5.73}$$

The hash function *Hash* outputs strings of length $outlen \in \{256, 384, 512\}$ bits as indicated by its name; if the name includes two numbers the output length is indicated by the second number.

892 Our first goal is to describe the HMAC_ DRBG by a 9-tuple (par. 139) and by the 4-tuples for the seeding procedure (par. 158) and reseeding procedure (par. 162).

893 In this subsection $S' := \{0,1\}^{outlen}$.

894 The internal space of the HMAC_ DRBG (denoted as working state in [SP800-90A]) is given by the cartesian product

$$S := S' \times S' \times \mathbb{Z}_{2^{48}}\,. \tag{5.74}$$

Its elements are triples (k, v, rc). The values k and v are kept secret while the value of the reseed_counter rc is publicly known. The reseed_counter is initialized by 1 and incremented by 1 after each request. The reseeding procedure is required after 2^{48} requests at the latest.

895 The first component S' of $S = S' \times S' \times \mathbb{Z}_{2^{48}}$ is the effective internal state; cf. par. 915.

896 Furthermore,

$$S_{req} := S' \times S' \tag{5.75}$$
$$A := \{0,1\}^* \tag{5.76}$$
$$R := \{0,1\}^{outlen} \tag{5.77}$$
$$I := \mathbb{Z}_{2^{19}} \tag{5.78}$$

The bit length of additional input $a \in A$ is $\leq 2^{35}$. Empty strings are possible (denoted by o).

897 Next, we define the state transition function ϕ, the output function ψ, the function ϕ_{req} (generating the internal request state). and the request state transition function ϕ_0. First, we repeat

the definition of the function HMAC_DRBG_Update in [SP800-90A] (up to the order of its arguments). In pars. 898 and 899 we use an algorithmic notation to simplify reading.

The function 898

$$\begin{aligned}&\mathrm{HMAC\_DRBG\_Update}\colon S' \times S' \times \{0,1\}^* \to S' \times S'\,,\\ &\mathrm{HMAC\_DRBG\_Update}(k,v,a) := \end{aligned} \tag{5.79}$$

is given by

$$\begin{aligned}&k := \mathrm{HMAC}(k, v\|0x00\|a)\\ &v := \mathrm{HMAC}(k, v\|0x00\|a)\\ &\text{if } a \neq o \text{ then } \{\\ &\quad k := \mathrm{HMAC}(k, v\|0x01\|a)\\ &\quad v := \mathrm{HMAC}(k, v\|0x01\|a)\\ &\} && (5.80)\\ &\text{return } (k,v) && (5.81)\end{aligned}$$

In particular, HMAC_DRBG_Update(k,v,a) updates k and v.

After each request the internal state S has been updated by the state transition function ϕ where 899

$$\phi\colon S \times A \times I \to S\,, \quad \phi(k,v,rc,a,p) := \tag{5.82}$$

is given by

$$\begin{aligned}&\text{if } a \neq o \text{ then } (k,v) := \mathrm{HMAC\_DRBG\_Update}(k,v,a)\\ &\text{for } j := 1 \text{ to } m \text{ do } \{\\ &\quad v := \mathrm{HMAC}(k,v);\\ &\}\\ &(k,v) := \mathrm{HMAC\_DRBG\_Update}(k,v,a)\\ &\text{return } (k,v,rc+1) && (5.83)\end{aligned}$$

As in par. 141 it is $m := \lceil \frac{p}{k} \rceil$.

The internal request state s_{req} is generated by 900

$$\phi_{req}\colon S \times A \to S_{req}\,, \quad \phi_{req}(k,v,rc,a) := \begin{cases}\mathrm{HMAC\_DRBG\_Update}(k,v,a) \text{ if } a \neq o\\ (k,v) \text{ if } a = o\end{cases} \tag{5.84}$$

Finally, 901

$$\phi_0\colon S_{req} \to S_{req}\,, \quad \phi_{req}(k,v) := (k, \mathrm{HMAC}(k,v)) \tag{5.85}$$

$$\psi\colon S_{req} \to R\,, \quad \psi(k,v) := \mathrm{HMAC}(k,v) \tag{5.86}$$

This completes the specification of the describing 9-tuple.

Note: From a logical point of view, the internal state is updated by the state transition function ϕ (cf. (5.82) and (5.83) for details), and each value $v := \mathrm{HMAC}(k, v)$ in the for-loop of (5.81) is computed two more times (by ϕ_0 and ψ). The implementation, of course, computes each value only once (by ψ), and s_{req} equals the first two components of the current internal state s. The request counter rc is initialized by 1 and is increased by 1 after each request.

903 The following paragraphs provide formal descriptions of the seeding procedure and the reseeding procedure.

904 The initial internal state is computed from the seed material (denoted by seed_material in [SP800-90A]). Like for the Hash_DRBG

$$\text{seed\_material} \;= (\text{ entropy\_input } || \text{ personalization\_string }) \tag{5.87}$$

The requirements on the seed_material and its components match with those for the Hash_ DRBG; cf. par. 835 for details.

905 In the notation of the 4-tuple, which describes the seeding procedure (par. 158)

$$SM = PS = \{0,1\}^* \tag{5.88}$$

906 The nonce and the personalization_string are constructed from the current value of PS.

907 The first initial state (k_1, v_1, rc) is computed from seed_material via

$$\begin{aligned}
&\phi_{seed} \colon \{0,1\}^* \to \{0,1\}^{outlen} \times \{0,1\}^{outlen} \times \mathbb{Z}_{2^{48}},\\
&\phi_{seed}(\text{seed_material}) := (\text{HMAC_DRBG_Update}(k_0, v_0, \text{seed_material}), 1) = (k_1, v_1, rc)\\
&\quad \text{with } k_0 := 0x00\ldots,00 \quad \text{and } v_0 := 0x01\ldots,01
\end{aligned} \tag{5.89}$$

Note: In the seeding procedure the bit string seed_material (5.87) is the concatenation of values in SM (entropy_input) and PS (nonce and personalization_string).

908 For the reseeding procedure

$$\text{seed\_material} \;= (\text{ entropy\_input } || \text{ additional\_input}) \tag{5.90}$$

The maximum bit length of both the entropy_input and the additional_input is 2^{35} [SP800-90A], Table 2. The additional_input may be empty. The additional_input belongs to the set PS.

909 In the notation of the 4-tuple which describes the reseeding procedure (par. 162)

$$SM = PS = \{0,1\}^* \tag{5.91}$$

910 The first initial state (k_1, v_1, rc) after the reseeding procedure is computed from seed_material via

$$\begin{aligned}
&\phi_{reseed} \colon S \times \{0,1\}^* \to \{0,1\}^{outlen} \times \{0,1\}^{outlen} \times \mathbb{Z}_{2^{48}}\\
&\phi_{reseed}((k, v, rc), \text{seed_material}) := (\text{HMAC_DRBG_Update}(k, v, \text{seed_material}), 1) = (k_1, v_1, rc)
\end{aligned} \tag{5.92}$$

The first two arguments of HMAC_DRBG_Update$(\cdot,\cdot,\cdot)$, k, v, denote the current values of the internal state before reseeding.

When proving the conformity to functionality class DRG.3 the main difficulty lies in verifying that the DRNG fulfils backward secrecy, forward secrecy, and enhanced backward secrecy. If the conformity to functionality class DRG.4 is targeted, enhanced forward secrecy has to be verified, too. For a security proof of the HMAC_ DRBG we refer to [Kels23]. This paper uses the random oracle model. It is proved that specified games cannot be won by an attacker. 911

[Forward secrecy] The HMAC_ DRBG fulfils Requirement DRG.3.5 (forward secrecy); see [Kels23], Section 5.3. 912

[Backward secrecy] The HMAC_ DRBG fulfils Requirement DRG.3.6 (backward secrecy); see [Kels23], Section 5.3. 913

[Enhanced backward secrecy] The HMAC_ DRBG fulfils requirements DRG.3.7 (enhanced backward secrecy); see [Kels23], Section 5.5. 914

[effective internal state] The set of admissible internal states is $S = S' \times S' \times \mathbb{Z}_{2^{48}}$. The third component equals the number of the next request and may be publicly known, and each random number equals the v-value of an intermediate internal state; cf. (5.82), for-loop. Hence the admissible effective internal states are contained in $S' = \{0,1\}^{outlen}$, the first component of the set of admissible internal states. On the other hand, due to the properties of ϕ and ψ, ultimately due to the one-way-property of the functions HMAC and *Hash*, the admissible effective internal states equal $S' = \{0,1\}^{outlen}$. Hence Requirement DRG.3.3 is fulfilled. Due to the properties of the seeding procedure and reseeding procedure Requirement DRG.3.4 is fulfilled, if the seed_material contains at least 240 bits of min-entropy. Alternatively, under suitable conditions (cf. par. 124, third bullet point) it is accepted if the seed_material contains at least 250 bits Shannon entropy. 915

[request requirement] By specification the Hash_ DRBG fulfills requirement DRG.3.2. 916

[additional input] Due to the properties of the HMAC function the HMAC_ DRBG fulfills requirement DRG.3.8. 917

[cryptographic functions] Both the state transition function ϕ and the output function ψ are cryptographic and composed of recognized cryptographic primitives. Thus, the HMAC_ DRBG fulfills requirement DRG.3.9. 918

[statistical tests] Per request at most $2^{19}/outlen \leq 2^{19-8} = 2^{11}$ internal random numbers are generated if $outlen \geq 256$. Thus, at most $2^{48+11} = 2^{59}$ internal numbers are generated until reseeding. The internal random numbers are computed from the current internal state (k, v, rc) via HMAC(k, v). Next, we estimate the probability for a collision of pre-images (k, v) until reseeding. The probability that no collision of pre-images (k, v) in request j occurs under the 919 condition that no collision has occurred within the requests $1, \ldots, j-1$ is

$$\geq \left(1 - \frac{(j-1)2^{11}}{2^{2outlen}}\right) \cdot \prod_{i=1}^{2^{11}} \left(1 - \frac{i-1}{2^{outlen}}\right) \tag{5.93}$$

The first factor covers the update mechanism of the internal state at the end of request $j-1$ (and possibly at the beginning of request j) and compares the actual pair (k, v) before the for-loop with all pairs (k, v) within in the requests $1, \ldots, j-1$. The second factor covers the updates of the internal state within the for loop in (5.82). Note that k remains fixed until the end of the request, resulting in the denominator 2^{outlen}. Altogether,

$$\begin{aligned} &\text{Prob (collision of pre-images until reseeding)} \leq 1 - \prod_{j=1}^{2^{48}} \left(1 - \frac{(j-1)2^{11}}{2^{2outlen}}\right) \cdot \left(\prod_{i=1}^{2^{11}} \left(1 - \frac{i-1}{2^{outlen}}\right)\right)^{2^{48}} \\ &\approx 1 - e^{-2^{48+48-1+11-2outlen}} e^{-2^{11+11-1+48-outlen}} = 1 - e^{-2^{106-2outlen} - 2^{69-outlen}} \\ &\approx 2^{106-2outlen} + 2^{69-outlen} \end{aligned} \tag{5.94}$$

The last steps is justified by a linear Taylor expansion of the exponential function. Hence a collision of pre-images occurs only with negligible probability. Due to the properties of HMAC and *Hash* we may assume that the internal random numbers are generated from randomly selected pre-images. Since, for the permitted hash functions (5.73), no statistical weaknesses are known, we may assume that the HMAC_ DRBG fulfills requirement DRG.3.10.

920 [summary] The Hash_DRBG fulfills the algorithmic requirements DRG.3.2 (par. 916), DRG.3.3 (pars. 895, 915), DRG.3.5 (pars. 912), DRG.3.6 (par. 913), DRG.3.7 (pars. 914), DRG.3.8 (par. 917), DRG.3.9 (par. 918), and DRG.3.10 (par. 919).

921 [(re-)seeding] It remains the verification that the requirements DRG.3.1 and DRG.3.4 are fulfilled. These requirements concern the true RNG, which is used for the seeding procedure / reseeding procedure and thus require case-by-case considerations. Par. 922 formulates sufficient conditions, which are easy to verify. In the case 'DRNG seeds DRNG' these requirements are dropped.

922 [(re-)seeding] Assume that the entropy_input string consists of ≥ 256 bits which have been generated by a TRNG that is compliant with class PTG.2, PTG.3 or NTG.1. Then requirement DRG.3.1 is fulfilled. Furthermore, requirement DRG.3.4 is fulfilled, too. We note that the Shannon entropy of the entropy_input is > 255. If the length of the entropy_input string is ≈ 256 bits (with given (fixed) nonce and personalization_string, resp. with given additional_input) the projection of $\phi_{seed}(\cdot, \cdot)$, resp. $\phi_{reseed}(\cdot, \cdot)$, onto the first component should be 'almost' injective, thus losing only marginal entropy. If the length of the entropy_input increases, the defect caused by collisions increases much slower than the overall entropy of the entropy_input string.
As an alternative to a TRNG which is compliant with PTG.2, PTG.3 or NTG.1, one can use a TRNG with some guaranteed entropy bound (DRG.3-compliance; cf. DRG.3.1). To achieve compliance to class DRG.4, the TRNG has to be a PTRNG (cf. DRG.4.1). The minimum number of bits needed from this TRNG (resp. PTRNG depends on the guaranteed entropy bound.

923 [Enhanced forward secrecy] The HMAC_ DRBG fulfils requirements DRG.4.10 (enhanced forward secrecy) if the seed_material for contains at least 240 bits of min-entropy; see [Kels23],

Section 5.4. Under suitable conditions (cf. par. 124, third bullet point) it is alternatively accepted if the seed_material contains at least 250 bits Shannon entropy. It is also accepted if (sufficient) fresh entropy is inserted by high-entropy additional input (if applicable).

5.4 Noise Sources and Stochastic Models

Subsect. 5.4.1 addresses examples of different noise sources that are often used by PTRNGs and NPTRNGs. In Subsects. 5.4.2 to 5.4.6 several stochastic models are discussed and analyzed. Subsects. 5.4.2, 5.4.5, and 5.4.6 consider concrete designs of physical noise sources while Subsects. 5.4.3 and 5.4.4 focus on the mathematical analysis of generic designs. Several designs of physical noise sources fit these generic designs. 924

5.4.1 Examples of physical and non-physical noise sources

Below, a number of noise sources are mentioned that are used by PTRNGs (pars. 927 to 934) and NPTRNGs (pars. 936 to 939). This list does not claim to be complete and does not provide any kind of quality assessment. AIS 20 and AIS 31 are technology neutral. The applicant has to give evidence that the requirements of the aimed functionality class are fulfilled. 925

The quality of a PTRNG does not only depend on the analog part of the physical noise source but on the whole design, including the digitization mechanism. Due to (inadvertent) band-pass filtering, inherent noise, and probabilistic detection, for example, a digitization mechanism may undesirably blur even a physically perfect noise signal or introduce dependencies between samples. For this reason, this document assumes that the digitization mechanism (and, if applicable, the sampling mechanisms) is part of the physical noise source of a PTRNG. This applies to non-physical noise sources as well. 926

Shot entropy of a tube diode. The shot entropy of a parallel-plane temperature-limited tube diode is non-deterministic. The number of electrons emitted from the tube's cathode during a time interval follows a Poisson distribution, cf. [DaRo87], Sect. 7-2. 927

Thermal resistive entropy. The voltage between resistors varies randomly due to the vibration of atoms. Ideally, the thermal entropy signal has the same energy in all frequency bands (so called "white noise"). Sampling an ideally-amplified white noise signal would generate a sequence of independent bits. 928

Semi-conductor diode breakdown entropy. The reverse current through semi-conductor diodes varies randomly due to the tunneling of electrons. The power of the entropy signal is inversely proportional to the frequency. 929

Free running oscillators. Free running oscillators generate digital signals with an edge-to-edge random analog 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 (nearly) uncorrelated. 930

931 Designs based on metastability in digital circuits. This comprises various designs where parts of a digital circuit are forced into a state between logic levels '0' and '1' to induce unpredictable behavior.

932 Chaos based noise source. This comprises designs whose behavior is highly sensitive to small variations (e.g., in voltage, current, or time due to inherent noise). The entropy of the raw random numbers results from the entropy introduced by physical disturbances and the noise source's ability to amplify them and make them measurable ([BuLu08; BuLu16]). Although classical (mathematical) chaos theory only considers variations of the initial conditions, this kind of modeling (finally, a DRNG with unlimited entropy in the seed material) is not appropriate for real-world PTRNGs and will not be accepted. Instead, it has to be shown that the average supply of entropy to the system exceeds the output rate.

933 Radioactive atomic disintegration. The number of decay events (detected particles) per time interval follows a Poisson distribution; see Subsect. 5.4.5.

934 Quantum noise source. Quantum RNGs exploit physical phenomena that contain randomness according to the laws of quantum mechanics. This document does not distinguish between quantum entropy and entropy from physical phenomena based on other physical models. AIS 31 considers quantum RNGs as PTRNGs already because of the digitization mechanism that transfers the analog data to raw random numbers.

935 Pars. 936 to 939 consider noise sources for NPTRNGs.

936 General system data. A computer (e.g., a PC or a server) offers a variety of possibilities to collect data that are non-deterministic. It should be noted, however, that many noise sources deliver information that are comparatively easy to guess, to influence, or determine in a different way. Examples: network data, file system or process header information, threads, current time and date, time since system start, disk I/O operations, interrupts, etc. The reference [Linux_RNG_2022] treats the Linux /dev/random and /dev/urandom.

937 Time stamps. If available (e.g., CPU instruction RDTSC), a highly precise time stamp counter can be used to generate data which are hard to predict by an adversary. The least significant bits of time stamps should be affected by all activities currently running on the computer. Under suitable conditions virtualization does not negatively influence the suitability of time stamps as noise sources for NPTRNGs; cf. [Linux_RNG_2022; RNG_virtual_env].

938 Human interaction. Input data generated by the user (e.g., mouse movement and key strokes) usually contains little entropy. In order to generate a (considerable) amount of entropy from human interaction, the computer needs to apply highly precise time resolution (similar to par. 937).

939 Software execution jitter. This approach uses specially crafted software whose execution time varies greatly. The cause for variances of the execution times depends on the platform; see [Jitter-RNG] etc.

940 Note: Applicants for certificates, designers, and evaluators may apply the results and strategies

from Subsects. 5.4.2 to 5.4.6. But, of course, it has to be verified that the assumptions are justified for the noise source under evaluation.

5.4.2 PTRNG with two noisy diodes

In Subsect. 5.4.2 we discuss a PTRNG design that exploits two noisy diodes. The design is analyzed, relevant conclusions are summarized, and finally a stochastic model is developed. For many details we refer the interested reader to [KiSc08]. This and related designs are also treated in [Schi09b]. Some new considerations are added in the following. We mention that this design is also treated in ISO / IEC 20543 [ISO_20543], A.3.4, Example 2. 941

[diodes] Zener diodes have a reverse avalanche effect (depending on diode type, $3-4\ V$ or about $10\ V$) and produce more than 1mV of noisy voltage with a cutoff frequency of 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. 942

[design] Fig. 12 illustrates the PTRNG design that is discussed in Subsect. 5.4.2. The circuit of 943

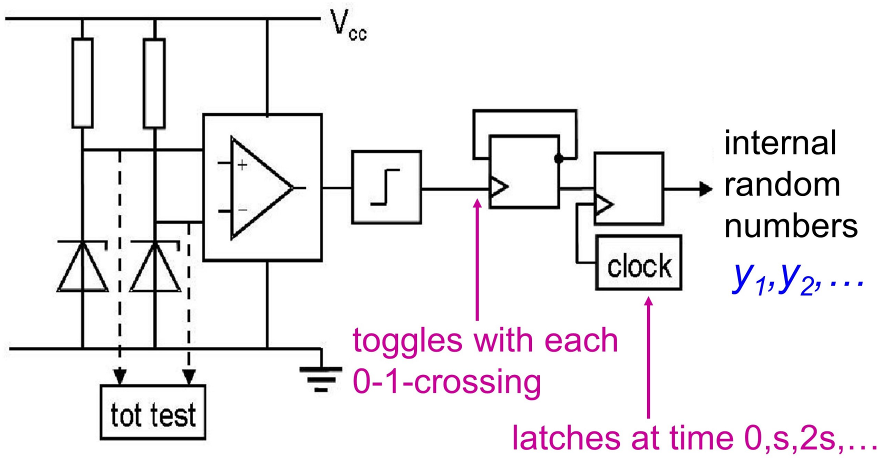

Figure 12: PTRNG with two noisy diodes (schematic design), created by W. Killmann

the AC coupling, the negative feedback for the operational amplifier, the stabilizing mechanism for the power supply, and compensating effects of temperature have been omitted for clarity in Fig. 12.

[design] In Fig. 12 the outlets of two identical noisy diodes provide the input to an operational amplifier. The operational amplifier applies bandpass filters and amplifies the difference of the noisy voltages (with a very high amplification rate). Its output voltage is fed into a Schmitt trigger. The mean voltage of the output signal of the amplifier is approximately in the middle of the two threshold values of the Schmitt trigger. When the input voltage is below the lower threshold value, the output signal of the Schmitt trigger assumes the value 'low' (= 0); when 944

the input voltage exceeds the higher threshold value, the output value is 'high' (= 1); and when the input voltage is between the two levels, the output retains its value. For the generation of random numbers, the proposed design exploits the 0-1-crossings ('up-crossings'). Each up-crossing switches the output value of the Schmitt trigger from 0 to 1 and clocks an intermediate flip-flop that inverts the D-input of a second flip-flop. The second clock is latched by a regular clock signal at equidistant times $s_0 := 0, s_1 := s, s_2 := 2s, \ldots$.

945 The physical noise source exploits the random time intervals between subsequent 0-1-crossings. Due to the steep edges of the output of the operational amplifier and since only the 0-1-crossings are exploited, the hysteresis effect should be negligible.

946 [random numbers] The number of 0-1-crossings within the n^{th} clock cycle, i.e., within the time interval $I_n := (s_{n-1}, s_n] = ((n-1)s, ns]$, gives the raw random number $r_n \in \mathbb{N}_0$. Finally, the internal random numbers are given by $y_{n+1} = y_n \oplus r_{n+1} (= y_n \oplus r_{n+1} (\bmod 2))$. We denote the sequence $r_1, r_2, \ldots$ as 'virtual' in this context because these integers never appear explicitly. Although the internal random numbers (5.95) depend only on $r_n (\bmod 2)$, the least significant bit of r_n, the stochastic model and the online tests should consider the virtual raw random numbers $r_1, r_2, \ldots$, as they contain more information than their least significant bits.
Note: 'y_0' denotes the output value of the flip-flop when the 'observation' (at time $t = 0$) starts.

947 Principally, the design in Fig. 12 would also work with a single noisy diode in place of two. A single diode is potentially more vulnerable to environmental conditions, and in particular, to an adversary who aims to manipulate the output voltage of the diode(s) by active attacks, e.g., by applying an external electromagnetic field.
Note: Designs based on single diodes are not generally unsuitable, but additional measures should then be considered to mitigate these threats.

948 [experiments] We provide experimental results from a PTRNG prototype for which the design left to the first flip-flop equals the schematic design from Fig. 12; cf. [KiSc08], Sect. 5). We mention that the operations that follow the up-crossings are principally deterministic as soon as the position of one 0-1-crossing relative to the regular clock signal has been fixed; cf. par. 989
The PTRNG prototype was kindly provided by Frank Bergmann.

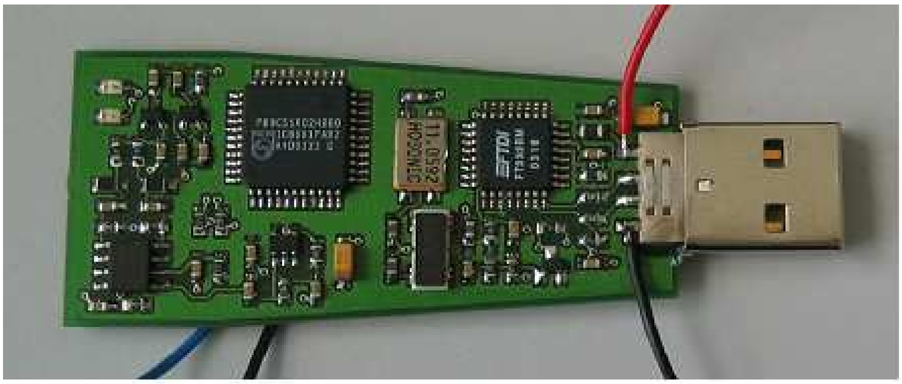

Figure 13: Hardware setup of the PTRNG [KiSc08], Fig. 2

949

Obviously,

$$y_n \equiv y_{n-1} + r_n \equiv y_0 + r_1 + \cdots + r_n \pmod 2 \quad \text{for } n \geq 1 \tag{5.95}$$

where y_0 denotes the internal random number at time $t = 0$. This simple algorithmic post-processing allows the transfer of results on the raw random numbers $(\bmod 2)$ to the internal random numbers.

Below, we summarize analyses and facts that are relevant for the evaluation and for the stochastic model. As usual, we interpret the (virtual) raw random numbers $r_1, r_2, \ldots$ and the internal random numbers $y_0, y_1, \ldots,$ as realizations of random variables $R_1, R_2, \ldots$ and $Y_0, Y_1, \ldots,$ respectively. Our goal is to (at least) determine lower bounds for the following average conditional entropies 950

$$H(R_{n+1} \mid R_1, \ldots, R_n) \quad \text{and finally} \tag{5.96}$$

$$H(R_{n+1} (\bmod 2) \mid R_1 (\bmod 2), \ldots, R_n (\bmod 2)) = H(Y_{n+1} \mid Y_0, Y_1, \ldots, Y_n) \tag{5.97}$$

The right-hand conditional entropy in (5.97) corresponds to the real-world scenario where an adversary knows several internal random numbers $y_0, y_1, y_2, \ldots, y_n$.

We interpret the lengths $t_1, t_2, \ldots$ of the time intervals between consecutive 0-1-crossings as realizations of a stochastic process $T_1, T_2, \ldots$. One may assume that the analog part of the physical noise source is in an equilibrium state when enough time has passed since the start of the PTRNG (a fraction of a second should suffice). The stochastic behavior of the PTRNG is determined by several operational constants (as breakdown voltages of the noisy diodes, electronic characteristics of the amplifier, or threshold levels of the Schmitt trigger). Consequently, shortly after the start-up the stochastic process, $T_1, T_2, \ldots$ should be stationary, or more precisely, time-local stationary. Long-term drifts (caused by the feedback loop of the amplifier, by changing environmental conditions, or by ageing effects) are ignored in the following, cf. pars. 668 to 671. If needed, earlier time intervals between consecutive 0-1-crossings (before equilibrium) may be denoted with negative indices $(\ldots, t_{-2}, t_{-1}, t_0)$. 951

It should be noted that in a modification of this design where both the 0-1-crossings and the 1-0-crossings are counted, the sequence of random intervals between two consecutive crossings would presumably lose the time-local stationarity property. The reason for that is that the random intervals between consecutive 1-0-crossings and 0-1-crossings are in general not identically distributed. And even if the time-local stationarity would still hold, its justification and verification would become significantly more difficult. The selected design increases the robustness of the design and simplifies its security analysis at the cost of halving the output rate. Recall that physical noise sources that generate non-stationary raw random numbers are not compliant with the functionality classes PTG.2 and PTG.3. 952

Due to the nature of shot noise, one may assume that the stochastic process $T_1, T_2, \ldots$ is q-dependent (cf. par. 495) with small q. This assumption was supported by experiments, see [KiSc08], Sect. 5. For the lags $\tau = 1, \ldots, 5$ the autocorrelation 953

$$\frac{E\left((T_j - E(T_j))(T_{j+\tau} - E(T_{j+\tau}))\right)}{\sqrt{\mathrm{Var}(T_j)}\sqrt{\mathrm{Var}(T_{j+\tau})}} \tag{5.98}$$

of the stochastic process $T_1, T_2, \ldots$ was estimated. In all cases the absolute value was < 0.002, which suggests that both q and the magnitude of dependencies is small. In fact, this observation is consistent with the hypothesis that the random variables $T_1, T_2, \ldots,$ are essentially iid. But the conclusion that the variables are indeed iid cannot be drawn from this observation alone. Instead, this would require further analysis. As in [KiSc08] we cautiously assume $q \leq 1$ but especially point to results that apply when the T_j are iid.

954 The q-dependence $T_1, T_2, \ldots$ ensures that a version of the CLT (Central Limit Theorem) applies to the random variables $T_1, T_2 \ldots$; cf. par. 966. We introduce the notation $\mu := E(T_1)$ and $\sigma_T^2 := \mathrm{Var}(T_1)$. Of course, $\sigma_T^2 > 0$ since otherwise the 0-1-crossings would appear periodically, and the random numbers would not have any entropy. Further (natural and non-restrictive) assumptions are that $E(|T_j|^3) < \infty$ (necessary condition for the applied version of the CLT) and $\mathrm{Prob}(T_1 = 0) = 0$ which is ensured by the technical properties of a Schmitt trigger.

955 Par. 956 considers the one-dimensional distribution of the T_j, while pars. 957 and 958 address the output of the operational amplifier.

956 Fig. 14 plots the distribution of the time intervals between successive 0-1-crossings, and Fig. 15 illustrates the percentiles of the distribution. (Both diagrams belong to different measurements.) These experiments verify that the random variables $T_1, T_2, \ldots$ are approximately Gamma-distributed (cf. par. 456). In [KiSc08], Sect. 5, the shape parameter α and the rate parameter β were estimated to $\widetilde{\alpha} = 3.0949$ and $\widetilde{\beta} = 0.0240$: Furthermore, the mean length between successive 0-1-crossings, $E(T_j)$, was ≈ 128.85 ns, and the standard deviation ≈ 72.9 ns.

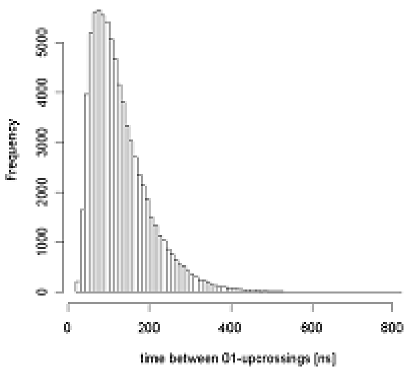

Figure 14: Empirical distribution of the time intervals between successive 0-1-crossings (in ns) [KiSc08], Fig. 3

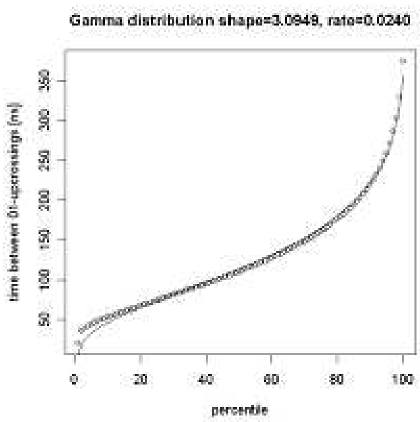

Figure 15: Percentiles of the Gamma distribution (curve) vs. the observed percentiles (circles) of the time intervals between successive 0-1-crossings (in ns) [KiSc08], Fig. 3

957 Fig. 16 and Fig. 17 plot the power spectrum and the autocorrelation function of the output signal of the amplifier.

958 Fig. 18 and Fig. 19 show typical output curves of the operational amplifier within time intervals of 1ns (resolution: 8 bits).

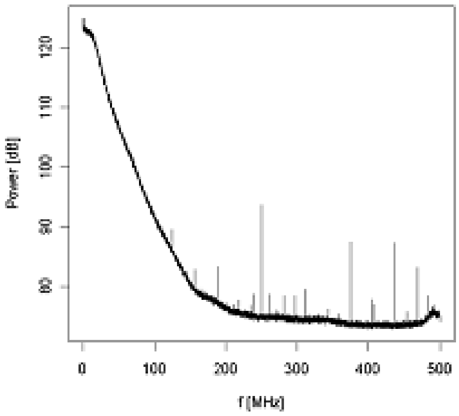

Figure 16: Mean power spectrum of the output of the amplifier (low amplification), created by W. Killmann

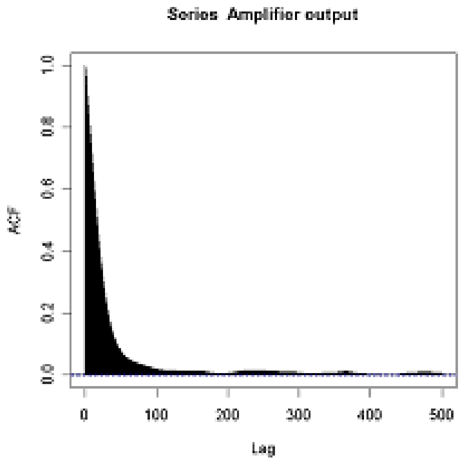

Figure 17: Autocorrelation of the amplified difference of noise voltages (maximum amplification, time in ns), created by W. Killmann

The term w_0 denotes the time of the first 0-1-crossing after $t = 0$ (when the observation of the raw random numbers begins). The term z_n denotes the index of the first 0-1-crossing that follows after time $s_n = ns$ when the clock has latched the n^{th} time. Furthermore, $w_n := w_0 + t_1 + \cdots t_{z_n} - s_n$ equals the time interval from latching time s_n of the second flip-flop to the next 0-1-crossing. In particular, 959

$$w_0 + t_1 + \cdots + t_{z_n - 1} \leq s_n < w_0 + t_1 + \cdots + t_{z_n} \,. \tag{5.99}$$

The equations (5.100) to (5.103) show relations between several random variables 960

$$\begin{align} & T_1, T_2, \ldots \quad \text{are stationary and } q\text{-dependent}\,, \tag{5.100} \\ & Z_n := \min\{m \in \mathbb{N}_0 \mid W_0 + T_1 + T_2 + \ldots + T_m > s_n\}\,, \tag{5.101} \\ & R_n := Z_n - Z_{n-1}\,, \tag{5.102} \\ & W_n := W_0 + T_1 + \cdots + T_{Z_n} - s_n \,. \tag{5.103} \end{align}$$

The relations (5.100) to (5.103) fit for other PTRNG designs, too. This would be the case if we would replace the two noisy diodes by a single noisy diode, and Example 3.5 in [Schi09b] considers a physical noise source with two independent ring oscillators. The distribution of the random variables $T_1, T_2, \ldots$ and thus the distribution of $R_1, R_2, \ldots$ and $Y_1, Y_2, \ldots$, may vary significantly for different PTRNG designs. Thus, it is profitable to study the system of random variables that is defined by (5.100) to (5.103), under general (weak) assumptions as well as for the specific distribution of the T_j, e.g., for iid or Markovian random variables $T_1, T_2, \ldots$. For our design $q \leq 1$. 961

A special feature of this PTRNG design is that under mild assumptions, the sequence $T_1, T_2, \ldots$ 962

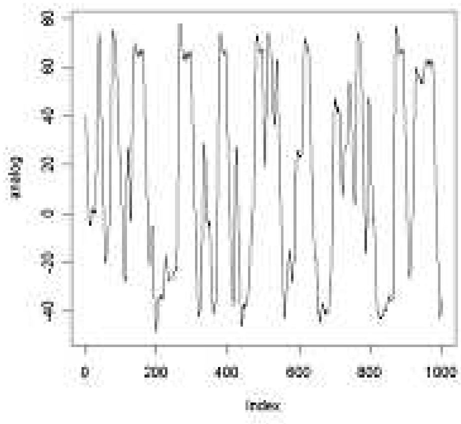

Figure 18: Output signal of the operational amplifier (low amplification), time-scale in ns, created by W. Killmann

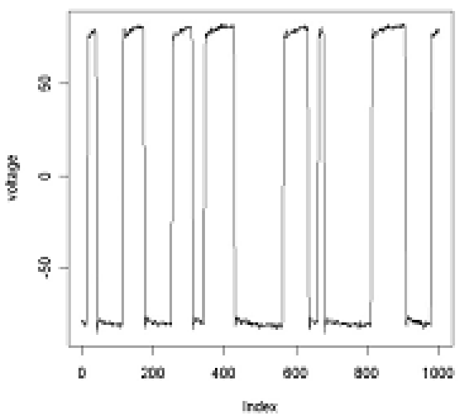

Figure 19: Output signal of the operational amplifier (maximum amplification), time-scale in ns, created by W. Killmann

'inherits' the stationarity property to other random variables; for details see [KiSc08], Lemma 1 and Assumption 1. In particular, the random variables

$$(T_j)_{j\in\mathbb{N}}\,,\ \ (R_j)_{j\in\mathbb{N}}\,,\ \ (W_j)_{j\in\mathbb{N}_0}\,,\ \ (R_j(\bmod 2))_{j\in\mathbb{N}}\,,\ \ \text{and}\ \ (Y_j)_{j\in\mathbb{N}}\ \ \text{are stationarily distributed.} \tag{5.104}$$

The stationarity suggests the analysis of the autocovariance and autocorrelation of these stochastic processes.

963 [definition] The terms

$$G_T(u) := \mathrm{Prob}(T_j \le u) \quad \text{and} \quad G_W(u) := \mathrm{Prob}(W_j \le u) \tag{5.105}$$

denote the cumulative distribution functions of the random variables T_j and W_j. Furthermore, for $u \in (0,\infty)$ the random variable

$$V_{(u)} := \inf\left\{\tau\in\mathbb{N} \mid \sum_{j=1}^{\tau+1} T_j > u\right\} = \sup\left\{\tau\in\mathbb{N} \mid \sum_{j=1}^{\tau} T_j \le u\right\} \tag{5.106}$$

quantifies the number of random 0-1-crossings in the interval $(0,u]$ if $W_0 \equiv 0$. The paragraphs below summarize important results from [KiSc08], Lemma 2 and Theorem 1.

964 For $k \ge 1$ we have

$$\mathrm{Prob}(V_{(u)} = k) = \mathrm{Prob}\left(T_1 + \cdots + T_k \le u\right) - \mathrm{Prob}\left(T_1 + \cdots + T_{k+1} \le u\right)\,. \tag{5.107}$$

965 The cycle length s should be 'large' compared to the mean length between two 0-1-crossings where the quantitative meaning of 'large' also depends on the generalized variance of the T_j. Otherwise,

the entropy of the random variables R_j might be very small, in particular, if the cycle length s is close to a small integer multiple of μ. Even if the parameter s would be selected in the middle of two integer multiples of μ, such a design would be rather sensitive to a variation of the parameter μ. The internal random numbers depend only on $R_j (\bmod 2)$, the least significant bit of R_n. However, it is advisable to select the cycle length s so large that the distribution of R_j has several probable outcomes around the ratio s/μ. This provides distributions which are more robust against deviations of the parameters.

Since $T_1, T_2, \ldots$ are assumed to be stationary and q-dependent (with $q \leq 1$, cf. par. 953) and since $s \gg \mu$, the CLT may be applied to the right-hand probabilities of (5.107) so that 966

$$\text{Prob}\left(\frac{T_1 + \cdots + T_k - k\mu}{\sqrt{k}\sigma} \leq x\right) \longrightarrow_{k\to\infty} \Phi(x) \quad \text{for } x \in \mathbb{R}. \tag{5.108}$$

The variance σ^2 is computed by (4.44) with $q = 1$, while Φ denotes the cumulative distribution function of the standard normal distribution (cf. par. 489, (4.35)). The mathematical background is sketched in pars. 496 and 497. We follow [KiSc08].

Let $u = v\mu$ with $v \gg 1$. By (5.107) and (5.108) we obtain 967

$$\text{Prob}\left(V_{(v\mu)} = k\right) \approx \Phi\left(\frac{v-k}{\sqrt{k}} \cdot \frac{\mu}{\sigma}\right) - \Phi\left(\frac{v-(k+1)}{\sqrt{k+1}} \cdot \frac{\mu}{\sigma}\right) \quad \text{for } k \geq 1 \tag{5.109}$$

$$\text{Prob}\left(V_{(v\mu)} = 0\right) \approx 1 - \Phi\left((v-1)\frac{\mu}{\sigma}\right). \tag{5.110}$$

Interestingly, the distribution of the random variable $V_{(v\mu)}$ (or more precisely, its normal approximation) depends only on the ratios μ/σ and $u/\mu = v$ but not on the absolute values of the parameters $\mu, \sigma^2, u = v\mu$. The mass of $V_{(v\mu)}$ is essentially concentrated on the values k around $k \approx v$. Since $s \gg \mu$ the approximation error should be negligible.

[iid case] If the random variables $T_1, T_2, \ldots$ are iid, it is well-known (cf. par. 510) that then 968

$$G_W(x) := \text{Prob}(W_n \leq x) = \frac{1}{\mu}\int_0^x (1 - G_T(u))\, du\,. \tag{5.111}$$

If $G_T(\cdot)$ is continuous (or equivalently, if $\text{Prob}(T_1 = y) = 0$ for all $y \in [0,\infty)$), then $G_W(\cdot)$ has density $g_W(x) := (1 - G_T(x))/\mu$.

Equations (5.112) and (5.113) provide an expression for the k^{th} moment for the sum $R_1 + R_2 + \cdots + R_j$). 969

$$E((R_1 + \cdots + R_j)^k) = \int_0^{js} E((V_{(js-u)} + 1)^k \mid W_0 = u)\, G_W(du) \tag{5.112}$$

$$\approx \int_0^{js} E((V_{(js-u)} + 1)^k)\, G_W(du) \quad \text{for each } k \in \mathbb{N} \tag{5.113}$$

with equality for iid random variables T_j. The term '+1' in (5.112) and (5.113) is needed because the random variables $V_{(u)}$ do not consider the origin ($t = 0$). For $j \geq 1$ the stationarity of the R_j implies

$$E((R_1 + \ldots + R_j)^2) = jE(R_1^2) + 2\sum_{i=2}^{j}(j+1-i)E(R_1 R_i)\,. \tag{5.114}$$

Beginning with $E(R_1^2)$ and then adding in (5.112), successively the random variables $R_2, R_3, \ldots$ one obtains $E(R_1R_2)$, $E(R_1R_3), \ldots$ in terms of expressions which are already known.

970 Under mild regularity assumptions on the random variables $T_1, T_2, \ldots$, heuristic arguments provide the inequality

$$\begin{aligned} H(Y_{n+1} \mid Y_1, \ldots, Y_n) &= H(R_{n+1}(\bmod 2) \mid R_1(\bmod 2), \ldots, R_n(\bmod 2)) \\ &\geq \min\{H(V_{(s-u)}(\mathrm{mod} 2)) \mid u \in [0, \mu + a\sigma)\} G_W(\mu + a\sigma). \end{aligned} \quad (5.115)$$

where $a > 0$ should be selected such that $G_W(\mu + a\sigma) \approx 1$. The idea of the min-operation is to consider the worst-case (depending on W_n). Apart from the impact on the time remaining until the next latch by the regular clock, potential dependencies between W_n and R_{n+1} are not considered. Due to the preceding such dependencies, if existent at all, should be rather small. If the random variables $T_1, T_2, \ldots$ are (almost) iid, easier formulae are derived below.

971 For the raw random numbers we obtain

$$\mathrm{Prob}(R_{n+1} = k) \approx \int_0^s \mathrm{Prob}(V_{(s-u)} = k-1)\, G_W(du) \ \text{ for } k \in \mathbb{N}_0 \quad (5.116)$$

$$H(R_{n+1}(\bmod 2)) \geq H(R_{n+1}(\bmod 2) \mid W_n) \approx \int_0^s H(V_{(s-u)}(\bmod 2))\, G_W(du) \quad (5.117)$$

If the random variables $T_1, T_2, \ldots$ are iid in (5.116) and (5.117), '$\approx$' signs can be replaced by '='.

972 [Case: $T_1, T_2, \ldots$ are iid] If the random variables $T_1, T_2, \ldots$ are iid, then $(W_{n-1}, R_n)_{n \in \mathbb{N}}$ defines a Markov chain on the state space $\mathbb{R}_+ \times \mathbb{N}_0$. In this case $W_0, T_1, T_2, \ldots$ induces a *stationary renewal process* $Z'(t) := \inf\{k \mid W_0 + T_1 + \cdots + T_k > t\}$, where t ranges in $[0, \infty)$; cf. pars. 510 to 512. In particular, $Z_n = Z'(sn)$. Even for iid random variables $T_1, T_2, \ldots$ the random variables $R_1, R_2, \ldots$ are usually not iid because large R_n (i.e., many 0-1-crossings in the n^{th} interval) makes it plausible that the last 0-1-crossing has occurred shortly before the end of this interval and thus that W_n is likely to be 'large'. Hence, R_n and R_{n+1} are weakly negatively correlated. For deeper mathematical analysis, see Subsect. 5.4.3, pars. 1028 ff.
However, if $s \gg \mu$ (as recommended) and if σ/μ is not 'small', the dependency between R_n and R_{n+1} should be small. Table 1 in [KiSc08] underlines that this is the case for the evaluated design.

973 If the sequence $T_1, T_2, \ldots$ is iid and the random variables $R_1, R_2, \ldots$ are 'almost' independent (since $\mu \ll s$), then

$$H(Y_{n+1} \mid Y_0, \ldots, Y_n) \approx \int_0^s H(V_{(s-u)}(\bmod 2))\, G_W(du) \ \text{ for all } n \in \mathbb{N}. \quad (5.118)$$

If $G_T(\cdot)$ is continuous (may be assumed here; cf. par. 953), then (5.118) reads

$$H(Y_{n+1} \mid Y_0, \ldots, Y_n) \approx \int_0^s H(V_{(s-u)}(\bmod 2))\, \frac{1}{\mu}(1 - G_T(u))\, du. \quad (5.119)$$

974 [iid case] For iid T_j (5.111) provides an explicit formula for the cumulative distribution function $G_W(\cdot)$, and if G_T is continuous, also for the density of W_n. If $T_j \sim \gamma_{\alpha,\beta} \cdot \lambda$ (par. 956), then

Fubini's Theorem, $\mu = \alpha/\beta$ (par. 456), and the properties of the Gamma function imply that

$$E\left(W_n^m\right) = \int_0^\infty u^m \frac{1}{\mu} \int_u^\infty \gamma_{\alpha,\beta}(v)\,dv\,du = \int_0^\infty \int_0^v u^m \frac{1}{\mu} \frac{\beta^\alpha}{\Gamma(\alpha)} v^{\alpha-1} e^{-\beta v}\,du\,dv =$$
$$\int_0^\infty \frac{1}{m+1} v^{m+1} \frac{1}{\mu} \frac{\beta^\alpha}{\Gamma(\alpha)} v^{\alpha-1} e^{-\beta v}\,dv = \int_0^\infty \frac{1}{m+1} \frac{1}{\mu} \frac{\beta^{\alpha+m+1}\Gamma(\alpha+m+1)}{\beta^{m+1}\Gamma(\alpha)\Gamma(\alpha+m+1)} v^{\alpha+m+1-1} e^{-\beta v}\,dv =$$
$$\frac{1}{m+1} \frac{1}{\mu} \frac{\Gamma(\alpha+m+1)}{\beta^{m+1}\Gamma(\alpha)} \int_0^\infty \gamma_{\alpha+m+1,\beta}(v)\,dv = \frac{1}{m+1} \frac{\beta}{\alpha} \frac{(\alpha+m)\cdots\alpha}{\beta^{m+1}} \cdot 1 =$$
$$\frac{1}{(m+1)\beta^m} (\alpha+m)\cdots(\alpha+1) \quad \text{for } m \geq 1\,. \tag{5.120}$$

As an immediate consequence we obtain

$$E(W_n) = \frac{\alpha+1}{2\beta} \quad \text{and} \quad \operatorname{Var}(W_n) = \frac{(\alpha+2)(\alpha+1)}{3\beta^2} - \frac{(\alpha+1)^2}{4\beta^2} = \frac{\alpha^2+6\alpha+5}{12\beta^2}\,. \tag{5.121}$$

Subsect. 5.4.3 provides a thorough analysis of a generic stochastic model with iid random variables $T_1, T_2, \ldots$, allowing the computation of the joint distribution of $(R_1, \ldots, R_m)$ for $m \geq 1$. Therefrom, both the Shannon entropy and the min-entropy of the the random variables $(R_1(\text{mod}2), \ldots, R_m(\text{mod}2))$ can be computed. Under suitable circumstances these results can also be applied if the T_j are only weakly dependent; cf. Subsect. 5.4.3. 975

[stochastic model] Essential for the evaluation is the stochastic behavior of the (random) intervals between consecutive 0-1-crossings. We described these random intervals by random variables $T_1, T_2, \ldots$. Our first task was to characterize these random variables. By technical arguments that consider the properties of the noisy diodes and of the operational amplifier, and supported by experiments, evidence was given in [KiSc08] that the random variables $T_1, T_2, \ldots$ are stationary and q-dependent. It was concluded that q can be selected ≤ 1 with only small dependencies between T_j and T_{j+1}. The one-dimensional distribution of the random variables T_j can be approximated by a Gamma distribution; cf. pars. 948, 951, 953, 956, 957, 958. Further analysis depends on these findings. 976

[stochastic model] A Gamma distribution depends on two parameters, the shape parameter α and the rate parameter β. Possibly, it can be shown that the random variables $T_1, T_2, \ldots$ are 'almost' iid (i.e., 0-dependent). This conclusion (as far as applicable) would require deeper analysis of the design but would simplify the further analysis to some degree. 977

[stochastic model] Actually, the stochastic model should comprise a family of admissible distributions of the raw random number variables $R_1, R_2, \ldots$, called 'virtual raw random numbers' in [KiSc08] because they do not 'really' appear. The specification of this family of distributions is not easy, not even if the T_j are iid (par. 972). It should be noted that par. 971, formula (5.116), specifies the one-dimensional distribution of R_n. If $s \gg \mu$ (as recommended), the dependencies between neighboring R_n should be weak, and it is rather likely that the (mod 2) operation reduces dependencies additionally . In particular, the virtual raw random numbers $R_1, R_2, \ldots$ and $R_1(\text{mod } 2), R_2(\text{mod } 2), \ldots$ are stationary (5.104). 978

[stochastic model] On the other hand, the relevant conclusions from above are closely connected to the random variables $V_{(u)}$; cf. pars. 969, 970, 971, 973. In particular, this allows the determi- 979 nation of lower entropy bounds for the internal random numbers (cf. (5.115), (5.118), (5.119)). Hence, we consider the 'auxiliary' random variables $V_{(u)}$; cf. par. 651.

980 [entropy] Tab. 1 in [KiSc08] summarizes the results for several parameter sets. Recall that $\tilde{\mu} = 128.85$ ns (estimate of $E(T_j)$; cf. par. 956). For the most conservative design parameter $s = 15.017\tilde{\mu}$, the conditional Shannon entropy per raw random bit, $R_j(\text{mod }2)$, $(R_{n+1}(\text{mod }2) \mid R_{n+1}(\text{mod }2), \ldots, R_{n+1}(\text{mod }2))$, is assumed to be $> 1 - 10^{-4}$; cf. [KiSc08], Sect. 5. This gives an output rate of raw random bits (= output rate of internal random numbers) of a little more than 500 kBit/sec.

981 [stochastic model] The distribution of the random variable $V_{(u)}$ depends on the parameters $u = v\mu$ and μ/σ, or equivalently, on $v = u/\mu$ and μ/σ, cf. (5.109) and (5.110). The distribution of the raw random numbers $R_1, R_2, \ldots$ as well as of $R_1(\text{mod }2), R_2(\text{mod }2), \ldots$, and their average conditional entropy depends on the ratios s/μ and μ/σ.

982 [stochastic model] This means that the stochastic model of the raw random numbers is a 2-parameter model with parameters $(s/\mu, \mu/\sigma) \in (0, \infty)^2$. A central task of the evaluation is to specify subsets $A_{real}, A_{good}, A_{bad} = (0, \infty)^2 \setminus A_{good} \subseteq (0, \infty)^2$; cf. Subsect. 4.5.3. In particular, the parameters in A_{real} and A_{good} provide enough conditional entropy; cf. (5.115), (5.118), (5.119). The online test shall detect if the true parameters leave the subset A_{real} when the PTRNG is in operation.
Note: If s is fixed, the 2-parameter-family of admissible distributions can alternatively be parametrized by $(\mu, \mu/\sigma)$, (μ, σ^2), or (μ, σ).

983 [online test] The distribution of the raw random numbers (and thus the lower entropy bound for the internal random numbers) depends on the ratios s/μ and μ/σ. The online test shall detect when these values leave the set of appropriate parameters A_{good}. General considerations are found in [KiSc08], Sect. 6. We do not deepen this aspect here but refer to Subsect. 5.4.3.

5.4.3 Sampling events with iid intermediate time intervals – Design A

984 This subsection does not develop and analyze the stochastic model for a concrete design of a physical noise source. Instead, thorough analysis of a generic stochastic model is provided that potentially fits several different designs; cf. pars. 992 and 1012. Unlike (5.118) and (5.119) this subsection also covers scenarios for which the random variables R_j are far from being independent.
Note: The developer may apply the results from this subsection but, of course, has to give evidence that the stochastic model defined below indeed fits the design under evaluation.

985 In the following we assume that a physical noise source counts (design-specific) events. The time intervals between two successive events are denoted by $t_1, t_2, \ldots$. The integers r_j denote the number of events within the interval $I_j := ((j-1)s, js]$, where s denotes the (fixed) length of the sampling intervals, e.g., the cycle length of a stable clock. Furthermore,

$$y'_j = r_j(\text{mod }2) \tag{5.122}$$

Note: The definition of the binary random number y'_j differs from that of y_j in Subsect. 5.4.2, but the results on the entropy of the random variables Y'_j can easily be transferred to the random variables $Y_j = Y_0 + Y_1' + \cdots + Y_j' (\bmod 2)$.

The first goal is to determine the joint distribution of random variables $R_1, \ldots, R_m$. Therefrom, the joint distribution of random variables $Y_1', \ldots, Y_m'$ can be deduced. This allows determining the joint entropy and conditional entropy for both Shannon entropy and min entropy. Later, we develop an effective online test. 986

Example: In Subsect. 5.4.2 the design-specific events are 0-1-crossings of a Schmitt trigger. Likewise, such events could be radioactive decays as in Subsect. 5.4.5. 987

Finally, we are interested in the intermediate random numbers $y_1', y_2', \ldots$. However, with regard to the online test, we recommend counting the raw random numbers $r_1, r_2, \ldots$ because r_j contains much more information than y_j'. After the raw random number r_j has been read, the counter is reset to 0. 988

As already mentioned above the applicant / the developer has to show that this generic stochastic model fits the concrete physical noise source. In this subsection, the stochastic model assumes that sampling is ideal in the sense that $r_1, r_2, \ldots$ equal the exact numbers of events that have occurred in the intervals $I_1, I_2, \ldots$. In real-world designs, occasional detection errors can occur, e.g., because an event occurs when a counting flip-flop is in the state of metastability because the setup-time or hold-time restrictions are undercut. If such detection errors occur rarely and if the entropy proof is not based on the metastability (e.g., of the flip-flop), they may be neglected in the stochastic model. Furthermore, one may assume that occasional detection errors (caused by metastability) even slightly increase the entropy of the random numbers. 989

The most critical task within the evaluation of the physical noise source is to verify that the random variables $T_1, T_2, \ldots$ fulfill (at least approximately) the iid assumption. The physical effects contributing to the variance of T_j should be investigated. A favorable scenario is, for example, if the variance is essentially caused by thermal noise or shot noise. If (low-frequent) flicker noise has relevant impact, the situation becomes more complicated. Here, the Allan variance could be used to estimate the size of the 'useful jitter'. Perhaps the results from this subsection cannot directly be applied, but then possibly the basic ideas and strategies might be used and adjusted. 'Worst-case analysis', assuming the least favorable assumptions, might be necessary in place of the integrals developed below, presumably losing some information. For appropriate designs proving a lower entropy bound should be possible. 990

[iid assumption] We assume that the time lengths $t_1, t_2, \ldots$ can be viewed as realizations of iid random variables $T_1, T_2, \ldots$. As already noted in par. 972 the random variables $Z'(t) := \inf\{k \mid W_0 + T_1 + \cdots + T_k > t\}$ form a stationary renewal process, where t ranges in $[0, \infty)$; cf. pars. 1028 ff. 991

The iid assumption from par. 991 applies (at least approximately) to several noise source designs. Example (cf. par. 987): Subsect. 5.4.2 (under the iid assumption; cf. par. 972), radioactive decays (cf. Subsect. 5.4.5, par. 1055, although there a different sampling mechanism is used). 992

For each $x \in \mathbb{R}$ there exist unique $k \in \mathbb{Z}$ and $b \in [0, s)$ such that $x = ks + b$. We write $x (\bmod s) = b$. Assume that the real-valued random variables Y_1 and Y_2 are independent. If 993

Y_1 is uniformly distributed on $[0, s)$, then $Y_1 + Y_2 (\bmod s)$ is uniformly distributed on $[0, s)$, too, regardless of the distribution of Y_2.

994 Let $S_j = T_1 + \cdots + T_j (\bmod s)$. Under weak assumptions on the distribution of the T_j, the random variables $S_1, S_2 \ldots$ converge exponentially fast to the uniform distribution on $[0, s)$.
Note: It suffices that the distribution of T_j has a density that is > 0 on some interval $I \subseteq [0, s)$.
Note: We may assume that the PTRNG has started some time before (at time $-Js$ for some $J > 0$), and that $S_0, S_1, \ldots$ are uniformly distributed on $[0, s)$ (equilibrium state).

995 We use the same notation as in Subsect. 5.4.2. To simplify reading we repeat the definitions. The random variable W_0 describes the (random) time when the first event is detected after time $t = 0$ (when the observation of the raw random number starts). Furthermore,

$$T_1, T_2, \ldots \quad \text{are iid}\,, \tag{5.123}$$
$$Z_n := \min\{m \in \mathbb{N}_0 \mid W_0 + T_1 + T_2 + \ldots + T_m > s_n\}\,, \tag{5.124}$$
$$R_n := Z_n - Z_{n-1}\,, \tag{5.125}$$
$$W_n := W_0 + T_1 + \cdots + T_{Z_n} - s_n\,. \tag{5.126}$$

Unlike in Subsect. 5.4.2, par. 960, the random variables $T_1, T_2, \ldots$ are not only stationarily distributed and q-dependent but even assumed to be iid.

996 Of course, as in Subsect. 5.4.2, par. 962 (which covers a more general case), the random variables

$$(T_j)_{j\in\mathbb{N}}\,,\quad (R_j)_{j\in\mathbb{N}}\,,\quad (W_j)_{j\in\mathbb{N}_0}\,,\quad \text{and}\quad \left(Y'_j = R_j (\bmod 2)\right)_{j\in\mathbb{N}} \quad \text{are stationarily distributed.} \tag{5.127}$$

Note: The renewal process $Z'(t)$ is stationary (cf. par. 991), and thus the sequence $R_1, R_2, \ldots$ is stationary (cf. par. 510, it is $R_n = Zn - Z_{n-1} = Z'(ns) - Z'((n-1)s)$). This in particular implies the stationarity of $Y'_1, Y'_2, \ldots$, while the stationarity of $W_0, W_1, \ldots$ follows from (4.56).

997 [Assumption] The distribution of the random variables $T_1, T_2, \ldots$ has density (to be mathematically precise: a Lebesgue density) $g(\cdot)$. If $G_T(\cdot)$ denotes the cumulative distribution function of T_j then W_j has density $g_W(\cdot) = \frac{1}{\mu}(1 - G_T(\cdot))$ (par. 510).
Note: In particular, $\mathrm{Prob}(W_j > 0) = 1$ for all $j \in \mathbb{N}$.
Note: The case that the distribution of T_j has a density constitutes the most relevant case for applications. We mention that similar results can be derived if the random variables $T_1, T_2, \ldots$ do not have a density $g(\cdot)$ (although with greater mathematical efforts). If the random variables T_j are discrete, the integrals below turn into sums.

998 For each integer $\ell \geq 1$ the term $g^{*(\ell)}(\cdot)$ denotes the ℓ-fold convolution of the density $g(\cdot)$. In particular, $g^{*(1)}(\cdot) = g(\cdot)$.
Note: For each $t \in \mathbb{N}_0$ and $\ell \in \mathbb{N}$ the sum $T_{t+1} + \cdots + T_{t+\ell}$ has density $g^{*(\ell)}(\cdot)$; cf. pars. 1010 and 1011.

999 For $m \in \mathbb{N}$ and $k_1, \ldots, k_m \in \mathbb{N}_0$ it is

$$\begin{aligned}
&\mathrm{Prob}\left(R_1 \leq k_1, R_1 + R_2 \leq k_1 + k_2, \ldots, R_1 + \cdots + R_m \leq k_1 + \cdots + k_m\right) = \\
&\mathrm{Prob}\left(W_0 + T_1 + \cdots + T_{k_1} > s, W_0 + T_1 + \cdots + T_{k_1+k_2} > 2s, \ldots,\right. \\
&\qquad \left. W_0 + T_1 + \cdots + T_{k_1+\cdots+k_m} > ms\right)
\end{aligned} \tag{5.128}$$

In the next paragraphs we develop integral representations for the joint probabilities of $(R_1, \ldots, R_m)$. 1000

$[k_1, \ldots, k_m > 0]$ If $k_1, \ldots, k_m > 0$ we obtain from (5.128) the integral representation 1001

$$\begin{aligned}&\operatorname{Prob}(R_1 \leq k_1, R_1 + R_2 \leq k_1 + k_2, \ldots, R_1 + \cdots + R_m \leq k_1 + \cdots + k_m) = \\ &\int_{ms}^{\infty} \int_{(m-1)s}^{\infty} \cdots \int_{s}^{\infty} \int_{0}^{\infty} g^{*(k_m)}(u_m - u_{m-1}) g^{*(k_{m-1})}(u_{m-1} - u_{m-2}) \cdots \\ &\qquad g^{*(k_1)}(u_1 - u_0) g_W(u_0)\, du_0 du_1 \cdots du_{m-1} du_m \end{aligned} \tag{5.129}$$

Note: The term du_0 belongs to '$\int_0^\infty$'.
Note: Since $g_W(u), g^{*(\ell)}(u) = 0$ for $u < 0$ $(\ell \geq 1)$, the integrand does not contribute to the integral unless $0 \leq u_0 \leq u_1 \leq \cdots \leq u_m$. Likewise, the lower integration boundaries js could be replaced by $\max\{js, u_{j-1}\}$ (for $j = 1, \ldots, m$).

If $k_j = 0$ for one or several indices j, the integral representation (5.129) has to be adjusted. We begin with an example. 1002

[Example: $m = 4$, $k_3 = 0$] Let $(k_1, k_2, k_3, k_4) = (3, 7, 0, 2)$. By (5.128) we obtain 1003

$$\begin{aligned}&\operatorname{Prob}(R_1 \leq 3, R_1 + R_2 \leq 10, R_1 + R_2 + R_3 \leq 10, R_1 + R_2 + R_3 + R_4 \leq 12) = \\ &\operatorname{Prob}(W_0 + T_1 + \cdots + T_3 > s, W_0 + T_1 + \cdots + T_{10} > 2s, \\ &\qquad W_0 + T_1 + \cdots + T_{10} > 3s, W_0 + T_1 + \cdots + T_{12} > 4s) \end{aligned} \tag{5.130}$$

Since the condition $(W_0 + T_1 + \cdots + T_{10} > 3s)$ implies the weaker condition $(W_0 + T_1 + \cdots + T_{10} > 2s)$, this saves one integral in the integral representation. In particular,

$$\begin{aligned}&\operatorname{Prob}(R_1 \leq 3, R_1 + R_2 \leq 10, R_1 + R_2 + R_3 \leq 10, R_1 + R_2 + R_3 + R_4 \leq 12) = \\ &\operatorname{Prob}(W_0 + T_1 + \cdots + T_3 > s, W_0 + T_1 + \cdots + T_{10} > 3s, W_0 + T_1 + \cdots + T_{12} > 4s) = \\ &\int_{4s}^{\infty} \int_{3s}^{\infty} \int_{s}^{\infty} \int_{0}^{\infty} g^{*(2)}(u_4 - u_3) g^{*(7)}(u_3 - u_1) g^{*(3)}(u_1 - u_0) g_W(u_0)\, du_0 du_1 du_3 du_4 \end{aligned} \tag{5.131}$$

Note: The integral '$\int_{3s}^{\infty} \ldots g^{*(k_2)}(u_3 - u_1) \cdots du_3$' replaces '$\int_{3s}^{\infty} \int_{2s}^{\infty} \ldots g^{*(k_3)}(u_3 - u_2) g^{*(k_2)}(u_2 - u_1) \ldots du_3 du_2$' (compared with the integral representation (5.129) for $k_1, k_2, k_3, k_4 > 0$).

$[k_j = 0$ for at least one index $j]$ In this and in the following paragraph, we generalize the insights from the preceding example. The goal is to derive an integral representation corresponding to (5.129). At first, isolated subsequences of 0's in $k_1, \ldots, k_m$ are identified. Each of these subsequences is treated as explained in par. 1005. 1004
Example: Let $(k_1, k_2, k_3, k_4, k_5, k_6, k_7) = (9, 0, 0, 3, 1, 5, 0)$. There are two subsequences of 0's, namely k_2, k_3 and k_7.

$[k_j = 0$ for at least one index $j]$ In this paragraph we consider the impact of the isolated subsequences of 0's (cf. par. 1004) on the integral representation. We assume that $k_j, \ldots, k_{j+t} = 0$. We distinguish four cases $(t \geq 0)$. 1005

Case (a) $1 < j$, $j + t < m$ and $k_{j-1}, k_{j+t+1} > 0$.
Then $W_0 + T_1 + \ldots + T_{k_1 + \cdots + k_{j-1}} > (j + t)s$.

Impact on the integral representation (5.129), compared to the case $k_1, \ldots, k_m > 0$:
The integrals '$\int_{(j+t)s}^{\infty} \cdots g^{*(k_{j-1})}(u_{j+t} - u_{j-2})\, du_{j+t}$' replace the integrals
$\int_{(j+t)s}^{\infty} \cdots \int_{(j-1)s}^{\infty} \cdots g^{*(k_{j+t})}(u_{j+t} - u_{j+t-1}) \cdots g^{*(k_{j-1})}(u_{j-1} - u_{j-2}) \ldots du_{j-1} \cdots du_{j+t}$.

Case (b) $1 = j$, $1 + t < m$ and $k_{t+2} > 0$.
Then $W_0 > (t+1)s$, $W_0 + T_1 + \ldots + T_{k_{t+2}} > (t+2)s$.
Impact on the integral representation (5.129), compared to the case $k_1, \ldots, k_m > 0$:
The integrals '$\int_{(t+2)s}^{\infty} \int_{(t+1)s}^{\infty} \cdots g^{*(k_{t+2})}(u_{t+2} - u_0) g_W(u_0)\, du_{t+2} du_0$' replace the integrals
$\int_{(t+2)s}^{\infty} \cdots \int_0^{\infty} \ldots g^{*(k_{t+2})}(u_{t+2} - u_{t+1}) \cdots g_W(u_0)\, du_0 \cdots du_{t+2}$.

Case (c) $1 < j$, $j + t = m$ and $k_{j-1} > 0$
Then $W_0 + T_1 + \ldots + T_{k_{j-1}} > ms$.
Impact on the integral representation (5.129), compared to the case $k_1, \ldots, k_m > 0$:
The integrals '$\int_{ms}^{\infty} \ldots g^{*(k_{j-1})}(u_m - u_{j-2})\, du_m$' replace the integrals
$\int_{ms}^{\infty} \cdots \int_{js}^{\infty} \ldots g^{*(k_m)}(u_m - u_{m-1}) \cdots g^{*(k_j)}(u_j - u_{j-1})\, du_j \cdots du_m$.

Case (d) $1 = j$, $j + t = m$ (i.e., $k_1 = \ldots, k_m = 0$).
Then $W_0 > ms$.
Impact on the integral representation (5.129), compared to the case $k_1, \ldots, k_m > 0$:
The integral '$\int_{jm}^{\infty} g_W(u_0)\, du_0$' replaces (5.129).

1006 So far, we have learned how to compute probabilities of the type $\text{Prob}(R_1 \leq k_1, \ldots, R_1 + \cdots + R_m \leq k_1 + \cdots + k_m)$. However, finally we are interested in probabilities of the type $\text{Prob}(R_1 = k_1, \ldots, R_m = k_m)$.

1007 Equation (5.132) provides the desired formula.

$$
\begin{aligned}
&\text{Prob}\,(R_1 = k_1, R_2 = k_2, \ldots, R_m = k_m) = \\
&\sum_{T \subseteq \{1,\ldots,m\}} (-1)^{|T|}\, \text{Prob}\,(R_1 \leq \ell_1, R_1 + R_2 \leq \ell_2, \ldots, R_1 + \cdots + R_m \leq \ell_m \mid \\
&\qquad\qquad \ell_j = k_1 + \cdots + k_j - 1 \text{ if } j \in T, \ell_j = k_1 + \cdots + k_j \text{ else}) \\
&\qquad\qquad\qquad \text{for } k_1, \ldots, k_m \geq 0
\end{aligned} \tag{5.132}
$$

We prove (5.132) by induction on m. For $m = 1$ (5.132) is $\text{Prob}\,(R_1 = k_1) = \text{Prob}\,(R_1 \leq k_1) - \text{Prob}\,(R_1 \leq k_1 - 1)$, which is correct. Assume that (5.132) is valid for all $1 \leq m' \leq m$. The inductive step can be verified as follows

$$
\begin{aligned}
&\text{Prob}\,(R_1 = k_1, R_2 = k_2, \ldots, R_m = k_m, R_{m+1} = k_{m+1}) = \\
&\text{Prob}\,(R_1 = k_1, R_2 = k_2, \ldots, R_m = k_m, R_1 + \cdots + R_{m+1} = k_1 + \cdots + k_{m+1}) = \\
&\text{Prob}\,(R_1 = k_1, R_2 = k_2, \ldots, R_m = k_m, R_1 + \cdots + R_{m+1} \leq k_1 + \cdots + k_{m+1}) - \\
&\quad \text{Prob}\,(R_1 = k_1, R_2 = k_2, \ldots, R_m = k_m, R_1 + \cdots + R_{m+1} \leq k_1 + \cdots + k_{m+1} - 1)
\end{aligned} \tag{5.133}
$$

We apply the induction hypothesis separately to the first m components of both summands of the last equation in (5.133). This leads to

$$\begin{aligned}
&\text{Prob}\,(R_1 = k_1, R_2 = k_2, \ldots, R_m = k_m, R_{m+1} = k_{m+1}) = \\
&\sum_{T\subseteq\{1,\ldots,m\}} (-1)^{|T|}\,\text{Prob}\,(R_1 \le \ell_1, R_1 + R_2 \le \ell_2, \ldots, R_1 + \cdots + R_{m+1} \le k_1 + \cdots + k_{m+1} \mid \\
&\qquad \ell_j = k_1 + \cdots + k_j - 1 \text{ if } j \in T, \ell_j = k_1 + \cdots + k_j \text{ else}) - \\
&\sum_{T\subseteq\{1,\ldots,m\}} (-1)^{|T|}\,\text{Prob}\,(R_1 \le \ell_1, R_1 + R_2 \le \ell_2, \ldots, R_1 + \cdots + R_{m+1} \le k_1 + \cdots + k_{m+1} - 1 \mid \\
&\qquad \ell_j = k_1 + \cdots + k_j - 1 \text{ if } j \in T, \ell_j = k_1 + \cdots + k_j \text{ else})
\end{aligned} \tag{5.134}$$

Actually, the right-hand probabilities of the second sum in (5.134) correspond to subsets $T' = T \cup \{m+1\} \subseteq \{1, \ldots, m+1\}$. Furthermore, since $|T'| = |T| + 1$, we can combine the last two sums of (5.134), which leads to

$$\begin{aligned}
&\text{Prob}\,(R_1 = k_1, R_2 = k_2, \ldots, R_m = k_m, R_{m+1} = k_{m+1}) = \\
&\sum_{T\subseteq\{1,\ldots,m,m+1\}} (-1)^{|T|}\,\text{Prob}\,(R_1 \le \ell_1, R_1 + R_2 \le \ell_2, \ldots, R_1 + \cdots + R_{m+1} \le \ell_{m+1} \mid \\
&\qquad \ell_j = k_1 + \cdots + k_j - 1 \text{ if } j \in T, \ell_j = k_1 + \cdots + k_j \text{ else})
\end{aligned} \tag{5.135}$$

This completes the proof of (5.132).

[Example] Equation (5.136) illustrates the general formula (5.132) for the special case $m = 2$. 1008

$$\begin{aligned}
&\text{Prob}\,(R_1 = k_1, R_2 = k_2) = \\
&\quad \text{Prob}\,(R_1 \le k_1, R_1 + R_2 \le k_1 + k_2) - \text{Prob}\,(R_1 \le k_1 - 1, R_1 + R_2 \le k_1 + k_2) - \\
&\quad \text{Prob}\,(R_1 \le k_1, R_1 + R_2 \le k_1 + k_2 - 1) + \text{Prob}\,(R_1 \le k_1 - 1, R_1 + R_2 \le k_1 + k_2 - 1)
\end{aligned} \tag{5.136}$$

[special cases] If $k_1, \ldots, k_m$ contains one or more 0s in the right-hand probabilities of (5.132), 'special cases' can occur. If $\ell_j = -1$ for some j, then the whole probability is 0. If $\ell_j > \ell_{j+1}$ then ℓ_j can be replaced by ℓ_{j+1} because the random variables R_j assume non-negative values. 1009

[convolution] The expectation and the variance of the sum $T_{t+1} + \cdots + T_{t+\ell}$ are $\ell E(T_j)$ and $\ell \text{Var}(T_j)$. 1010

[convolution densities, special cases] If the random variables $T_1, T_2, \ldots$ are iid normally distributed, the sum $T_{t+1} + \cdots + T_{t+\ell}$ is normally distributed, too (cf. par. 455). If the random variables $T_1, T_2, \ldots$ are iid Gamma distributed, the sum $T_{t+1} + \cdots + T_{t+\ell}$ is also Gamma distributed (par. 457). In both cases it is easy to determine the convolution densities $g^{*(\ell)}$. 1011

[convolution densities, CLT] In the general case (for arbitrary densities $g(\cdot)$), it can be difficult to provide exact expressions for the convolution densities $g^{*(\ell)}$. However, if the CLT applies to the relevant parameters k_j (cf. par. 1016), one can use normal densities for the computation of integrals (5.129). Furthermore, as an additional advantage, in such cases there is no need to determine the distribution of the T_j exactly. Instead, it suffices to estimate their expectation and variance. 1012

1013

[convolution densities, CLT] Whether the CLT applies to the densities $f^{*(\ell)}(\cdot)$ for $\ell \geq L_0$ (where L_0 is a suitable lower bound) has to be checked in each case. Of course, the 'closer' f is to a normal distribution the lower L_0 can be chosen. Note that the Berry-Esséen-Theorem (see, e.g., par. 494) considers the worst-case.

1014 [convolution densities, CLT] If the CLT applies to the T_j (as in Subsect. 5.4.2), the results from this subsection may also be applicable to noise sources for which the random variables $T_1, T_2, \ldots$ are time-locally stationary but show (weak) dependencies (but 'large' relevant values k_j). Recall that the CLT is rather robust and, e.g., applies to q-dependent and Markovian random variables (cf. pars. 496, 497, 498, 504, 505).

1015 Recall that $Y'_j = R_j (\bmod 2)$. Finally, we are interested in the distribution of $Y'_1, \ldots, Y'_m$. Obviously,

$$\operatorname{Prob}(Y'_1 = y'_1, \ldots, Y'_m = y'_m) = \sum_{k_j \equiv y'_j (\bmod 2) \text{ for } 1 \leq j \leq m} \operatorname{Prob}(R_1 = k_1, \ldots, R_m = k_m)$$
$$\text{for } y'_1, \ldots, y'_m \in \{0,1\} \qquad (5.137)$$

1016 In principle, the right-hand side (5.137) comprises infinitely many probabilities. However, if k_j is 'far' from the expectation $E(R_j) = \frac{s}{E(T)}$, the corresponding probabilities are negligible. (The quantitative meaning of 'far' depends on the distribution of the T_j, in particular on their variance.) With regard to (5.132) and (5.137), it seems reasonable to calculate probabilities $\operatorname{Prob}(R_1 \leq k_1, \ldots R_1 + \cdots + R_m \leq k_1 + \cdots + k_m)$ for relevant $(k_1, \ldots, k_m)$ first because these probabilities may be needed several times.

1017 Equation (5.137) provides a formula for the joint probability of the random variables $Y'_1, \ldots, Y'_m$. Applying Bayes's formula, we obtain the conditional probabilities

$$\operatorname{Prob}\left(Y'_m = y'_m \mid Y'_1 = y'_1, \ldots, Y'_{m-1} = y'_{m-1}\right) = \frac{\operatorname{Prob}(Y'_1 = y'_1, \ldots, Y'_m = y'_m)}{\operatorname{Prob}\left(Y'_1 = y'_1, \ldots, Y'_{m-1} = y'_{m-1}\right)} \qquad (5.138)$$

1018 Equations (5.137) and (5.138) allow calculating the joint entropy and the conditional entropy of the random variables $Y'_1, \ldots, Y'_m$. This concerns both Shannon entropy and min entropy.

1019 [Numerical example] Table 5 provides several numerical examples. In all cases the random variables T_j are assumed to be $N(\mu, \sigma^2)$-distributed. The figures were not gained by the evaluation in (5.129), (5.132), and (5.137). Instead, the random variables T_j, and implicitly the random variables R_j, were simulated (sample size N).

The conditional Shannon entropy is computed with formula (4.73) (with $m-1$ in place of m). The conditional min-entropy (last column of Table 5) applies the formula

$$H_{min}(Y'_m \mid Y'_1, \ldots, Y'_{m-1}) =$$
$$\min\{H_{min}(Y'_m \mid Y'_1 = y'_1, \ldots, Y'_{m-1} = y'_{m-1}) \mid y'_1, \ldots, y'_{m-1} \in \{0,1\}\}. \quad (5.139)$$

1020

Table 5: Simulation experiments (design type A): $T_j \sim N(\mu, \sigma^2)$, $\mu = 1.0$, sample size $N = 10,000,000$

$\left(\frac{s}{\mu}, \frac{\sigma}{\mu}\right)$	m	$\frac{H(Y_1',\ldots,Y_m')}{m}$	$H(Y_m' \mid Y_1',\ldots,Y_{m-1}')$	$\frac{H_{min}(Y_1',\ldots,Y_m')}{m}$	$H_{min}(Y_m' \mid Y_1',\ldots,Y_{m-1}')$
$(25, 0.2)$	4	0.99998	0.99998	0.9928	0.9914
$(10, 0.2)$	4	0.9907	0.9907	0.843	0.842
$(100, 0.1)$	4	0.99997	0.99997	0.9914	0.9905
$(10000, 0.01)$	4	0.99998	0.99998	0.9922	0.9915

Tab. 5 provides exemplary figures for pairs of parameters. An important question is how sensitive the corresponding entropy values are with regard to deviations of the parameters. To investigate this question, in Tab. 6 we considered subsets of parameters. Let

$$A_{[\alpha_1,\alpha_2,\beta_1,\beta_2]}(\mu,\sigma) := \{(\mu',\sigma') \mid \alpha_1\mu \le \mu' \le \alpha_2\mu, \beta_1\sigma \le \sigma' \le \beta_2\sigma\}, \quad \alpha_1,\beta_1 \le 1 \le \alpha_2,\beta_2 \tag{5.140}$$

Note: The sensitivity of the entropy values is a crucial feature for the robustness of the design of a physical noise source. Little sensitivity against deviations of the parameters is a desirable feature ('robust design'), which reduces the requirements on the online test, and also the protection against active attacks should become easier.

[Numerical example] Tab. 6 provides numerical examples for sets $A_{[0.9,1.1,0.9,1.1}(\mu,\sigma)$. As in Tab. 5 the random variables T_j are assumed to be $N(\mu,\sigma^2)$-distributed. Again, the figures were not gained by the evaluation of the formulae (5.129), (5.132), and (5.137). Instead, the random variables T_j, and implicitly the random variables R_j, were simulated (sample size N). We treated pairs of parameters (μ',σ) for which $\mu' \in \{0.9\mu, 0.95\mu, \mu, 1.05\mu, 1.10\mu\}$ and $\sigma' \in \{0.9\sigma, 0.95\sigma, \sigma, 1.05\sigma, 1.10\sigma\}$ The conditional Shannon entropy is computed with formula (4.73) (with $m-1$ in place of m). As for Tab. 5 we computed the conditional min-entropy for each pair (μ',σ') with formula (5.139). 1021
Note: The experiments confirm the intuitive understanding that $(\mu',\sigma') = (1.1\mu, 0.9\sigma)$ is the 'worst-case'.

[Numerical example, Gamma distribution] Subsect. 5.4.2 summarizes results from [KiSc08] on a physical noise source design that exploits two noisy diodes. In [KiSc08] it was shown that the (one-dimensional) distribution of the T_j can be approximated by a Gamma distribution with estimated shape parameter $\widetilde{\alpha} = 3.0949$ and estimated rate parameter $\widetilde{\beta} = 0.0240$. Furthermore, in [KiSc08] it was concluded that consecutive T_j should be 1-dependent and only weakly autocorrelated. Under the *idealized assumption* that the T_j are iid Gamma distributed with parameters $\widetilde{\alpha}$ and $\widetilde{\beta}$, simulations (as for Tabs. 5 and 6) show that the conditional min-entropy $H_{min(A)}(Y_4' \mid Y_1',\ldots,Y_3')$ is > 0.99 for all parameter sets considered in Tab. 1 of [KiSc08]. 1022

[stochastic model] The distribution of the variables $R_1, R_2, \ldots$ depends on the distribution of the iid random variables $T_1, T_2, \ldots$, and thus also on the number of parameters that specify the admissible distributions. 1023

[stochastic model] If the T_j are normally distributed or Gamma distributed, for example, their 1024

Table 6: Simulation experiments (design type A): $T_j \sim N(\mu, \sigma^2)$, $\mu = 1.0$, subsets $A_{[0.9,1.1,0.9,1.1}(\mu, \sigma)$, the index '(A)' indicates that sets are considered. The values in the upper line denote the average, the values in the lower line the worst-case, sample size $N = 10,000,000$,

$\left(\frac{s}{\mu}, \frac{\sigma}{\mu}\right)$	m	$\frac{H_{(A)}(Y'_1,\ldots,Y'_m)}{m}$	$H_{(A)}(Y'_m \mid Y'_1,\ldots,Y'_{m-1})$	$\frac{H_{min(A)}(Y'_1,\ldots,Y'_m)}{m}$	$H_{min(A)}(Y'_m \mid Y'_1,\ldots,Y'_{m-1})$
$(25, 0.2)$	4	0.99995	0.99995	0.9925	0.9905
		0.99975	0.99975	0.9758	0.9700
$(10, 0.2)$	4	0.99172	0.99169	0.8893	0.8764
		0.95734	0.95738	0.6803	0.6759
$(100, 0.1)$	4	0.99988	0.99988	0.9877	0.9860
		0.99896	0.99895	0.9465	0.9429
$(10000, 0.01)$	4	0.99998	0.99998	0.9878	0.9857
		0.99892	0.99892	0.9422	0.9422

distribution is defined by μ and σ^2 or by the shape parameter α and the rate parameter β. If the T_j are normally distributed, the distribution of the R_j depends on the triples (s, μ, σ), while for the Gamma distribution it depends on (s, α, β), or equivalently, on $(s, \mu = \alpha/\beta, \sigma = \sqrt{\alpha}/\beta)$. In both cases the distribution of the R_j remains unchanged if the triplet (s, μ, σ) is multiplied by some factor $r > 0$ (transformation theorem). Consequently, the distribution of the variables $R_1, R_2, \ldots$ depends on the two parameters $(s/\mu, \mu/\sigma)$.
Note 1: If the length of the sampling interval s is fixed, the 2-parameter-family of admissible distributions of the R_j can alternatively be parametrized, e.g., by $(\mu, \mu/\sigma)$, (μ, σ^2), or (μ, σ). This is because the parameters can be transformed into each other by bijective mappings on $[0, \infty)^2$
Note 2: These conclusions hold approximately, if the CLT is applicable; see pars. 1012 to 1014.
Note 3: Interestingly, $E(R_j)$ and (asymptotically) $\mathrm{Var}(R_j)$ depend only on $(s/\mu, \sigma/\mu)$ for any distribution of the T_j; cf. pars.1028 to 1029.

1025 [stochastic model] If the random variables $T_1, T_2, \ldots,$ are iid exponentially distributed with parameter τ, it is well-known that the random variables $R_1, R_2, \ldots$ are iid Poisson distributed with parameter τs. In this case the stochastic model depends only on one-parameter.
Note 1: This models a physical noise source that counts radioactive decays with the intervals $I_1, I_2, \ldots,$ using an *ideal* Geiger counter.
Note 2: Also, designs with real-world (non-ideal) Geiger counters are candidates that potentially fit the stochastic model considered in this subsection; cf. Subsect. 5.4.5, see in particular par. 1073.

1026 [online test] Intuitively, it seems to be clear that s/μ and σ/μ should be 'large'. This insight has been confirmed by numerical experiments above. The task of the online test is to detect if any of these conditions are violated. Generally, one should proceed as follows: In a first step subsets A_{real}, A_{good}, and A_{bad} need to be specified, e.g., for the parameters $(s/\mu, \sigma/\mu)$ as described in Subsect. 4.5.3. The online test shall detect when the true parameters leave the subset A_{good} (moving into A_{bad}) when the PTRNG is in operation. Usually, testing the empirical distribution of the T_j is not possible as it would require a precise clock.

1027

[online test] Intuitively, one might expect that a large variance of the random variables $R_1, R_2, \ldots$ ensures large conditional min-entropy of the internal random numbers. This suggests testing the integer-valued raw random numbers $r_1, r_2, \ldots$. At first, we have a closer look at $E(R_j)$ and $\mathrm{Var}(R_j)$.

[renewal process] The random variables $Z'(t) := \inf\{k \mid W_0 + T_1 + \cdots + T_k > t\}$ define a stationary renewal process ($t \in [0, \infty)$); cf. par. 991. It is $Z_n = Z'(ns)$ for all $n \in \mathbb{N}_0$. This means that the random variables $Z_1, Z_2, \ldots$ coincide with $Z'(t)$ at the times $t = s, 2s, \ldots$. Let $s = \tau\mu$, or equivalently, $\tau = s/\mu$. From (4.54) and (4.55) we obtain 1028

$$E(Z_n) = \frac{n\tau\mu}{\mu} = n\tau \tag{5.141}$$

$$\mathrm{Var}(Z_n) = \left(\frac{\sigma^2}{\mu^2}\right) n\tau + \frac{1}{6} + \frac{\sigma^4}{2\mu^4} - \frac{E\left((T_j - \mu)^3\right)}{3\mu^3} + o(1) \tag{5.142}$$

Note 1: Setting $n = 1$ in (5.141) and (5.142) provides $E(R_j)$ and $\mathrm{Var}(R_j)$
Note 2: Furthermore, the stationarity of the R_j implies $E(R_j + \ldots R_{j+i-1}) = E(Z_i)$ and $\mathrm{Var}(R_j + \ldots R_{j+i-1}) = \mathrm{Var}(Z_i)$.

[renewal process] If $T_j \sim N(\mu, \sigma^2)$, for example, then $E\left((T_j - \mu)^3\right) = 0$. Then (5.141) and (5.142) simplify to 1029

$$E(R_j) = \tau \quad \text{and} \tag{5.143}$$

$$\mathrm{Var}(R_j) \approx \left(\frac{\sigma}{\mu}\right)^2 \tau + \frac{1}{6} + \frac{1}{2}\left(\frac{\sigma}{\mu}\right)^4 \quad \text{for 'large' } \tau = \tfrac{s}{\mu} \tag{5.144}$$

In our applications usually $\sigma/\mu < 1$ or even $\sigma/\mu \ll 1$ so that (5.144) further simplifies to

$$\mathrm{Var}(R_j) \approx \left(\frac{\sigma}{\mu}\right)^2 \tau + \frac{1}{6} \quad \text{for 'large' } \tau = \tfrac{s}{\mu} \tag{5.145}$$

[renewal process] Since the random variables $R_1, R_2, \ldots$ are stationary 1030

$$\begin{aligned}
\mathrm{Cov}(R_n, R_{n+1}) &= 0.5\left(\mathrm{Var}(R_n + R_{n+1}) - \mathrm{Var}(R_n) - \mathrm{Var}(R_{n+1})\right) = \\
&= 0.5\left(\mathrm{Var}(R_1 + R_2) - 2\mathrm{Var}(R_1)\right) = \\
&= 0.5\left(-\frac{1}{6} - \frac{1}{2}\left(\frac{\sigma}{\mu}\right)^4 + \frac{E\left((T_j - \mu)^3\right)}{3\mu^3} + o(1)\right) \\
&= -\frac{1}{12} - \frac{1}{4}\left(\frac{\sigma}{\mu}\right)^4 + \frac{E\left((T_j - \mu)^3\right)}{6\mu^3} + o(1)
\end{aligned} \tag{5.146}$$

If the third central moment $E\left((T_j - \mu)^3\right)$ vanishes (e.g., because $T_j \sim N(\mu, \sigma^2)$), if $\sigma/\mu \ll 1$, and if τ (and by this, the sampling interval s) is sufficiently large, then (5.146) simplifies to

$$\mathrm{Cov}(R_n, R_{n+1}) \approx -\frac{1}{12} \tag{5.147}$$

1031 [renewal process] Equation(5.148) follows from the definition and by substituting of (5.142) and (5.146). This is an equivalent to (5.114) for variances and covariances.

$$\begin{aligned}
0 &= \mathrm{Var}\left(R_1 + \ldots + R_j\right) - j\mathrm{Var}(R_1) - 2\sum_{i=2}^{j}(j+1-i)\mathrm{Cov}(R_1, R_i) \\
&= \mathrm{Var}\left(R_1 + \ldots + R_j\right) - j\mathrm{Var}(R_1) - 2(j-1)\mathrm{Cov}(R_1, R_2) - \sum_{i=3}^{j}(j+1-i)\mathrm{Cov}(R_1, R_i) \\
&= \left(\frac{\sigma^2}{\mu^2}\right) j\tau + \frac{1}{6} + \frac{\sigma^4}{2\mu^4} - \frac{E\left((T_j-\mu)^3\right)}{3\mu^3} + o(1) - j\left(\left(\frac{\sigma^2}{\mu^2}\right)\tau + \frac{1}{6} + \frac{\sigma^4}{2\mu^4} - \frac{E\left((T_j-\mu)^3\right)}{3\mu^3} + o(1)\right) \\
&\quad +2(j-1)\left(-\frac{1}{12} - \frac{1}{4}\left(\frac{\sigma}{\mu}\right)^4 + \frac{E\left((T_j-\mu)^3\right)}{6\mu^3} + o(1)\right) - \sum_{i=3}^{j}(j+1-i)\mathrm{Cov}(R_1, R_i) \\
&= o(1) - \sum_{i=3}^{j}(j+1-i)\mathrm{Cov}(R_1, R_i) \quad \text{for } j \geq 3 \qquad (5.148)
\end{aligned}$$

Setting $j = 3$ (5.148) implies that $\mathrm{Cov}(R_1, R_3) = o(1)$. By induction on j one concludes that

$$\mathrm{Cov}(R_1, R_j) = o(1) \quad \text{for } j \geq 3 \qquad (5.149)$$

Thus, if the ratio s/μ is sufficiently large, $\mathrm{Cov}(R_1, R_j) \approx 0$ for $j \geq 3$, which confirms the intuitive feeling.

1032 [renewal process] The first and the third term of (5.142) can be expressed in terms of s/μ and μ/σ. For a given interval length s, both terms depend on the two first moments of the T_j, i.e., on μ and σ^2. The forth term is a multiple of the third central moment of T_j. If the random variables T_j are normally distributed, the fourth term vanishes. Of course, if the interval length s is sufficiently large, the first term dominates anyway.

1033 [online test] In par. 1024 it was pointed out that if the random variables T_j are normally distributed or Gamma distributed, the distribution of the R_j only depends on s/μ and μ/σ. This should be approximately true if the CLT applies. Together with the observation in par. 1032, this suggests applying an online test that exploits estimates of $E(R_j)$ and $\mathrm{Var}(R_j)$. This means that we have to determine regions A'_{real} and A'_{bad} such that $(E(R_j), \mathrm{Var}(R_j)) \in A'_{real}$ implies $(s/\mu, \sigma/\mu) \in A_{real}$, whereas $(s/\mu, \sigma/\mu) \in A_{bad}$ implies $(E(R_j), \mathrm{Var}(R_j)) \in A'_{bad}$. (Ideally, the implications would be equivalences.) And, of course, an appropriate online test must be able to 'separate' A'_{real} from A'_{bad}. In particular, the online test shall quickly detect if $(E(R_j), \mathrm{Var}(R_j)) \in A'_{bad}$.

1034 [online test, $T_j \sim N(\mu, \sigma^2)$] Fig. 20 illustrates the close connection between the min-entropy of the random variables Y'_j and $\mathrm{Var}(R_j)$ for normally distributed T_j. Parameters $(s/\mu, \sigma/\mu)$, for which the conditional min-entropy $H_{min}(Y'_2 \mid Y'_1)$ is 0.98 lie almost perfectly on the curve $\{(s/\mu, \sigma/\mu) \mid \mathrm{Var}(R_j) = 1.05\}$, or shortly, $\{\mathrm{Var}(R_j) = 1.05\}$. This curve was computed by (5.144). For other conditional min-entropy values the situation is similar; the corresponding parameters $(s/\mu, \sigma/\mu)$ lie almost perfectly on curves with constant $\mathrm{Var}(R_j)$.

Note 1: For parameters that imply large conditional min-entropy values (relevant for the class PTG.2), it is $H_{min}(Y_2' \mid Y_1') \approx H_{min}(Y_4' \mid Y_1', Y_2', Y_3')$.
Note 2: For parameters with small min-entropy values, this need not be the case.

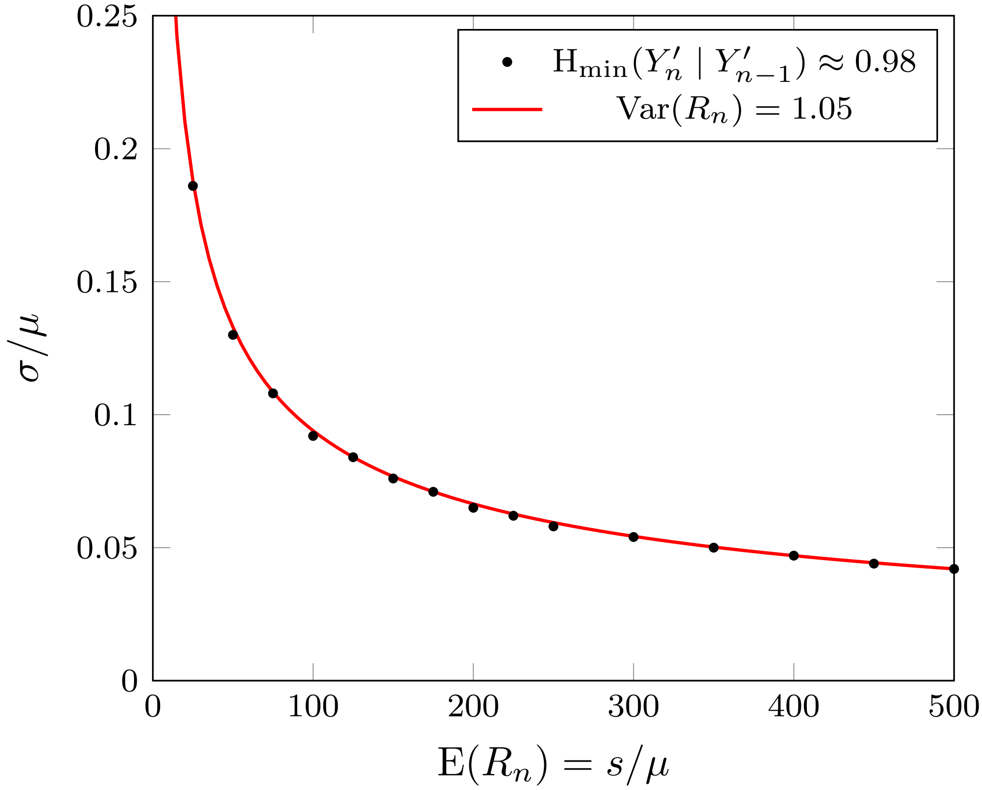

Figure 20: $T_j \sim N(\mu = 1, \sigma^2)$. The black dots belong to parameters with $H_{min}(Y_2' \mid Y_1') = 0.98$

[online test, $T_j \sim N(\mu, \sigma^2)$] Fig. 21 illustrates how the variance of the R_j separates A_{real} and A_{bad}. The upper curve $\{\mathrm{Var}(R_j) = 1.40\}$ corresponds to the conditional min-entropy $H_{min}(Y_2' \mid Y_1') = 0.995$ 1035

[online test, $T_j \sim N(\mu, \sigma^2)$] For normally distributed random variables $T_1, T_2, \ldots$, pars. 1034 and 1035 suggest an online test that only exploits the variance $\mathrm{Var}(R_j)$. Here, analogously to par. 1033 subsets $A''_{real}, A''_{bad} \subsetneq (0, \infty)$ for the variance $\mathrm{Var}(R_j)$ need to be defined. This means that $\mathrm{Var}(R_j) \in A''_{real}$ implies that $(s/\mu, \sigma/\mu) \in A_{real}$, whereas $(s/\mu, \sigma/\mu) \in A_{bad}$ implies that $\mathrm{Var}(R_j) \in A''_{bad}$. In par. 1037 the values $A''_{real} = (1.40, \infty)$ and $A''_{bad} = (0, 1.05)$ are used. 1036

[online test, $T_j \sim N(\mu, \sigma^2)$] For the sample of raw random numbers $r_1, \ldots, r_m$, the online test calculates the one-dimensional empirical variance $\overline{s}^2$ of the random variables R_j with formula (4.23). The online test fails if $\overline{s}^2 < 1.20$. Tab. 7 collects the probabilities for false positives and false negatives, depending on the sample size m of the online test. False positive means that the online test fails although the true distribution is in A''_{real}. If the online test does not fail although the true distribution is in A''_{bad}, we speak of a false negative. 1037

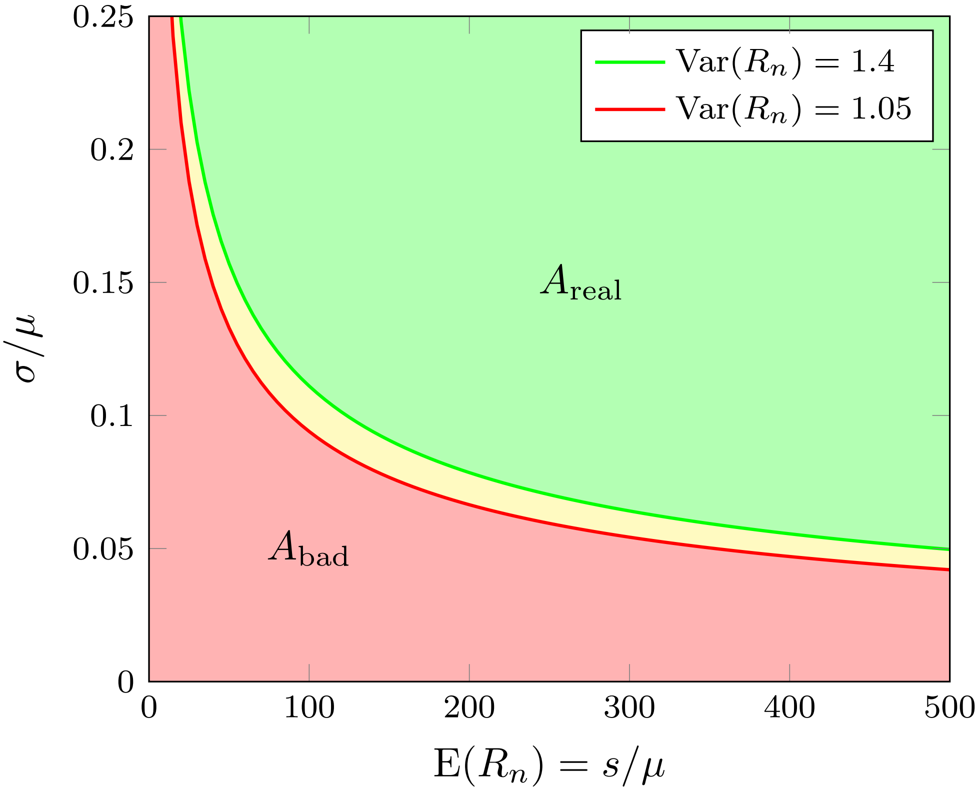

Figure 21: $T_j \sim N(\mu = 1, \sigma^2)$. The variance $\mathrm{Var}(R_n)$ separates A_{real} from A_{bad}.

Note 1: Tab. 7 shows that the online test is very strong, especially for $N = 2048$. In this case, a complex online test procedure as described in Subsect. 5.5.2 may not be necessary.
Note 2: Assume that $B_1, B_2, \ldots, B_m$ are iid $B(1,p)$-distributed with $H_{min} \geq 0.995$ (as for $\{\mathrm{Var}(R_j) \geq 1.40\}$)Then $|p - 0.5| \leq 0.0017$. There, it requires a much larger sample size m to separate equally efficient between $A^*_{real} = [0.4983, 0.5017]$ and $A^*_{bad} = [0, 0.4931] \cup [0.5069, 1]$.

1038 [online test, T_j is Gamma distributed] Finally, we consider Gamma-distributed random variables T_j (shape parameter α, rate parameter β). The variance $\mathrm{Var}(R_j)$ follows from (5.142) with $n = 1$, where we again assume that the $o(1)$-term is negligible for the selected parameters. For Gamma-distributed T_j, the third central moment does not vanish. Interestingly,

$$\frac{E\left((T_j - \mu)^3\right)}{3\mu^3} = \frac{\sigma^3 E\left(\left(\frac{T_j - \mu}{\sigma}\right)^3\right)}{3\mu^3} = \frac{1}{3}\left(\frac{\sigma}{\mu}\right)^3 \cdot \frac{2}{\sqrt{\alpha}} = \frac{2}{3}\left(\frac{\sigma}{\mu}\right)^4 \tag{5.150}$$

Fig. 22 illustrates the close connection between the min-entropy of the random variables Y'_j and $\mathrm{Var}(R_j)$ if the T_j are Gamma distributed. The situation is rather similar to normally distributed random variables T_j; cf. par. 1034, Fig. 20. Consequently, also for Gamma distributed T_j, the empirical variance is an appropriate online test.
Note: Care should be taken that the ratio s/μ is not too small. Otherwise, (even if σ/μ is sufficiently large) for $s/\mu \leq 2$, the results are sensitive to changes of the parameters.

Table 7: Simulated probabilities for false positives and false negatives: $T_j \sim N(\mu, \sigma^2)$, $\mu = 1.0$, test sample size m, number of simulated test values $N = 10,000,000$.

m	Prob(**false positive**)	Prob(**false negative**)
1024	0.00034	0.00096
1536	0.00002	0.00007
2048	0.0000007	0.0000045

[online test] If the random variables are neither normally distributed nor Gamma distributed, the relation between the conditional min-entropy of the random variables Y'_j and $\mathrm{Var}(R_j)$ needs to be analyzed as it was done above. Care should be taken that the empirical mean of the raw random numbers $r_1, r_2, \ldots$ does not become too small, neither by design nor accidentally while the PTRNG is in operation. Unless the variance of the T_j increases correspondingly, the second scenario should be detected by an online test that computes the empirical variance of R_j. Otherwise, if it might be possible (under consideration of the physical noise source) that both the empirical variance and the empirical mean of the T_j can significantly increase at the same time, the online test should additionally monitor the empirical mean of the raw random numbers. 1039

5.4.4 Sampling events with iid intermediate time intervals – Design B

As in Subsection 5.4.3 this subsection focuses on the mathematical treatment of a generic stochastic model, which may fit different noise sources designs. 1040
Note: The developer may refer to this subsection but, of course, has to give evidence that the stochastic model indeed fits the design under evaluation.

In this subsection we assume that a physical noise source latches a (perfect) square wave with constant period length s whenever a (design-specific) random event occurs. This event might be, for example, that a ring oscillator has terminated N periods since the last latching. The time intervals between two successive events are denoted by $t_1, t_2, \ldots$. Thus, the square wave is latched at time instants $t_1, t_1 + t_2, \ldots$, and the raw random numbers $r'_1, r'_2, \ldots$ are given by 1041

$$r'_j = \begin{cases} 0 & \text{if } w_0 + t_1 + \ldots + t_j \in [ks, (k+0.5)s) \text{ for some } k \in \mathbb{N}_0 \\ 1 & \text{if } w_0 + t_1 + \ldots + t_j \in [(k+0.5)s, (k+1)s) \text{ for some } k \in \mathbb{N}_0 \end{cases} \tag{5.151}$$

Note: Compared to Subsection 5.4.3 the roles of the noise source and of the constant signal are interchanged.

The goal is to determine the joint distribution of random variables $R'_1, \ldots, R'_m$. This allows a determination of the joint entropy and conditional entropies for both Shannon entropy and min entropy. Unlike in Subsection 5.4.3 the raw random numbers are not integer-valued but already binary-valued. 1042

As in Subsection 5.4.3 we assume that sampling is ideal in the sense that the raw random numbers $r'_1, r'_2, \ldots$ are given by (5.151). The latching event can occur around times ks or $(k+0.5)s$ (for 1043

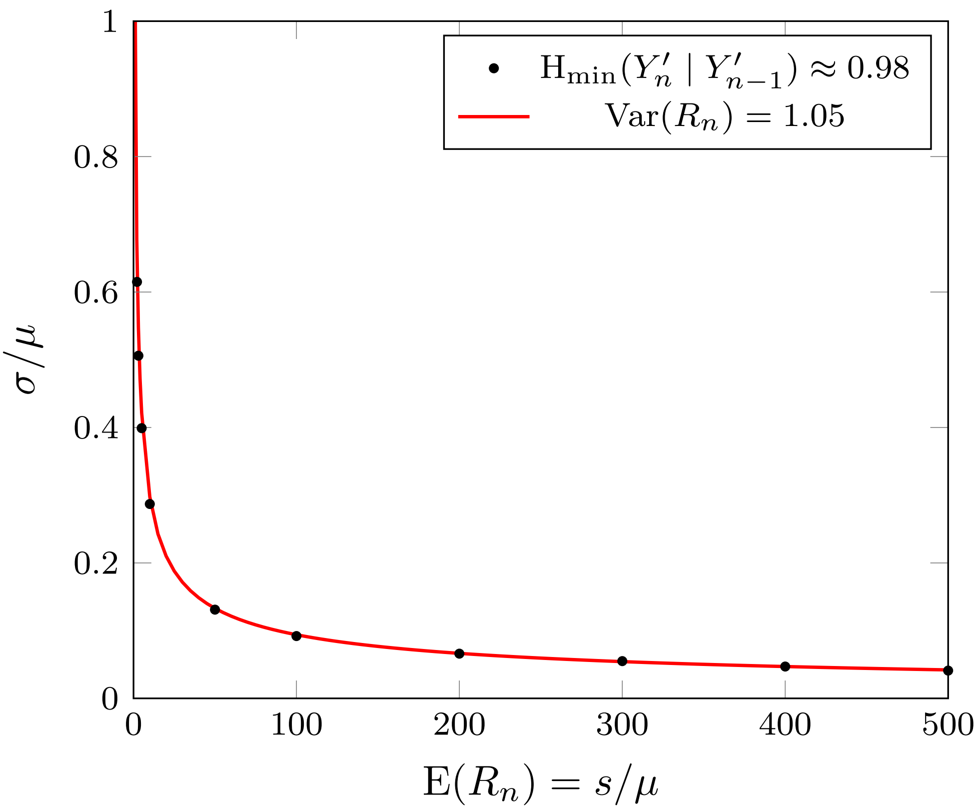

Figure 22: T_j is Gamma-distributed. The black dots belong to parameters with $H_{min}(Y'_2 \mid Y'_1) = 0.98$

$k \in \mathbb{N}_0$) when the square wave changes its value (from 1 to 0 or from 0 to 1), which may cause deviations from (5.151). If such 'latching errors' occur rarely, they may be neglected in the stochastic model.

1044 [iid assumption] We assume that the time lengths $t_1, t_2, \ldots$ can be viewed as realizations of iid random variables $T_1, T_2, \ldots$. In particular, the random variables $X(t) := \sup\{k \mid W_0 + T_1 + \cdots + T_k \leq t\}$ define a renewal process ($t \in [0, \infty)$); cf. par. 972.

1045 As in Subsection 5.4.3 the random variable W_0 describes the (random) time when the first event occurs after time $t = 0$. The random variable W_0 quantifies the phase of the random events relative to the square wave when the first considered time interval begins.

1046 [Assumption] The distribution of the random variables $T_1, T_2, \ldots$ has density (to be mathematically precise: Lebesgue density) $g(\cdot)$. If $G_T(\cdot)$ denotes the cumulative distribution function of T_j, then $W_0 j$ has the density $g_W(\cdot) = \frac{1}{\mu}(1 - G_T(\cdot))$ (par. 968).
Note: In particular, $\mathrm{Prob}(W_j > 0) = 1$ for all $j \in \mathbb{N}$.
Note: The case that the distribution of T_j has a density constitutes the most relevant case for applications. We mention that similar results can be derived if the random variables $T_1, T_2, \ldots$

do not have a density $g(\cdot)$ (although with greater mathematical effort). If the random variables T_j are discrete, the integrals below turn into sums.

For each integer $\ell \geq 1$ the term $g^{*(\ell)}(\cdot)$ denotes the ℓ-fold convolution of the density $g(\cdot)$. In particular, $g^{*(1)}(\cdot) = g(\cdot)$. 1047
Note: For each $t \in \mathbb{N}_0$ and $\ell \in \mathbb{N}$, the sum $T_{t+1} + \cdots + T_{t+\ell}$ has density $g^{*(\ell)}(\cdot)$.

We define the sets $A(0)$ and $A(1)$: 1048

$$A(0) := \bigcup_{k=0}^{\infty} [ks, (k+0.5)s) \quad \text{and} \quad A(1) := \bigcup_{k=0}^{\infty} [(k+0.5)s, (k+1)s) \tag{5.152}$$

It is $R'_j = b$ iff $W_0 + T_1 + \cdots + T_j \in A(b)$ for $b = 0, 1$.

For $m \in \mathbb{N}$ and $b_1, \ldots, b_m \in \{0, 1\}$ we have 1049

$$\begin{aligned} &\text{Prob}\,(R'_1 = b_1, R'_2 = b_2, \ldots, R'_m = b_m) = \\ &\text{Prob}\,(W_0 + T_1 \in A(b_1), W_0 + T_1 + T_2 \in A(b_2), \ldots, W_0 + T_1 + \cdots T_m \in A(b_m)) = \\ &\int\limits_{A(b_m)} \int\limits_{A(b_{m-1})} \cdots \int\limits_{A(b_1)} \int\limits_{0}^{\infty} g^{*(m)}(u_m - u_{m-1}) g^{*(m-1)}(u_{m-1} - u_{m-2}) \cdots \\ &\qquad\qquad g^{*(1)}(u_1 - u_0) g_W(u_0)\, du_0 du_1 \cdots du_{m-1} du_m \end{aligned} \tag{5.153}$$

Note: The term du_0 belongs to '$\int_0^\infty$'.
Note: Since $g_W(u), g^{*(\ell)}(u) = 0$ for $u < 0$ ($\ell \geq 1$), the integrand does not contribute to the integral unless $0 \leq u_0 \leq u_1 \leq \cdots \leq u_m$.

If the noise source and the square wave signal are synchronized at time $t = 0$, then $W_0 \equiv 0$, and the inner integral ($\int_0^\infty$) in (5.153) drops out. 1050

[Numerical example] (to be continued) Table 8 provides several numerical examples. In all cases the random variables T_j are assumed to be $N(\mu, \sigma^2)$-distributed. The figures were not gained by the evaluation (5.153). Instead, the random variables T_j, and implicitly the random variables R'_j, were simulated (sample size N). 1051

The conditional Shannon entropy is computed with formula (4.73) (with $m - 1$ in place of m). The conditional min-entropy (last column of Table 8) applies the formula

$$\begin{aligned} &H_{min}(R'_m \mid R'_1, \ldots, R'_{m-1}) = \\ &\qquad \min\{H_{min}(R'_m \mid R'_1 = b_1, \ldots, R'_{m-1} = b_{m-1}) \mid b_1, \ldots, b_{m-1} \in \{0, 1\}\}\,. \end{aligned} \tag{5.154}$$

It is much easier to express the joint probability $\text{Prob}\,(R'_1 = b_1, R'_2 = b_2, \ldots, R'_m = b_m)$ by integrals than in Subsection 5.4.3, and concrete calculations require much less effort. On the negative side, it seems to be very difficult to develop an effective online test unless the device is able to measure the intermediate times $t_1, t_2, \ldots$ (or, equivalently, the times $t_1, t_1 + t_2, \ldots$). The reason is that the binary-valued raw random numbers $r'_1, r'_2, \ldots$ contain much less information than the integer-valued raw random numbers $r_1, r_2, \ldots$ in Subsection 5.4.3. 1052

Table 8: Simulation experiments (design type B): $T_j \sim N(\mu, \sigma^2)$, $\mu = 1.0$, sample size $N = 10,000,000$

$\left(\frac{s}{\mu}, \frac{\sigma}{\mu}\right)$	m	$\frac{H(R'_1,\ldots,R'_m)}{m}$	$H(R'_m \mid R'_1,\ldots,R'_{m-1})$	$\frac{H_{min}(R'_1,\ldots,R'_m)}{m}$	$H_{min}(R'_m \mid R'_1,\ldots,R'_{m-1})$
$(1971, 0.24)$	4	0.9945	0.9913	0.9211	0.7776
$(3942, 0.34)$	4	0.9975	0.9967	0.9306	0.9012
$(7885, 0.48)$	4	0.9994	0.9999	0.9969	0.9948

5.4.5 PTRNG exploiting radioactive decay

1053 In Subsect. 5.4.5 we discuss the stochastic model of a PTRNG design which exploits a physical phenomenon, namely radioactive decay, detected and digitized by a Geiger counter. Mathematically, we follow [Neue04], Sect. 4.2. We mention that this design is also treated in ISO / IEC 20543 [ISO_20543], A.3.4, Example 3.

1054 [design] The noise source is a radioactive source that is assumed to decay spontaneously. There is a Geiger counter close to this radioactive source. The Geiger counter is connected to a computer. The device measures the intermediate times $t_1, t_2, \ldots$ between consecutive impulses of the Geiger counter. From these intermediate times the PTRNG computes the raw random numbers. Furthermore, the half-life L of the radioactive material is substantially larger than the expected lifetime of the PTRNG.

1055 Note: In this subsection we assume that the radioactive material decays to stable products. Our considerations apply to decay chains, too, if the number of decays of the generated radioactive products is negligible compared to the number of radioactive decays of the radioactive starting material. This is the case if the half-life of the generated radioactive products is very large (absolute and compared to that of the radioactive starting material). Of course, the considerations below can be adjusted to arbitrary decay chains, although in the general case the mathematical treatment becomes more difficult.

1056 The measured intermediate times $t_1, t_2, \ldots$ are interpreted as realizations of random variables $T_1, T_2, \ldots$.

1057 [ideal Geiger counter] To become familiar with this design and its special features at first, pars. 1058 to 1063 consider an ideal Geiger counter. This means that it detects all radioactive decays and measures the intermediate times exactly. In particular, the dead time of an ideal Geiger counter is 0.

1058 [ideal Geiger counter] If the Geiger counter is ideal, the random intermediate times are independent and exponentially distributed with parameter θ. The density $f(\cdot)$ and cumulative distribution function $F(\cdot)$ of the random variables T_j are given by

$$f(t) = \frac{1}{\theta} e^{-\theta t} \quad \text{and} \quad F(t) = 1 - e^{-\theta t} \quad \text{for } t > 0 \tag{5.155}$$

for some parameter $\theta > 0$. The parameter θ does not only depend on the radioactive material but also on its quantity. For a realistic lifetime of the RNG, we may assume that the parameter θ essentially remains constant; cf. par. 1055.

[ideal Geiger counter] Under this assumption, a random variable N that counts the number of impulses within a time interval of length s is Poisson distributed with parameter $\lambda = \theta s$. In particular, 1059

$$\text{Prob}\,(N = k) = \frac{\lambda^k e^{-\lambda}}{k!} \quad \text{for } k \in \mathbb{N}_0. \tag{5.156}$$

[ideal Geiger counter] If T_j is exponentially distributed with parameter θ, then the random variable $U_j = e^{-\theta T_j}$ is uniformly distributed on the unit interval. (Note that $\text{Prob}(U_j \leq x) = x$ for each $x \in (0, 1)$.) Thus, the k most significant bits of $u_1, u_2, \ldots$ (derived from the times $t_1, t_2, \ldots$) may be used as k-bit raw random numbers. 1060

[ideal Geiger counter] The drawback of both methods described in par. 1060 is that one needs to know the (exact) parameter θ, resp. $\lambda = \theta s$. Additionally, for real-world (non-ideal) Geiger counters, the detection rate q affects θ and λ. These properties are in particularly unfavorable if many PTRNGs have to be evaluated and if it is not a realistic option to estimate θ (and the detection rate q) individually for each PTRNG. Instead, par. 1062 proposes an algorithm that gets by without knowledge of θ. Additionally, over time the parameter θ shrinks to some degree. The cost is that the output rate decreases to 50%. 1061

[ideal Geiger counter] Under the assumptions from par. 1058, the random variables 1062

$$U_i = \frac{T_{2i}}{T_{2i-1} + T_{2i}} \quad \text{for } i \in \mathbb{N}. \tag{5.157}$$

are uniformly distributed on the unit interval $[0, 1)$, regardless of θ. The k most significant bits R_i of the binary representation of U_i are uniformly distributed on $\{0, 1\}^k$. Thus, from $t_1, t_2, \ldots$, one can compute the values $u_1, u_2, \ldots$ and therefrom, the k-bit raw random number vectors $r_1, r_2, \ldots$.

[ideal Geiger counter] Of course, a stochastic model as developed in pars. 1057 to 1062 will not be accepted by the evaluator because ideal Geiger counters do not exist in the real world. Below, we consider a more realistic scenario. 1063

A 'real-world' detection mechanism is not able to measure the intermediate times between decays exactly but only in multiples of a positive constant Δ (i.e., length of a clock cycle). If multiple impulses occur within one clock cycle, they are only counted once. 1064

The random intermediate times $T_1, T_2, \ldots$ are discrete and can assume values that are integer multiples of the dead time Δ. In particular, the random variables $T_1, T_2, \ldots$ are iid geometrically distributed with parameter $p = 1 - e^{-\theta\Delta}$ ($= \text{Prob}(T_j \leq \Delta)$). 1065

These considerations are also appropriate for a Geiger counter that detects a decay with probability q. If we assume $\Delta = 0$ for the moment, the random variables N of the detected decays per time unit would be Poisson distributed with parameter $\lambda = q\theta s$ instead of $\lambda = \theta s$. Thus, the 1066 random variables $T_1, T_2, \ldots$ (modeling a Geiger counter with detection rate q and dead time Δ) are iid geometrically distributed with parameter $p = 1 - e^{-q\theta\Delta}$.

1067 Formula (5.157) requires a modification that considers these real-world assumptions. Analogously to (5.157), we set

$$U'_i = \frac{T_{2i} - 0.5\Delta}{T_{2i-1} + T_{2i} - \Delta} \quad \text{for } i \in \mathbb{N}. \tag{5.158}$$

This leads to the following inequalities ([Neue04], Theorem 4.1)

$$\frac{1}{2}\tanh\left(\frac{p}{2}\right) \leq \|F_U - F_{U'}\|_\infty \leq 1 - e^{-\frac{p}{2}} \leq \frac{p}{2}. \tag{5.159}$$

Recall that $p = 1 - e^{-q\theta\Delta}$. The right-hand inequality is a well-known property of the exponential function. Here, $F_U(\cdot)$ and $F_{U'}(\cdot)$ denote the cumulative distribution functions of the uniform distribution U on $[0,1)$ and of U'. Furthermore, $\|\cdot\|_\infty$ denotes the supremum norm in $\mathbb{R}$. This means that

$$\|F_U - F_{U'}\|_\infty = \sup_{x\in[0,1]} \{|F_U(x) - F_{U'}(x)|\}. \tag{5.160}$$

For our purposes the upper bound in (5.159) is relevant. To avoid confusion we point out that [Neue04], Sect. 4.2, applies an alternate definition of the geometric distribution, cf. par. 442.

1068 Formula (5.158) can be used to generate values in $u'_1, u'_2, \ldots \in [0,1)$ from the intermediate times $t_1, t_2, \ldots$ between consecutive impulses of the Geiger counter. From the sequence $u'_1, u'_2, \ldots$, the final k-bit raw random numbers $r_1, r_2, \ldots$ can be computed. The random variables $R_1, R_2, \ldots$ are no longer uniformly distributed on $\{0,1\}^k$ as would be the case for an ideal Geiger counter.

1069 If the random variables T_j are geometrically distributed with parameter $p_\theta = 1 - e^{-q\theta\Delta}$, raw random numbers R_j have distribution π_p. For the moment, let $\vec{b} = (b_1, \ldots, b_k) \in \{0,1\}^k$ and $s(\vec{b})$ be the binary representation $(0.b_1 \ldots b_k)_2 = \sum_{j=1}^k b_j 2^{-j}$. Formula (5.159) quantifies the deviation of the random variables R_j from the uniform distribution on $\{0,1\}^k$

$$\begin{aligned}\pi_p(\vec{b}) \quad &= \quad \text{Prob}\left(U' \in [s(\vec{b}), s(\vec{b}) + 2^{-k})\right) = \sum_{\ell=0}\sum_{m=0} \text{Prob}\left(T_{2i} = \ell, T_{2i-1} = m, U'_i \in [s(\vec{b}), s(\vec{b}) + 2^{-k})\right) = \\ &\sum_{\ell=0}\sum_{m=0} (1-p)^2 p^\ell p^m 1_{\{s(\vec{b}) \leq \frac{\ell+0.5}{m+\ell+1} < s(\vec{b}) + 2^{-k}\}}.\end{aligned} \tag{5.161}$$

From (5.159) and (5.161) we conclude that

$$\left|\pi_p(\vec{b}) - 2^{-k}\right| \leq p \quad \text{for all } \vec{b} \in \{0,1\}^k \tag{5.162}$$

which provides an estimate for the min-entropy. Of course, more accurate evaluations of the right-hand term in (5.161) may yield larger entropy bounds.

1070 This PTRNG is based on well-understood physical laws that, in particular, describe the number of radioactive decay events per time unit. The chain of reasoning that connects random events to the entropy of the generated random numbers contains model assumptions.

1071 Within the evaluation process the applicant has to give evidence that the design under evaluation indeed fulfills these model assumptions that were specified in pars. 1064 and 1065. Depending on the concrete design, it may turn out that refinements or corrections of the stochastic model will be necessary.

Note: To be PTG.2-compliant an appropriate online test and an appropriate total failure test need to be implemented. 1072

Note: Alternatively, raw random numbers can be derived from the number of impulses of the Geiger counter within time intervals $I_1, I_2, \ldots$, with $I_j = [(j-1)s, js)$; cf. Subsect. 5.4.3. 1073

5.4.6 A PLL-based physical noise source

In Subsect. 5.4.6 we briefly discuss a PLL-based physical noise source. The design is described, and central features are explained. For details we refer the interested reader to [FiDr02; BeFV10; FiBB19]. 1074

[PLL] Fig. 23 shows (a particular type of) a PLL. As usual, the acronym 'PLL' stands for 'phase-locked loop'. 1075

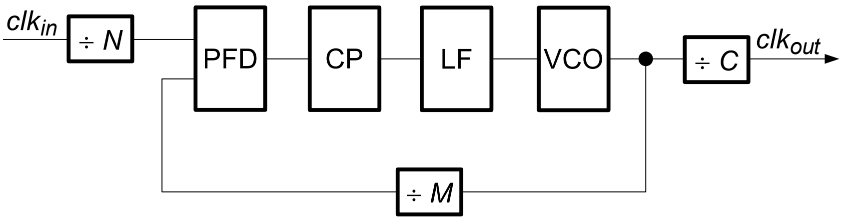

Figure 23: PLL (Phase-locked loop); source: [FiBB19], Fig. 2

[PLL] The PLL in Fig. 23 divides the frequency f_{in} of the input signal clk_{in} by a factor N and the output frequency of the VCO (voltage controlled oscillator) by factor a C. The PFD (phase-frequency detector) compares the phase and the frequency of the input clock signal clk_{in} with the output signal clk_{out} of the PLL. More precisely, the PFD ensures that the output frequency $f_{out(VCO)}$ of the VCO, divided by M, equals f_{in}, divided by N. Furthermore, in Fig. 23 the acronyms 'CP' and 'LF' stand for the charge pump and loop filter. In the following we may assume that the phase difference between clk_{in} and clk_{out} remains constant. The numbers N, M, C are integers. Altogether, this gives the relation 1076

$$f_{out} = f_{in} \frac{M}{NC} = f_{in} \frac{K_M}{K_D} \, . \tag{5.163}$$

The terms $K_M, K_D \in \mathbb{N}$ denote the frequency multiplication and division factors of the PLL. We assume that $\gcd(K_M, K_D) = 1$ in the following, i.e., that K_M and K_D are relatively prime.

[PLL] The jitter of the input signal clk_{in} is intended to be kept as small as possible. This aim can be reached, for example, by using a low-jitter quartz oscillator. On the other hand, the PLL 1077 parameters (the bandwidth of the loop filter, for example) should be selected in a way such that the impact of the thermal noise on the output clock jitter is maximal. The VCO provides the main contribution to the jitter.

1078 [design] The PLL-based physical noise source discussed in this subsection uses a PLL as a source of randomness and a coherent sampling mechanism to convert the jitter of clk_1 into raw random numbers; see Fig. 24. In coherent sampling, both the sampled signal and the sampling signals are periodic signals with a known (fixed) frequency ratio. In the following we assume that both signals are binary clock signals. The use of a PLL enables coherent sampling.

1079 [design] The lower part of Fig. 24 illustrates a design of a physical noise source that is based on one PLL. We note that PLL-based physical noise sources can use more than one PLL; cf. [FiBB19], Fig. 5, for example. In the following we restrict our attention to the most elementary design with one PLL and refer the interested reader to the literature. The reference clock signal

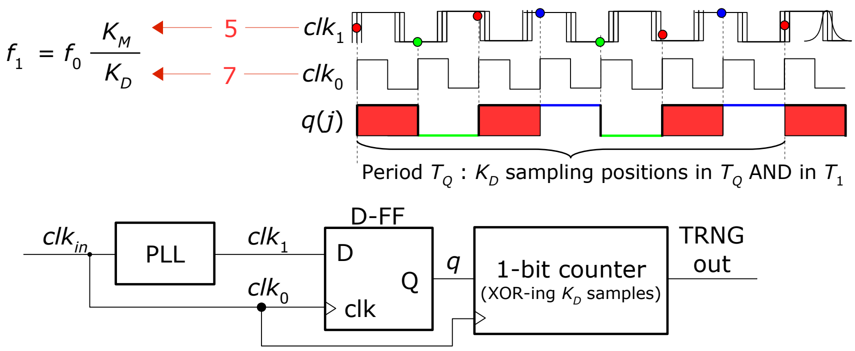

Figure 24: Sampling mechanism of a PLL-based physical noise source (schematic design); source: [FiBB19], Fig. 3

clk_0 latches the jittered clock signal clk_1 of the PLL with a flip flop (D-FF) on the rising edge. The 1-bit counter outputs the parity of the number of sampled signals that equal 1 (XOR sum) within K_D cycles of the sampling signal clk_0. This XOR sum provides a single binary-valued raw random number r_k.
Note: We follow the notation in [FiBB19]. If the physical noise source uses only one PLL (as in Fig. 24), clk_0 (reference clock) and clk_1 (sampled clock signal) coincide with clk_{in} and clk_{out} (input and output clock of a PLL). In [FiBB19] the authors also consider designs that exploit more than one PLL, which motivates this notation.

1080 [FPGA] PLLs are usually available on FPGAs, and PLLs are physically separated from the rest of the FPGA. This is a good feature for physical noise sources. Furthermore, PLLs are robust to environmental conditions.

1081 The PLL fixes the frequency relationship $f_1/f_0 = K_M/K_D$ and also ensures that the signals clk_{in} and clk_{out} are in phase. If the sampled clock signal clk_1 would be a regular signal with a fixed cycle length, i.e., if clk_1 (and clk_0) was jitter-free, the output sequence of the flip-flop would be K_D-periodic (in absolute time: $K_D\times$ the cycle length of clk_0). The sampled signal

clk_1 is jittered so that the output sequence is only 'almost' K_D-periodic, and the deviations induce the entropy. Also here, since the phase difference between clk_0 and clk_1 is (time-locally) constant, many output values of the flip-flop are always 0 or always 1 because the time period to the previous or to the next switch of clk_1 from 0 to 1 or from 1 to 0 is large compared to the jitter of clk_1. Only the remaining output values of the flip-flop contribute to the entropy of the raw random numbers. The upper part of Fig. 24 illustrates this phenomenon by a toy example. The green circles and blue circles mark sampled values that are always 0 or always 1, respectively. Only the 'random' red circles, which are close to the jittered clock edges, contribute to the entropy of the raw random numbers.

Actually, both clk_0 and clk_1 are jittered. Since the PLLs (clock generators) are physically isolated (if more than one PLL is being used), it is supposed that their jitters are independent. Consequently, the jitter of the reference clock signal clk_0 can be transferred to the sampled signal clk_1. Finally, the relative jitter between clk_0 and clk_1 is relevant. Thus, it is reasonable to consider a stochastic model where the reference clock signal is jitter-free. 1082

Within one 'conversion period' (time interval needed to generate one raw random bit), the signal clk_1 is sampled K_D times on the rising edges of signal clk_0. In particular, $\mu_1 = (K_D/K_M)\mu_0$, and a conversion period takes time $K_D\mu_0 = K_M\mu_1$. Here, μ_1 denotes the average cycle length of signal clk_1 and μ_0 the cycle length of clk_0. 1083
Note: To simplify reading we refrain from an additional index k that labels the conversion period.

The distribution of the output sequence of the flip-flop is far from being stationary. On the other hand, due to the control mechanism of the PLL, we may assume that the raw random numbers $r_1, r_2, \ldots$ can be interpreted as realizations of stationarily distributed binary-valued random variables $R_1, R_2, \ldots$. However, due to the feedback mechanism within the PLL, dependencies between the random variables $R_1, R_2, \ldots$ could arise. In particular, within the evaluation, the autocorrelation of the raw random numbers should be investigated thoroughly to detect (or exclude) possible (long-term) dependencies. After that, it remains to formulate, to verify, and to analyze a stochastic model to obtain a reliable lower entropy bound for the raw random numbers. 1084
Note: Maybe the K_D output bits of the flip-flop D-FF within each conversion period (viewed as a binary vector) can be assumed to be (time-locally) stationarily distributed (to be investigated).

During each conversion period the signal clk_1 is latched K_D times. The sampled value x_i depends on the relative phase z_i within a cycle of clk_1 at the time of sampling. If $z_i \in [0, \mu_1/2)$ (first half-period), the sampled value x_i is 1, and if $z_i \in [\mu_1/2, \mu_1)$ (second half-period), it equals 0. This observation suggests ordering the sampled values x_i with regard to the average relative phases at the sampling times. The average (expected) relative phase of the i^{th} sampled value equals $(\phi_0 + i \cdot \mu_0)(\bmod \mu_1)$. Here, ϕ_0 denotes the relative phase between clk_1 and clk_0 at the beginning of the conversion period. Regardless of the phase difference ϕ_0, the K_D sampled average values are uniformly distributed on $[0, t_1)$, where the distance between neighboured values equals $\Delta := t_1/K_D$. 1085

Actually, the signal clk_1 is jittered, and thus the intervals between the neighboured relative phases are not perfectly identical but jittered. In [BeFV10; FiBB19] the authors interpret the relative phase z_j of the j^{th} sampled point x_j as a realization of a random variable $Z_j \sim N(\mu_{(j)}, \sigma^2)$ with $\mu_{(j)} := \phi_0 + j \cdot \mu_0 (\bmod \mu_1)$. 1086
Note: Due to the jitter of clk_1, the random variable Z_j can assume values outside $[0, t_1)$.

1087 From the sampled values $x_0, \dots, x_{K_D-1}$ a new raw random number is computed via $r_k \equiv x_0 + \cdots + x_{K_D-1} (\mathrm{mod}\, 2)$. Thus, the distribution of the XOR sum $R_k = X_0 + \cdots + X_{K_D-1} (\mathrm{mod}\, 2)$ has to be studied.

1088 Since $\sigma \ll \mu_1$ it suffices to consider the nearest switch of a half-period of clk_1. In [FiBB19], Eq. (5), the searched probability is calculated as

$$\mathrm{Prob}(X_j = 1) = 1 - \frac{1}{\sqrt{2\pi}\sigma} \left(\int_{-\infty}^{\mu_1} e^{-\frac{(z-\mu_j)^2}{2\sigma^2}}\, dz - \int_{0}^{\mu_1/2} e^{-\frac{(z-\mu_j)^2}{2\sigma^2}}\, dz \right) \tag{5.164}$$

Note that $\mathrm{Prob}(Z_j < -t_1/2)$ and $\mathrm{Prob}(Z_j > 3t_1/2)$ are essentially 0.

1089 In [FiBB19] it is assumed that the random variables $X_0, \dots, X_{K_D-1}$ are independent. The independence assumption yields

$$\begin{aligned} \mathrm{Prob}(R = 1) &= 0.5 + B \quad \text{for } R := X_0 + \cdots + X_{K_D-1} \\ B &= \left(\frac{1}{2}\right)^{K_D-1} \prod_{j=0}^{K_D-1} (\mathrm{Prob}(X_j = 1) - 0.5)\,. \end{aligned} \tag{5.165}$$

1090 As already pointed out in par. 1081, some random variables X_j assume the values 0 or 1 with probability 1 because $\sigma \ll \mu_1$. These sampled values do not contribute to the entropy of the raw random numbers, and thus, these random variables are not considered in the following. The focus is on the random variables whose indices are in a subset of the indices $\mathcal{M} = \{j_1, \dots, j_s\} \subset \{0, \dots, K_D - 1\}$, for which $\mathrm{Prob}(X_j = 1)$ differs significantly from 0 and 1. These random variables contribute to the entropy of the raw random numbers due to the jitter of clk_1. Essentially, the distribution of

$$X_{j_1} + \cdots + X_{j_s} (\mathrm{mod}\, 2) \tag{5.166}$$

needs to be studied.
Note 1: The XOR sum of the remaining (i.e., non-considered) sampled bits may be 0 or 1, thereby affecting the value of the raw random number. However, these values have no impact on the entropy of the raw random numbers. In particular, if the clk_1 (and clk_0) signal would be jitter-free, the generated raw random numbers would be constant 0 or 1, depending on the number of sampled values equal to one.
Note 2: As the value of Δ decreases, more sample values will contribute to the entropy of the raw random numbers and the entropy will be increased.

1091 In [FiBB19] the total failure test and the online test are not applied on the raw random numbers but exploit all the sampled values within the conversion periods. More precisely, two parameters P_1 and P_2 are estimated; cf. [FiBB19] (9), (10). The first estimate is used by the total failure test, the second by the online test. We refer the interested reader, e.g., to [FiBB19].

5.5 Online tests

1092

In Section 5.5 different online test schemes are discussed, and their advantages and disadvantages are explained. The general considerations from Subsection 4.5.3 are supported by examples.

In this section we assume that binary-valued raw random numbers $r_1, r_2, \ldots$ are tested. Furthermore, $c = \xi(r_1, \ldots, r_m)$ denotes the test value, ξ is the applied statistical test, and m is the sample size of the statistical test ξ in bits. 1093

Consequently, we interpret the test value as a realization of the random variable $C = \xi(R_1, \ldots, R_m)$. It is a relevant part of the evaluation to understand the distribution of C under the admissible parameters of the stochastic model. 1094

As explained in Subsection 4.5.3 (par. 707), the online test shall be selected with regard to the stochastic model of the noise source. In Section 5.5 we tacitly assume that the online test, or more precisely, the applied statistical test(s), is appropriate for the stochastic model. Instead, we focus on the suitability of the whole online test procedure, including the evaluation rules. 1095

5.5.1 A look at single statistical tests

In this subsection we focus on single statistical (online) tests. A negative example is provided and desirable properties are discussed. The results motivate the design of more sophisticated test suites. 1096

[χ^2 goodness-of-fit test] If a χ^2 goodness-of-fit test on 4-bit words (a.k.a. poker test, often briefly denoted as χ^2-test) is applied to the raw random numbers, the sequence $r_1, r_2, \ldots, r_m$ is divided into non-overlapping 4-tuples $w_1, \ldots, w_{m/4}$ where $w_j = (r_{4j-3}, r_{4j-2}, r_{4j-1}, r_{4j})$. For $i = 0, \ldots, 15$ the term $fr(i) := |\{j \le n \mid w_j = i\}|$ equals the frequency of the 4-tuple i. Here, we identify the 4-bit vector w_i with the binary representation of an integer. The test value is given by 1097

$$c := \sum_{i=0}^{15} \frac{\left(fr(i) - \frac{m}{16}\right)^2}{\frac{m}{16}} \quad . \tag{5.167}$$

Note: The χ^2-test in (5.167) corresponds to a scenario where the null hypothesis says that the tested raw random numbers were generated by an ideal RNG.

Negative Example ([Schi01], Example 2): The online test applies the χ^2 goodness-of-fit test from par. 1097 with sample size $m = 320$. The online test fails if the test value c exceeds 65.0. It is claimed that $\text{Prob}(C > 65.0) = 3.4 \cdot 10^{-8}$ for ideal RNGs, i.e., for iid $B(1, 0.5)$-distributed random variables R_j. 1098

The example in par. 1098 shows several problems that may occur with online tests. We follow and extend the analysis and the conclusions in [Schi01]. 1099

The distribution of the test variable C of the poker test (5.167) converges to the χ^2-distribution with 15 degrees of freedom as the sample size $m \to \infty$. This indeed suggests the significance level $3.4 \cdot 10^{-8}$ from par. 1098. However, $C > 65.0$ is a very rare event, at least if the tested raw 1100 random numbers are 'almost' ideal. Generally speaking, at the tails of the distribution the rate of convergence may be low.

1101 [relative approximation error] Generally, when computing rejection probabilities from limit distributions, the relative approximation error

$$\frac{|p_{exact} - p_{approx}|}{|p_{approx}|} \qquad \text{(relative error)} \tag{5.168}$$

is relevant. Here, p_{exact} denotes the exact rejection probability, while p_{approx} is the approximate rejection probability given by the limit distribution (here: χ^2-distribution with 15 degrees of freedom).
Note: We use p_{approx} instead of p_{exact} in the denominator because the designer of the PTRNG bases his further considerations on p_{approx}.

1102 [relative approximation error] In the example from par. 1098 the sample size $m = 320$ is rather small. In fact, for ideal RNGs the relative approximation error is 10.1. (Exploiting the symmetries allows the calculation of the exact rejection probability.) In this case the developer would have underestimated the significance level of the online test (i.e., the number of (undesired) noise alarms under the null hypothesis) by a factor of more than 10. This may primarily affect the availability of the PTRNG but for other statistical tests, the approximation error might swing into the opposite direction, leading to a significant overestimation of the significance level, which definitely would be a security issue.

1103 The relative approximation error should decrease for an increasing sample size m; in the above example, e.g., to $m = 512$ or $m = 1024$, which are more typical sample sizes for poker tests. However, the considerations from pars. 1100 to 1102 point to a general problem when using limiting distributions.

1104 A further disadvantage of the online test from par. 1097 (in particular, for sample size $m = 320$ as in par. 1099) is that it is hardly feasible to estimate the true rejection probabilities if the distribution of the raw random numbers deviates from the output of an ideal RNG (due to a bias and dependencies). However, this is relevant to assess the suitability of the online test.
Note: A suitable online test shall reliably separate the sets of parameters A_{real} and A_{bad}.

1105 If the developer cannot (at least approximately) determine the failure probabilities for the relevant parameters in A_{real} and A_{bad}, there is a lack of evidence whether Requirement PTG.2.4 (resp. PTG.3.7) is indeed fulfilled, i.e., whether the online test is effective. Then the PTRNG cannot be certified to be PTG.2- or PTG.3-compliant.

1106 Remark: For iid stochastic models a monobit test may be applied. Then the Central Limit Theorem provides approximate rejection probabilities of the monobit test for iid $B(1,p)$-distributed random variables and (4.41) provides an upper bound for the approximation error. In principle, this would solve the problems addressed in pars. 1104 to 1105.

1107 However, the upper bound (4.41) converges in the order of $n^{-0.5}$, which means that the sample size of the monobit test had to be very large if extremely small rejection probabilities are concerned. Furthermore, the developer had to show that the proposed monobit test (with specified sample size m and evaluation rules) is sufficiently discriminating between A_{real} and A_{bad}.

These problems motivate the search for more sophisticated online test schemes that apply more complex evaluation rules than just considering individual tests; cf. Subsection 5.5.2. 1108

The key is to analyze the distribution of the test variable C for different parameters. The expectation $E(C)$, the variance $\mathrm{Var}(C)$, and the standard deviation $\sigma_C := \sqrt{\mathrm{Var}(C)}$ as well as probabilities $\mathrm{Prob}(C \in E)$ (e.g., $\mathrm{Prob}(C > x)$) can easily be estimated by stochastic simulations. The latter (rejection) probabilities $\mathrm{Prob}(C_j \in E)$ may be small on A_{real} (let's say, $\in [10^{-4}, 10^{-2}]$) but shall not be tiny (let's say, $< 10^{-6}$) since otherwise, the required sample size for (trustworthy) stochastic simulations would 'explode'. 1109

For stochastic simulations it is not necessary to use a DRNG that is suitable for cryptographic applications. Instead, one may use a linear congruential generator or a linear feedback shift register since both types of pseudorandom number generators are very fast, and their statistical properties are suitable for this purpose (cf. [Schi09a], Subsection 2.4.3). 1110

From so-called standard random numbers $z_1, z_2, \ldots \in [0, 1)$ (generated, e.g., by a linear congruential generator or a linear feedback shift register), one generates sequences of pseudorandom bits for different parameters. $B(1, p)$-distributed random numbers, for example, can be obtained via $r_j := 1_{\{z_j \leq p\}}$. The simulated standard random numbers are assumed to be uniformly distributed on $[0, 1)$. The specified test (e.g., a χ^2 goodness-of-fit test on 4-bit words or a monobit test) is applied to the simulated raw random numbers. Finally, these empirical values yield estimates for $E(C)$, $\mathrm{Var}(C)$, σ_C and $\mathrm{Prob}(C \in E)$. 1111

Table 9 provides exemplary results for the above-mentioned χ^2 goodness-of-fit test on 4-bit words. For simplicity, only iid random variables $R_1, R_2, \ldots$ are considered. However, it is easy to simulate other distributions of the raw random numbers (e.g., for Markovian models) as well. It might be noted that for $p = 0.500$ for each sample size m, the χ^2-approximation yields $E(C) = 15.0$ and $\mathrm{Var}(C) = 5.477$. Table 9 and Table 10 collect simulation results for different parameter values and sample sizes, 1112

Table 9: χ^2 goodness-of-fit test on 4-bit words: simulation results for iid raw random numbers ($R_j \sim B(1, p)$); $N_s = 10^6$ or 2^{20}

criteria	$p = 0.500$	$p = 0.503$	$p = 0.507$	$p = 0.510$	$p = 0.520$
$m = 512$					
$E(C)$	15.01	15.02	15.10	15.20	15.81
σ_C	5.46	5.46	5.49	5.53	5.76
$\mathrm{Prob}(C > 34.0)$	0.0036	0.0035	0.0038	0.0041	0.0062
$m = 1024$					
$E(C)$	15.00	15.04	15.19	15.41	16.64
σ_C	5.46	5.48	5.53	5.62	6.05
$\mathrm{Prob}(C > 34.0)$	0.0035	0.0036	0.0040	0.0045	0.0098
$m = 2^{20}$					
$E(C)$	15.01	52.79	229.43	434.9	1696.69
σ_C	5.47	13.4	29.24	41.4	82.53
$\mathrm{Prob}(C > 150.0)$	0.0000	0.0000	0.9955	1.000	1.000

Table 10: χ^2 goodness-of-fit test on 4-bit words: simulation results for iid raw random numbers $(R_j \sim B(1,p))$;$N_s = 10^6$

criteria	$p = 0.570$	$p = 0.560$	$p = 0.540$	$p = 0.530$	$p = 0.520$
$m = 512$					
$E(C)$	25.23	22.47	18.20	16.84	15.81
σ_C	8.74	7.93	6.63	6.13	5.76
Prob$(C > 34.0)$	0.1489	0.0825	0.0215	0.0110	0.0062
$m = 1024$					
$E(C)$	35.60	30.00	21.60	18.69	16.64
σ_C	11.06	9.77	7.59	6.73	6.05
Prob$(C > 34.0)$	0.5156	0.3065	0.0645	0.0248	0.0098
$m = 2^{16}$					
$E(C)$	1337.71	979.17	438.51	252.17	120.12
σ_C	76.48	64.52	42.19	31.52	21.30
Prob$(C > 600)$	1.0000	1.0000	0.0002	0.0000	0.0000

1113 The figures in Table 9 show that single χ^2 goodness-of-fit tests on 4-bit words reliably separate different parameters if the sample size m is extremely large. Of course, $m = 2^{20} = 1{,}048{,}576$ is by far too large for a single online test, but it comes into question for the test mode after a noise alarm has occurred or as an additional criterion for online test suites; cf. pars. 716, 1146, 1147, and 1127.

1114 Table 10 demonstrates that if the parameters differ more significantly, much smaller sample sizes m suffice to distinguish these parameters. While Table 9 primarily concerns PTRNG designs where the raw random numbers have enough entropy, Table 10 applies to designs that need data-compressing algorithmic post-processing, e.g., XORing non-overlapping pairs of raw random number bits. In the second scenario the raw random numbers may show considerable weaknesses. This facilitates the efforts of designing efficient tests.

1115 Assume that non-overlapping pairs of the raw random numbers are XORed. If $p = 0.44$ or $p = 0.47$, for example, the internal random numbers are $B(1,p')$-distributed with $p' = 0.5072$ or $p' = 0.5018$, respectively.

5.5.2 A more sophisticated online test procedure

1116 In this subsection we provide an *example* of a generic approach that mitigates several problems mentioned in Subsection 5.5.1. The considerations are based on [Schi01] but also go beyond. Pre-versions are explained in [AIS31An_01], Example E.7, and [AIS2031An_11], Subsect. 5.5.3. Note: The requirements on online tests have increased since then (mainly because of the increased entropy requirements).

1117 The central idea is not just to consider independent, single statistical tests but to combine the information from several statistical tests by suitable evaluation rules.

1118

Other online test schemes are permitted, too, of course. It is *not claimed* that the online test scheme discussed in this subsection is optimal! The suitability of an online test scheme depends on the concrete scenario; c.f. par. 1157.

At first the developer selects a so-called 'basic test' that is tailored to the stochastic model. For the remainder of this subsection, this selection will not be discussed. We assume that the choice of the basic test is appropriate. As in Subsection 5.5.1 m denotes the sample size of the basic test in bits. 1119

[start-up test] The basic test may also serve as start-up test (single application with an appropriate rejection boundary, such that an ideal RNG would fail, let's say, with probability $\approx 10^{-8}$ or 10^{-7}). The start-up test shall detect any total breakdown of the noise source and significant statistical weaknesses immediately when the PTRNG is started. The start-up test then fulfills functional requirement PTG.2.3 and PTG.3.6. 1120

Let $C_1, C_2, \ldots$ denote the random variables that correspond to the test values $c_1, c_2, \ldots$ of the basic tests within an online test suite. Furthermore, $H_0, H_1, \ldots$ are the so-called 'history variables'. We set 1121

$$H_0 := E\left(C_{1;\mathrm{IRNG},t}\right) \tag{5.169}$$

where $E(C_{1;\mathrm{IRNG},t})$ denotes the expectation of the test variable C_1 for an ideal RNG, rounded to a multiple of 2^{-t}. Moreover, we define the recursion

$$H_j := (1-\beta)\, H_{j-1} + \beta C_j \quad \text{for } j = 1, 2, \ldots \quad \text{with } \beta = 2^{-s} \tag{5.170}$$

where the H_j are rounded to t binary digits after the binary point. This allows the calculation of the 'history values' $h_0, h_1, \ldots$ by integer arithmetic.

[online test suite] In Step j of the online test suite, a basic test is applied ($\rightarrow$ test value c_j), and two criteria are evaluated unless a noise alarm has occurred within this online test suite before. The online test suite consists of N basic tests unless a noise alarm has occurred before. If no noise alarm has occurred during the N basic tests, a further test is applied. The evaluation rules (ER 1a), (ER 1b), (ER 2), and (ER 3) specify these criteria. 1122

(ER 1a) $c_j \in E_{1a} \Rightarrow$ noise alarm

(ER 1b) $c_{j-k+1}, \ldots, c_{j-1}, c_j \in E_{1b} \Rightarrow$ noise alarm

(ER 2) $h_j := (1-\beta)\, h_{j-1} + \beta c_j \in E_2 \Rightarrow$ noise alarm

(ER 3) at the end of the online test suite, if no noise alarm has occurred:
Apply the basic test to all raw random numbers that were tested within this online test suite (test value c_{total}).
$c_{total} \in E_3 \Rightarrow$ noise alarm

The parameter $k > 1$ is a small integer. A noise alarm terminates the current online test suite. If not interrupted by a noise alarm, an online test suite consists of N basic tests. If the online test suite has been terminated earlier due to a noise alarm, evaluation criteria ER 3 is not applied.

The evaluation criteria (ER 1a), (ER 1b), (ER 2), and ER 3 cover different aims. This topic will be taken up later. Possible consequences of a noise alarm are explained in pars. 716 to 718.

1123 Since the class requirements PTG.2 and PTG.3 do not permit (significant) long-term dependencies of the raw random numbers, we may assume that the random test variables $C_1, C_2, \ldots$ are iid.

1124 In par. 1122, the particular evaluation criteria serve different aims; cf. par. 1130.
Note: The probabilities $\mathrm{Prob}(C_j \in E_1)$ can be estimated by stochastic simulations for relevant parameters; cf. pars. 1109 to 1115.

1125 [Time-local stationarity] We assume that the raw random numbers can be viewed as stationarily distributed within an online test suite.

1126 The proposed online test scheme from par. 1122 is generic. To make the ideas more concrete, in the remainder of this subsection, we assume, as an example, that the basic test is given by a χ^2 goodness-of-fit test on 4-bit words. The sample size is m bits, or equivalently, $m/4$ four-bit words.
Note 1: The considerations can be transferred to any other basic test.
Note 2: (Reminder) Universally suitable online tests do not exist. The online test shall be tailored to the physical noise source. In Subsect. 5.4.3, for example, the online test considers the expectation and the variance of integer-valued raw random numbers.

1127 [online test suite: (special case: basic test = χ^2 goodness-of-fit test)] Local counters $fr(0)$ to $fr(15)$ count the number of 4-bit words within a basic test that equal $0, 1, \ldots, 15$ (interpreted as the binary representation of integers). At the beginning of each online test suite, global counters are initialized: $tot_{fr}(0) = \cdots = tot_{fr}(15) = 0$.
In Step j of the online test suite, a basic test is applied ($\rightarrow$ test value c_j); the evaluation criteria (ER 1a), (ER 1b), and (ER 2) are evaluated; and the global counters $tot_{fr}(0)$ to $tot_{fr}(15)$ are updated. A noise alarm terminates the online test suite. Unless the online test suite has been terminated by a noise alarm, the final evaluation criterion (ER 3) is applied.

(ER 1a) $c_j > x_a \Rightarrow$ noise alarm

(ER 1b) $c_{j-k+1}, \ldots, c_{j-1}, c_j > x_b \Rightarrow$ noise alarm

(ER 2) $h_j := (1-\beta)\, h_{j-1} + \beta c_j \notin [u, v] \Rightarrow$ noise alarm

(ER 3) for $i = 0$ to 15 do $tot_{fr}(i) := tot_{fr}(i) + fr_{(j)}(i)$
After all N basic tests have been evaluated:
$c_{total} > x_{total} \Rightarrow$ noise alarm

1128 [evaluation criteria] As already mentioned above the four evaluation criteria (ER 1a), (ER 1b), (ER 2), and (ER 3) serve different purposes. The sample size of the basic tests and thus, their discriminatory power, is usually much smaller than that of typical evaluator tests. Evaluation rule (ER 3) is applied at most once per online test suite. Due to its large sample size, the discriminatory power of this final test is very large. The extra costs are limited to 16 additional integer counters and the computation of one χ^2 test value.

Generally, one can expect that aging effects or changing environmental conditions (apart from attacks) change the entropy of the generated raw random numbers slowly. As will become clear below, such phenomena are reliably detected by (ER 3). Problems can arise shortly after start-up if the PTRNG in operation behaves very differently from typical copies of the same series (and, of course, within targeted attacks that are not yet in the scope of the online test). The evaluation criteria (ER 1a,b), (ER 2), and (ER 3) serve different goals. 1129

[aims of the evaluation criteria] The aims of the evaluation criteria (ER 1a) and (ER 1b) are to detect rapidly developing, significant weaknesses of the raw random numbers, which have to be detected very quickly. Evaluation rule (ER 2) mitigates the problem of the small sample size of a basic test to some degree, since the history values $h_1, h_2, \ldots$ are sensitive to deviations of the 'true' expectation $E(C_j)$ (depending on the true parameters). The task of evaluation criterion (ER 2) is to detect smaller (but still non-acceptable) weaknesses that are also rapidly developing. Finally, due to its large sample size, the aim of (ER 3) is the reliable (and sufficiently fast) detection of slowly developing small weaknesses, i.e., when the parameters (slowly) leave the set A_{good}. 1130

[total failure test] In principle, the online test schemes specified in pars. 1122 and 1127 can also include an evaluation rule that fulfills the requirements of a total failure test. This option was discussed in [Schi01] (cf. [AIS31An_01], Example E.7, and [AIS2031An_11], Subsect. 5.5.3) under the assumption that a total failure would imply a constant sequence of raw random number bits. A further evaluation rule was added to par 1127: 1131

(T) $c_j \geq 269.5 \Rightarrow$ noise alarm (total failure)

[total failure test] The disadvantage of the approach from par. 1131 is that the actual basic test may not detect a total failure if it occurs too late in the test sample. If the last 220 bits of a test sample ($m = 512$) are constant 0 or constant 1, then $c_j \geq 269.5$, which triggers a noise alarm due to decision rule (3) [Schi01]. This means that the detection of a noise alarm might be delayed by $219 + 512 = 731$ raw random number bits in the worst-case. Hence, the PTRNG design must provide a large buffer for the internal random numbers. Specially designed total failure tests usually have much smaller delay times and thus are preferable in most cases. 1132

Compared to the online test suite discussed in [Schi01], the proposed solutions from pars. 1122 and 1127 lack a total failure test. Instead, the evaluation criteria (ER 1a) and (ER 3) have been added. 1133

Compared to online test schemes that apply independent statistical tests, the proposed online test scheme has several advantages. First of all, it is feasible to estimate the probabilities of noise alarms for any distribution of the raw random numbers. Secondly, there is a whole set of parameters $(m, N, k, t, \beta, E_{1a}, E_{1b}, E_2, E_3)$, resp. $(m, N, k, t, \beta, x_a, x_b, u, v, x_{total})$, which allow 'fine-tuning', i.e., the optimization of the online test scheme under consideration of the PTRNG design. 1134
Note: The parameters x, u, v are specific for the selected χ^2-test.

1135

Evaluation rule (ER 2) defines a random walk on $\{u, u + 2^{-t}, \ldots, v\}$. Without rounding the history variables to a multiple of 2^{-t}, the expectations $E(H_j)$ would tend to $E(C_1)$ as $j \to \infty$. If $E(C_1) \notin [u, v]$ the history variables should 'drift out' of $[u, v]$ rather soon, causing a noise alarm. Even if $E(C_1) \in [u, v]$, a deviation of $E(C_1)$ from $E(C_{1;\mathrm{IRNG},t})$ should increase the probability that an absorbing barrier is reached earlier (provided, of course, that u and v have been properly selected.

1136 By par. 1123 the evaluation rules (ER 1a,b) and (ER 2) in par. 1127 can be modeled by a homogeneous Markov chain on the finite state space

$$\Omega = \left\{\left(2^{-t}\ell, i\right) \mid \ell \in \mathbb{N}, 2^{-t}\ell \in [u, v], 0 \leq i < k\right\} \bigcup \{\omega\} . \tag{5.171}$$

Recall that $k > 1$ is a small integer. Furthermore, u and v are multiples of 2^{-t}, while ω is an absorbing state. The online test scheme reaches state (s, i) after step j if the history variable $h_j = s$ and if $c_{j-i} \leq x$ but $c_{j-i+1}, \ldots, c_j > x$ (or if $j = i < k$). The absorbing state ω is reached when a noise alarm has occurred within the first j steps; see [Schi01] for details.

1137 The state space Ω consists of $((v-u)2^t+1)k+1$ elements. The initial distribution ν_0 has total mass on the state $(E(C_{1;IRNG,t}), 0)$. If P denotes the transition matrix on Ω, then $\nu_j(\omega) = \nu_0 P^j(\omega)$ equals the probability that a noise alarm has occurred within the first j steps.

1138 The probability $\mathrm{Prob}(C_j > x)$ is estimated, and the transition matrix P is determined on the basis of stochastic simulations. For each relevant distribution random numbers are simulated and basic tests are performed. This provides the empirical cumulative distribution function of the random variables C_j.

1139 A small weight factor β ensures that evaluation rule (2) in par. 1122 (resp. in par. 1127) does not depend on the occurrence of a single, very rare event but on several events that, taken individually, need not be rare. As the values of $\beta = 2^{-s}$ become smaller, the history values $h_0, h_1, \ldots$ become more inert. Reasonable values seem to be $s = 4, 5, 6$.

1140 The choice of the precision 2^{-t} also has impact on the probabilities for a noise alarm.

1141 [(ER 1b)] As explained above, the probability for a noise alarm within an online test suite can be calculated for any set of parameters. To support a targeted search for appropriate parameters, we consider the question about how many basic tests are needed on average until k successive failures occur (each with a probability $p \in (0, 1)$). For $j = 0, \ldots, k$, let $e_k(j)$ denote the expected number of basic tests until k successive failures occur under the condition A_j that the j previous basic tests failed. If $j < k$, then condition A_j implies condition C_0 in the next step with probability $1 - p$ and condition A_{j+1} with probability p. Condition A_k is the terminating condition. This leads to the following linear equations.

$$e_k(0) = (1-p)\left(e_k(0)+1\right) + p\left(e_k(1)+1\right) \Longleftrightarrow pe_k(0) - pe_k(1) = 1 \tag{5.172}$$

$$e_k(j) = (1-p)\left(e_k(0)+1\right) + p\left(e_k(j+1)+1\right) \Longleftrightarrow (p-1)e_k(0) + e_k(j) - pe_k(j+1) = 1 \tag{5.173}$$

$$\text{for } j = 1, \ldots, k-1$$

$$e_k(k) = 0 \tag{5.174}$$

The solution of the linear equations, in particular, yields $e_k(0)$, the value we are interested in.

1142

[(ER 1b), Example] For $k = 4$ we obtain the linear equations

$$\begin{pmatrix} p & -p & 0 & 0 & 0 \\ p-1 & 1 & -p & 0 & 0 \\ p-1 & 0 & 1 & -p & 0 \\ p-1 & 0 & 0 & 1 & -p \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} e_4(0) \\ e_4(1) \\ e_4(2) \\ e_4(3) \\ e_4(4) \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 0 \end{pmatrix} \tag{5.175}$$

In particular, $e_4(0) = \frac{p^3+p^2+p+1}{p^4}$. Thus, $e_4(0) \approx \frac{1}{p^4}$ for small p.

[(ER 1b), Example] For $k = 5$ we obtain the linear equations 1143

$$\begin{pmatrix} p & -p & 0 & 0 & 0 & 0 \\ p-1 & 1 & -p & 0 & 0 & 0 \\ p-1 & 0 & 1 & -p & 0 & 0 \\ p-1 & 0 & 0 & 1 & -p & 0 \\ p-1 & 0 & 0 & 0 & 1 & -p \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} e_5(0) \\ e_5(1) \\ e_5(2) \\ e_5(3) \\ e_5(4) \\ e_5(5) \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \end{pmatrix} \tag{5.176}$$

In particular, $e_5(0) = \frac{p^4+p^3+p^2+p+1}{p^4}$. Thus, $e_5(0) \approx \frac{1}{p^5}$ for small p.

PTRNG designs usually generate much better raw random numbers than required by PTG.2.2 1144
and PTG.3.5. As explained in Subsection 4.5.3, pars. 690 to 697, this eases the design of an effective and efficient online test. The probability for an (undesired) noise alarm for the 'very acceptable' parameters A_{real} should be small, while it should be large for inappropriate parameters A_{bad}. For the remaining 'in-between' distributions $A_{good} = A_{real} \setminus A_{bad}$, the probability for a noise alarm is not relevant.

[Quality assessment] The suitability of an online test primarily depends on the stochastic model 1145
but also on the parameter sets A_{real} and A_{bad}. Other aspects are, for example, the test strategy and the output rate of the PTRNG, which affects the number of accidental noise alarms. For simplicity, we assume an iid model as an example for the remainder of this section. The general procedure would be the same for a Markovian model.

[consequences of a noise alarm] In Subsection 4.5.3 several options for the consequences of a 1146
noise alarm were addressed; cf. pars. 716 to 718. One of these options is to trigger a test mode (without outputting any random numbers, i.e., an 'emergency test') in order to check whether the noise alarm was justified or accidental. In the following we focus on this possibility. The 'emergency test' can be a single basic test but with much larger sample size.

[Example: emergency test] A reasonable strategy is to apply the basic test to fresh Nm raw 1147
random numbers bits (the sample size of evaluation criterion (ER 3)). The decision rule can be the same as for (ER 3), but another decision rule can be selected as well. If the emergency test fails, the noise alarm is confirmed; otherwise, considered erroneous.
Natural requirement: If the raw random numbers belong to $\in A_{bad}$, this should be detected with overwhelming probability (cf. Tab. 9 and Tab. 10). On the other hand, if the true parameter(s) are in A_{real}, a failure of the emergency test should be very unlikely.

1148

[Scenario I, iid stochastic model] The designer is convinced that $\mathrm{Prob}(R_j = 1) \in (0.497, 0.503)$ for all properly working examples of this PTRNG design (e.g., for the PTRNGs on the chips of some product series). This is considerably better than required by requirement PTG.2.2 (resp. by PTG.3.5), and this feature supports the design of an effective online test. In the notation of Subsection 4.5.3, this means that $A_{real(I)} = [0.497, 0.503]$ and $A_{bad(I)} = [0, 0.4931) \cup (0.5069, 1]$. For for the parameters in $A_{good(1)} = [0.0, 1.1] \setminus A_{bad(I)}$, both the Shannon entropy and min-entropy of the corresponding distributions exceed the bounds that are specified in PTG.2.2 (0.9998 for Shannon entropy and 0.98 for min-entropy). Thus, there is no need for algorithmic post-processing.

1149 [Scenario II, iid stochastic model] As in par. 1148 we assume an iid model but here, $A_{real(II)} = [0.470, 0.530]$ and $A_{bad(II)} = [0, 0.441) \cup (0.559, 1]$. This requires a data-compressing algorithmic post-processing. The algorithmic post-processing XORs non-overlapping pairs of consecutive raw random number bits.

1150 [Example: Scenario I, test parameters] $(m, N, k, \beta, t, x_a, x_b, u, v, x_{total}) = (1024, 1024, 5, 1/32, 5, 75.0, 34.0, 11.0, 20.5, 150.0)$. If a noise alarm has occurred, the PTRNG goes into the test mode (cf. par. 1147) and performs an emergency test. The tested raw random numbers are not output. As for the evaluation rule (ER 3), the emergency test applies the χ2-test to $2^{20} = 1{,}048{,}576$ bits (the sample size of an online test suite). The emergency test fails if the test value is > 150.0. If the emergency test is passed, the PTRNG returns to the working mode (outputting random numbers).

1151 [Example: Scenario I, numerical values] Tab. 11 collects numerical results that illustrate the properties of the test parameters that were selected in par. 1150. Recall that 0.503 limits A_{real}, while (approximately) $p = 0.507$ defines the limit line between A_{good} and A_{bad}. The values in Tab. 11 are computed on the basis of the simulated cumulative distribution function of the test variable C_1 under the particular distributions of the raw random numbers, cf. par. 1109. Criterion (ER 1a) detects, with a probability of almost 1 (of > 0.9, of > 0.5) if $p \in [0, 0.34] \cup$

Table 11: The first row provides the average number of basic tests until 5 consecutive basic test values exceed 34.0 (event A_5). The second row quantifies the probability that a noise alarm is triggered by evaluation criterion (ER 1a,b) or (ER 2), while the third row contains the probabilities that the (ER 3) causes a noise alarm. Both row 2 and row 3 refer to a single online test suite.

p=	0.500	0.503	0.507	0.525	0.530	0.535	0.560	0.570
E(# basic tests until A_5 occurs)	$1.9 \cdot 10^{12}$	$1.7 \cdot 10^{12}$	$9.8 \cdot 10^{11}$	$1.2 \cdot 10^{9}$	$1.1 \cdot 10^{8}$	$8.8 \cdot 10^{6}$	532	55
Prob(noise alarm by (ER 1a,b) or (ER 2))	0.00000	0.00000	0.00000	0.05856	0.85597	1.00000	1.00000	1.00000
Prob(noise alarm by (ER 3))	0.00000	0.00000	0.99552	1.00000	1.00000	1.00000	1.00000	1.00000

$[0.66, 1]$ (if $p \in [0, 0.36] \cup [0.64, 1]$, if $p \in [0, 0.38] \cup [0.62, 1]$), while the probability is essentially 0 if $p \in [0.43, 0.57]$. If $p \in [0.43, 0.57] \cap A_{bad}$ the other criteria apply.

[Example: Scenario I, numerical values] Tab. 11 demonstrates the different aims of the particular evaluation rules; cf. par. 1130. Evaluation criterion (ER 3) reliably separates A_{real} from A_{bad}. The absolute time that an online test suite requires depends on how many raw random numbers are generated per second. For typical PTRNGs it should not last longer than a few seconds. 1152

[Example: Scenario II] $(m, N, k, \beta, t, x_a, x_b, u, v, x_{total}) = (512, 128, 4, 1/32, 5, 75.0, 34.0, 11.0, 23.0, 600.0)$. If a noise alarm has occurred, the PTRNG goes into the test mode (cf. par. 1147) and performs an emergency test. The tested raw random numbers are not output. As for the evaluation rule (ER 3), the emergency test applies the $\chi 2$-test to $2^{16} = 65536$ bits (sample size of an online test suite). The emergency test fails if the test value is > 600.0. If the emergency test is passed, the PTRNG returns to the working mode (outputting random numbers). 1153

[Example: Scenario II, numerical values] Tab. 12 collects numerical results that illustrate properties of the test parameters that were selected in par. 1153. Recall that 0.53 limits A_{real} while (approximately) $p = 0.56$ defines the limit line between A_{good} and A_{bad}. The values in Tab. 12 are computed on the basis of the simulated cumulative distribution function of the test variable C_1 under the particular distributions of the raw random numbers, cf. par. 1109. 1154

Table 12: The first row provides the average number of basic tests until 4 consecutive basic test values exceed 34.0 (event A_4). The second row quantifies the probability that a noise alarm is triggered by evaluation criterion (ER 1) or (ER 2), while the third row contains the probabilities the (ER 3) causes a noise alarm. Both row 2 and row 3 refer to a single online test suite.

p=	0.520	0.530	0.560	0.570	0.580	0.590	0.60
E(# basic tests until A_4 occurs)	$6.9 \cdot 10^8$	$6.8 \cdot 10^7$	$2.3 \cdot 10^4$	2387	335	72	23
Prob(noise alarm by (ER 1) or (ER 2))	0.00000	0.000002	0.64744	0.99983	1.00000	1.00000	1.00000
Prob(noise alarm by (ER 3))	0.00000	0.00000	1.00000	1.00000	1.00000	1.00000	1.00000

Criterion (ER 1a) detects, with a probability of almost 1 (of > 0.9, of > 0.5) if $p \in [0, 0.34] \cup [0.66, 1]$ (if $p \in [0, 0.36] \cup [0.64, 1]$, if $p \in [0, 0.38] \cup [0.62, 1]$), while the probability is essentially 0 if $p \in [0.43, 0.57]$. If $p \in [0.43, 0.57] \cap A_{bad}$ the other criteria apply.
Note: The discriminatory power of the evaluation rules (ER 1a), (ER 1b), and (ER 2) is rather high, even at the boundary to A_{bad}. It could be an option to omit evaluation criterion (ER 3) if the device is resource-constrained, possibly by simultaneously increasing the number N of basic tests.

[Example: Scenario II, numerical values] Tab. 12 demonstrates the different aims of the particular evaluation rules; cf. par. 1130. It is obvious that in Scenario II, far fewer online tests suffice than in Scenario I. 1155

1156

Note: The online test schemes specified in pars. 1122 and 1127 restart after N basic tests or after a noise alarm has occurred (provided that the noise alarm turned out to be erroneous). The limitation to at most N basic tests was introduced to simplify the computation of the probabilities for noise alarms. Of course, alternative designs where a test suite continues until a noise alarm occurs can also be appropriate.

1157 If the online test separates the sets A_{bad} and A_{real} better than the χ^2 test in this subsection, simpler online test (procedures) can be applied. An example would be the online test discussed in Subsect. 5.4.3, see pars. 1033, for suitable parameters s/μ and σ/μ.

1158 [Example] The designer expects only a few requests for internal random numbers per day. To save time and energy needed to perform a continuous online test, the PTRNG design buffers internal random numbers that are ready for output. As long as the buffer does not require fresh internal random numbers, random numbers generated by the PTRNG are discarded and not tested. The buffer is refilled when the number of remaining internal random numbers falls below a specified lower bound. Then the online tests are applied again.

1159 [Example, ctd.] Such an approach is principally acceptable but the online test has to adjusted to this situation. In particular, it does not suffice, e.g., to apply the online test procedure from par. 1127 and to continue the online test suite where it was interrupted after the buffer had been filled the last time. The reason is that between the subsequent basic tests, a large period of time may have elapsed. Thus, one aim of the online test suite, detecting slow drifts of the parameters, cannot be assured. However, it could be an option to apply an emergency test first (without outputting internal random numbers), and then to resume with continuously applied online tests.

5.6 Linux /dev/random and /dev/urandom

1160 The Linux operating system includes two RNG interfaces as part of the kernel:

- the random number generator `/dev/random/`
- the random number generator `/dev/urandom/`

1161 [Linux kernel versions 5.6 to 5.16] Figure 25 provides a schematic overview of both `/dev/random/` and `/dev/urandom/`. Both RNGs extract entropy from different non-physical noise sources that either depend on actions of the user or on internal system tasks. The bottom line of Figure 25 lists several possible non-physical noise sources. Every single event (e.g., keyboard and mouse actions; access to the hard disk; interrupt timestamps) is mapped to a bit string. The bit strings are mixed into a register called `input_pool` using a linear-feedback shift register. For the interrupt noise source, there is an additional register per CPU called `fast_pool` that accumulates several interrupt timestamps before its content is mixed into the `input_pool`. The entropy of the incoming raw random numbers is estimated using conservative heuristic rules, and an 'entropy counter' keeps track of the entropy supposedly contained in the `input_pool` at any time. Upon an internal request, seed material is generated from the `input_pool` using an output function

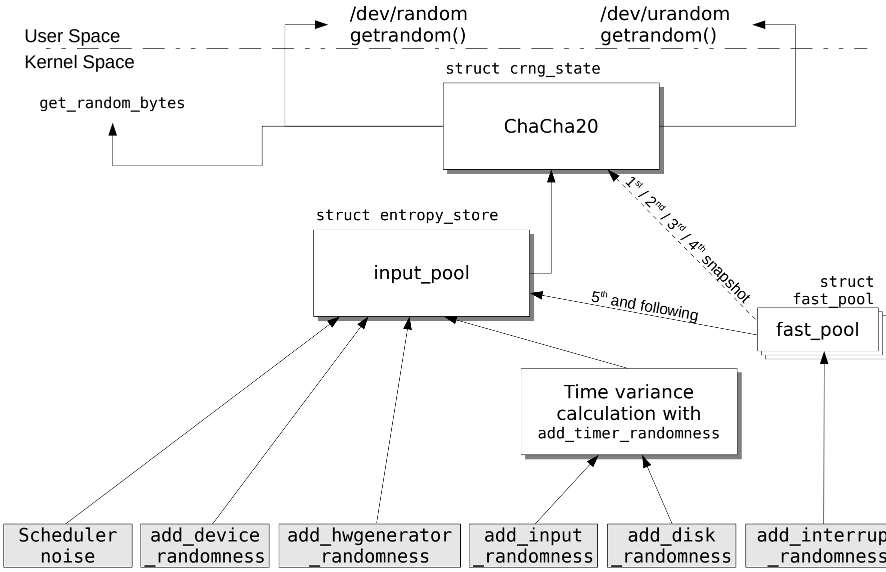

Figure 25: Functional design of the Linux NPTRNG (as of kernel version 5.6); source: [Linux_RNG_2022]

based on SHA-1 (with feedback into the `input_pool`). The number of bits returned from the `input_pool` is thereby limited by the current amount of entropy in the pool and subsequently subtracted from the entropy counter. Both `/dev/random` and `/dev/urandom` use a DRNG based on the ChaCha20 cipher, which is seeded using the `input_pool`.

[up to Linux kernel 5.5] Up to Linux kernel version 5.5, `/dev/random` used an additional register (first the so-called `blocking_pool` and later the internal state of the ChaCha20-based DRNG), which was seeded by the `input_pool` and had its own entropy counter. A blocking mechanism prevented the output of internal random numbers if the `blocking_pool` did not contain enough 'unconsumed' entropy. In other words, `/dev/random` required that the `input_pool` and the `blocking_pool`, resp. the internal state of the ChaCha20-based DRBG, were updated continuously as they could not output more bits than the entropy that has (probably) been harvested by the noise sources. In contrast, `/dev/urandom` did not apply a blocking mechanism. 1162

[blocking vs. non-blocking] As mentioned in par. 1162, up to Linux kernel 5.5 `/dev/random` applied a blocking mechanism, a necessary feature of an NPTRNG to be NTG.1-compliant. As of Linux kernel version 5.6 to kernel version 5.17, this strategy was changed. Currently, care is taken that `/dev/random` blocks until the ChaCha20-DRNG is seeded properly. After this time no more blocking is applied. This increases the output rate of `/dev/random` but prevents NTG.1-compliance. Under suitable circumstances compliance with functionality class DRG.3 is possible. In contrast, `/dev/urandom` does not apply any blocking mechanism, not even for the initial seeding procedure. 1163

Since `/dev/random` and `/dev/urandom` are used by many cryptographic applications, in 2012, the BSI has initiated a permanent study in which `/dev/random` and `/dev/urandom` have been 1164 evaluated for each Linux kernel. The results are contained in reports [Linux_RNG_2016] (treating the Linux kernels 3.2.0 and 3.5 to 4.8), [Linux_RNG_2020] (treating the Linux kernels 4.9 to 5.5), and [Linux_RNG_2022] (treating the Linux kernels as of 5.6). All documents are available on the BSI website. While the first report is in German, the second and the third document are in English. The document [Linux_RNG_2022] is continuously supplemented by evaluation results for new kernels.

1165 The documents [Linux_RNG_2016], [Linux_RNG_2020], and [Linux_RNG_2022] do not only contain the final results of the evaluations but also explain details of the evaluation methodology. The general methodology is also applicable to other RNG designs that use non-physical noise sources. The evaluation entails a detailed description and analysis of all components of `/dev/random`, including the non-physical noise sources and entropy gathering functions, the input, output, and state transition functions of the entropy pools, the heuristics utilized by the entropy counter, and the ChaCha20-based DRNG. The analysis concludes that `/dev/random` significantly underestimates the collected entropy. The theoretical considerations are supplemented by empirical entropy estimates. A suite of entropy estimators is thereby applied to the raw random numbers recorded from the non-physical noise sources of an instrumented Linux kernel during boot time and regular operation.

1166 The document [Linux_RNG_overview] provides a table that lists the compliance of `/dev/random` to functionality class NTG.1 or DRG.3 for the Linux kernel versions beginning with 3.5. These results are yet only applicable to the RNG `/dev/random` if several requirements are met; cf. [Linux_RNG_overview], Notes, for details. In particular,

- The Linux system runs on a x86 platform.
- The CPU of the system has the RDTSC instruction.
- The clock frequency of the CPU is at least 1 GHz.
- The Linux system is not running in a virtual machine.
- The source files of the kernel that are relevant to `/dev/random` are unchanged as compared to the analyzed upstream version.

It is part of an evaluation to confirm these requirements.
Note: The class definitions refer to [AIS2031An_11].

1167 Table 13 summarizes the results from [Linux_RNG_overview].

1168 The report [RNG_virtual_env] considers random number generation in virtual environments.

Table 13: Conformity of `/dev/random` to NTG.1 and DRG.3; see [Linux_RNG_overview]

Linux kernel	**Conformity to functionality class**
3.5 – 3.14	NTG.1
3.15 – 3.16	—
3.17 – 3.19	NTG.1
4.1 – 4.20	NTG.1
5.1 – 5.5	NTG.1
5.6 – 5.17	DRG.3

Glossary

additional input Any data that are input to a hybrid DRNG *between* invocations of the seeding procedure or reseeding procedure. These data may be provided by an internal or external noise source; they may or may not contain entropy (e.g., predictable, low-entropy, high-entropy; they may be provided by a reliable source or be under the control of an adversary).

adversary A malicious entity whose goal is to determine, to guess, or to influence the output of an RNG. The term *attacker* is used synonymously.

algorithmic post-processing A type of post-processing that is normally used for the purpose of increasing the entropy per data bit (entropy extraction). It is usually applied to the raw random numbers. The name is chosen to distinguish it from an analog transformation (e.g., amplification, band-pass filter).
Note 1: Viewed as a mathematical function, algorithmic post-processing algorithms usually have small domains and small ranges.
Note 2: Typical examples of algorithmic post-processing algorithms: XORing bits or binary vectors, modular addition, linear feedback shift registers. Cryptographic algorithms are also permitted; cf. cryptographic post-processing for differentiation.

attacker Synonym for adversary.

backtracking resistance Term from NIST SP 800-90[A,B,C]
Note: Backtracking resistance is similar to enhanced backward secrecy.

backward secrecy Assurance that knowledge about previous output values cannot be derived with practical computational effort from the knowledge of current or subsequent output values.
Note: 'Deriving knowledge' means gaining significant advantage over blind guessing.

biased A value that is chosen from a sample space is said to be biased if one value is more likely to be chosen than another value. Contrast with unbiased.

bit string A finite sequence of ones and zeroes.

black box An idealized mechanism that accepts inputs and produces outputs. It is designed such that an observer cannot see inside the box or determine exactly what is happening inside that box. In contrast with a glass box.

compliant seed tree (AIS 20-specific term) A seed tree (or a branch thereof including the root DRNG) that satisfies the requirements specified in this document for allowing a DRNG to be seeded by another DRNG (cf. par. **??**.

compression rate (Average) ratio between the average input bit length of the cryptographic post-processing algorithm and the bit length of the resulting internal random numbers per (short) time interval; ideally holds for each internal random number.

computational security Security against an adversary with bounded computing power. Quantified by the security strength (of cryptographic mechanisms).

conceptual atomicity (AIS 20-specific term) The requests of a DRNG satisfy the condition of conceptual atomicity if the DRNG finishes every request by the application of the state transition function before any of the requested bits are used (cf. par. 119).
Note 1: In the context of CPU instructions, atomic refers to operations that are non-interruptible by other operations.

consuming application An application that uses random outputs from an RNG.

cryptographic State transition functions and output functions are considered cryptographic if they are composed of cryptographic primitives (e.g., block ciphers or hash functions).
Note: Incrementation by 1, simple XOR-additions, additions and multiplications in small moduli, LFSRs, and projections, for example, are not viewed as cryptographic.

cryptographic post-processing Stateful post-processing (i.e., with memory) for the purpose of gaining DRNG security properties (computational security). It is usually applied to intermediate random numbers, or to internal random numbers of a separate TRNG. It can also be applied to raw random numbers.
Note: By the definition given in this document, cryptographic post-processing is always stateful.

das-random number Digitized-analog-signal random number. A bit string that results directly from the digitization of analog noise signals in a PTRNG. Das-random numbers constitute a special case of raw random numbers.

deterministic random bit generator A term from NIST SP 800-90. Equivalent to DRNG.

deterministic RNG An RNG that produces random numbers by applying a deterministic algorithm from a secret initial value called a seed or seed material.
Note 1: A deterministic RNG at least has access to a randomness source initially.
Note 2: equivalent to DRBG (NIST SP 800-90)
Note 3: This document uses the abbreviation DRNG
.

digitization The process of generating raw discrete digital values from non-deterministic events (e.g., analog noise sources) within a noise source.
Note 1: Raw discrete digital values are called raw random numbers.
Note 2: In addition to the actual conversion of analog data into digital values, the digitization mechanism may include elementary operations like skipping values (thinning out), dropping bits (e.g., casting 10-bit-values to bytes by cutting the two least significant bits), or counting.

digitization process See digitization.

effective internal state The security-critical part of the internal state of a DRNG that an adversary does not know and that he cannot determine or guess (with probability that is significantly greater than indicated by its size (assuming optimal encoding) even if he has seen many random numbers(cf. par. 102).

enhanced backward secrecy Assurance that knowledge about previous output values cannot be derived with practical computational effort from the knowledge of the current internal state of an RNG.
Note 1: 'Deriving knowledge' means gaining significant advantage over blind guessing.

Note 2: The notion of enhanced backward secrecy is trivial for memoryless RNGs. Therefore, it is only a useful notion for DRNGs and hybrid PTRNGs, the security of which rests at least in part on cryptographic properties of the state transition function and the output function of the RNG.
Note 2: A term related to enhanced backward secrecy is backtracking resistance (from NIST SP 800-90[A,B,C]).

enhanced forward secrecy Assurance that knowledge about subsequent output values cannot be derived with practical computational effort from the knowledge of the current internal state of an RNG.
Note 1: 'Deriving knowledge' means gaining significant advantage over blind guessing.
Note 2: Pure DRNGs are unable to achieve enhanced forward secrecy. Unlike forward secrecy and backward secrecy as well as enhanced backward secrecy, enhanced forward secrecy rests entirely on the capability of inserting as much entropy as is required to make the prediction of future outputs infeasible.
Note 3: A term related to enhanced forward secrecy is prediction resistance (from NIST SP 800-90[A,B,C]).

entropy A measure of disorder, randomness, or variability in a closed system (see par. 515).
Note 1: The entropy of a random variable X is a mathematical measure of the amount of information gained by an observation of X.
Note 2: The most common concepts are Shannon entropy and min-entropy. In this document, Shannon entropy and min-entropy are used, depending on the context
Note 3: Min-entropy is the measure used in NIST SP 800-90.

entropy extraction The process of increasing the entropy per data bit. Requires compression.

entropy source (Term from NIST SP 800-90B)The combination of a noise source, health tests, and optional conditioning component that produce bit strings containing entropy. A distinction is made between entropy sources having physical noise sources and those having non-physical noise sources.
Note 1: The terms 'entropy source', 'health test', and 'conditioning' belong to NIST SP 800-90 [A,B,C].
Note 2: In the terminology of AIS 20/31, health tests comprise start-up tests, online tests, and total failure tests, while conditioning components correspond to postprocessing algorithms. In the terminology of NIST SP 800-90 [A,B,C] health tests comprise continuous tests and startup tests.
Note 3: A PTG.2-compliant PTRNG can be viewed as a (coarse) equivalent to a physical entropy source that generates random numbers whose entropy per bit is very close to 1.
Note 4: In the literature the term entropy source is often used synonymously with noise source.

external random number Internal random numbers that have been output by an RNG, i.e., those internal random number bits that are actually delivered to a consuming application.
Note 1: (DRNG): Some bits of the last internal random number of a request might not be output.
Note 2: (PTRNG): If the PTRNG runs continuously, many internal random numbers might never be output.

false positive In the context of AIS 31, an online test, total failure test, or start-up test signaling an error even though the component was actually working correctly.

forward secrecy Assurance that knowledge about subsequent output values cannot be derived with practical computational effort from the knowledge of current or previous output values.
Note: 'Deriving knowledge' means gaining significant advantage over blind guessing.

fresh entropy A random bit string that has not been previously used to generate output or has otherwise been made externally available.
Note: The noise source should be compliant with PTG.2, PTG.3, or NTG.1.

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

granularity level (AIS 20-specific term) Auxiliary term to express for which segments of the output of a DRNG security properties such as forward secrecy, backward secrecy, and enhanced backward secrecy hold (cf. par. 117).

hybrid DRNG A DRNG accepting additional input during operation or being able to trigger reseeding procedures.
Note: The second condition requires that the DRNG has access to a true RNG.

hybrid PTRNG A hybrid TRNG with physical noise source.

hybrid RNG An RNG that uses design elements from both DRNGs and TRNGs.
Note: This requires a stateful post-processing with memory. See also hybrid DRNG and hybrid TRNG.

hybrid TRNG A TRNG with cryptographic post-processing (with memory). Usually, the goal is to increase the computational complexity of the output sequence (computational security), and possibly also to increase the entropy per bit by data compression.
Note: Cryptographic post-processing may be viewed as an additional security anchor for the case where the entropy from the noise source per output bit is smaller than assumed.

ideal RNG A mathematical construct that generates independent and uniformly distributed random numbers.

information-theoretic security Security against an adversary with unlimited computing power. Requires fresh entropy.

intermediate random number (PTG.3- and NTG.1-specific term) input data for cryptographic post-processing.
Example: Consider a PTG.3-compliant RNGs that consists of a PTG.2-compliant PTRNG with DRG.3-compliant cryptographic post-processing. Here, the intermediate random numbers equal the internal random numbers generated by the PTG.2-compliant PTRNG.

internal random number Final stage of the random numbers of an RNG that are ready to be output (cf. par. 70. Compare to external random numbers.

internal state The collection of all secret and non-secret digitized information of an RNG as stored in memory at a given point in time.
Note: This also applies to post-processing algorithms for TRNGs.

Kerckhoffs's principle A security analysis is made under the basic assumption that the design and public keys of a cryptosystem are known by an adversary. Only secret keys and seed material are assumed to be unknown by an adversary.

known-answer test A test that uses a fixed input/output pair to test the correctness of a deterministic mechanism.

min-entropy A measure of entropy based on the minimal (worst-case) gain of information from an observation (see par. 519).

multi-target attack A scenario in which an adversary applies guesses or the results of a pre-computation to attack many instances of the same cryptosystem at once in hope that at least one instance succumbs to the attack.

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. A noise alarm can be a false positive.

noise source A source of unpredictable data that outputs raw discrete digital values. The digitization mechanism is considered part of the noise source. A distinction is made between physical noise sources and non-physical noise sources
Note: In AIS 31, raw discrete digital values are called raw random numbers.

non-physical noise source A noise source that typically exploits system data and/or user interaction to produce digitized random data.
Note 1: It is usually infeasible to determine a sufficiently precise characterization of non-physical noise sources. Therefore, designers have to resort to heuristics to obtain a conservative entropy lower bound.
Note 2: Non-physical noise sources are used by non-physical true RNGs (NPTRNGs)
Note 3: Examples for system data: RAM data, system time of a PC, or the output of API functions. Examples for interaction: key strokes, mouse movement, etc.

non-physical true RNG A true RNG with a non-physical noise source.

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 an input for a specific output that maps to this output [ISO_11770-3].

online test A quality check of the random numbers (usually the raw random numbers) while a PTRNG is in operation; usually realized by a statistical test or by a test procedure that applies several statistical tests; often used synonymously for online test procedure.

online test procedure Consists of one or several statistical tests (online tests), evaluation rules, a calling scheme, and the specified consequences of a noise alarm.

online test scheme Synonym for online test procedure.

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

personalization string An optional input value to a DRNG during instantiation to make one RNG instance behave differently from other instantiations.
Note: Can be a secret parameter or public parameter.

physical noise source A noise source that exploits physical phenomena (thermal noise, shot noise, jitter, metastability, radioactive decay, etc.) from dedicated hardware designs (using diodes, ring oscillators, etc.) or physical experiments to produce digitized random data.
Note: Dedicated hardware designs can use general-purpose components (like diodes, logic gates, etc.) if the designer is able to understand, describe, and quantify the characteristics of the design that are relevant for the generation of random numbers.
Note: Physical noise sources are used by physical true RNGs (PTRNGs).

physical true RNG A TRNG that uses a physical noise source.
Note 1: We use the shorthand 'physical RNG' instead of 'physical true RNG' because all physical RNGs are, by definition, true RNGs.
Note 2: We use the abbreviation "PTRNG" instead of "PRNG" to avoid confusion with pseudorandom number generators.

post-processing Generic term for any kind of transformation applied to random numbers at different stages in the generation of internal random numbers in a TRNG (e.g., applied to raw random numbers).
Note 1: Post-processing can have different goals: reducing bias or dependencies, statistical inconspicuousness, entropy extraction, DRNG fallback (computational security), etc.
Note 2: In this document we distinguish between algorithmic post-processing and cryptographic post-processing.
Note 3: Post-processing is related to the term conditioning function in NIST SP 800-90.

prediction resistance Term from NIST SP 800-90[A,B,C]
Note: Prediction resistance is similar to enhanced forward secrecy.

pseudorandom number generator Another term for a deterministic RNG.

pure DRNG A DRNG that does not accept input except during the seeding procedure or (externally triggered) reseeding procedure.
Note 1: Identical seed material values result in identical internal random numbers
Note 2: A pure DRNG is not able to trigger a reseeding procedure.

pure PTRNG A PTRNG in which any post-postprocessing is non-cryptographic or stateless cryptographic.
Note: A total failure of a pure PTRNG's noise source typically results in constant output or periodic patterns if no post-processing or stateless post-processing is implemented, or in weak pseudorandom output if simple (non-cryptographic) algorithmic post-processing is implemented.

random number generator A group of components or an algorithm that outputs sequences of discrete values (usually represented as bit strings called internal random numbers).

random variable Mathematical construct that models probabilistic behavior. A real-valued random variable X is a function that assigns a value of $\mathbb{R}$ to each outcome in the sample space Ω, i.e., $X : \Omega \to \mathbb{R}$.

raw random number Raw random numbers are discrete values (usually bits, bit strings, or integers) that are derived at discrete points in time from a noise source of a PTRNG or NPTRNG. Raw random numbers have not been significantly post-processed.
Note: For certain noise sources it may not be obvious which discrete values should be interpreted as the raw random numbers. For a meaningful analysis it is recommended to choose the earliest possible stage.

request (AIS 20-specific term) Operation of a DRNG to generate a requested amount of random numbers that is completed by the update of the state transition function. This operation usually consists of concatenating shorter internal random numbers produced by a 'core function' to a larger sequence (cf. par. 116).

reseed To refresh the internal state of a DRNG with seed material. The seed material should contain sufficient entropy to allow recovery from a possible compromise.
Note: (verb), corresponds to reseeding procedure.

reseeding procedure Refreshing the internal state of an DRNG with sufficient entropy to allow recovery from a possible compromise.
Note: A reseeding procedure may either utilize or ignore the previous internal state, but the former is recommended by this document. Occasionally, the first type of reseeding is called seed update.

secret parameter An optional input value to the seeding procedure or reseeding procedure of a DRNG or to the initialization of the cryptographic post-processing of a PTRNG to achieve additional security against adversaries who are not in possession of this value.

security boundary A physical or conceptual perimeter that confines the secure domain which an adversary cannot observe or influence in a malicious way (according to the chosen threat model).

security strength (of cryptographic mechanisms) A cryptographic mechanism achieves a security strength of n bits if costs equivalent to 2^n calculations of the encryption function of an efficient block cipher (e.g., AES) are tied to each attack against the mechanism that breaks the security objective of the mechanism with a high probability of success.

seed material A bit string that is used as input to a DRNG. The seed material determines a portion of the internal state of a DRNG. The seed material should contain sufficient entropy to meet security requirements.
Note: This definition also applies to the cryptographic post-processing algorithm (with memory) of a TRNG.

seed tree (AIS 20-specific term) A connected acyclic (directed or undirected) graph with a distinguished node, called the *root DRNG*. The root DRNG receives seed material directly from a TRNG. The seed tree is a tool to convey an overview of how different DRNGs in a system depend on each other and where the entropy and seed material for the respective seeding procedures and reseeding procedures comes from.
Note 1: Most importantly, it allows to ensure that a DRNG is not transitively seeding or reseeding itself.
Note 2: The concept of a (static) seed tree presumes that a DRNG is always seeded/reseed by the same TRNG or DRNG.

seeding procedure Procedure for seeding (initialization) of the internal state of a DRNG.

seedlife The period between the (re)seeding of the internal state of an RNG (typically, of a DRNG) and reseeding the internal state with the next seed material or uninstantiation of the DRNG.

self test Synonym for start-up test.

Shannon entropy A measure of entropy based on the expected (average) gain of information from an observation (see par. 517).

start-up test A test that is applied when the PTRNG has been started. It is intended to detect severe statistical weaknesses and total failures.

stationarily distributed In general this property of a sequence of random variables means that they form a stationary stochastic process. In the context of AIS 31, the term may also mean a relaxed condition called time-local stationarity if the random variables describe the behavior of a physical noise source.

stationary Depending on the context, the term stationary has two closely related, separate meanings in this document. For a stochastic process, it has the usual meaning of time-invariance (see par. 471). For a physical noise source (which can never satisfy this condition in a strict mathematical sense), we mean a relaxed condition that is more precisely denoted as time-local stationarity (see pars. 668 to 671).

statistical inconspicuousness The application of standard statistical tests does not distinguish the generated random numbers from ideal random numbers.

stochastic model A stochastic model provides a partial mathematical description (of the relevant properties) of a (physical) noise source using random variables. It allows the verification of a (lower) entropy bound for the output data (internal random numbers or intermediate random numbers) during the lifetime of the physical RNG, even if the quality of the digitized data goes down. The stochastic model is based on and justified by an understanding of the noise source
Note 1: Ideally, a stochastic model consists of a family of probability distributions that contains the true distribution of the noise source output (raw random numbers) or of suitably defined auxiliary random variables during the lifetime of the physical RNG.
Note 2: It may suffice to model parts of the entropy contributions if it can be shown that the neglected effects do not decrease the entropy..

sufficiently soon AIS31-specific term to quantify the ability of an online test to detect entropy defects of random numbers. What is meant by "sufficient" in terms of tested bits or internal random number bits depends on the possible entropy defects as described in the stochastic model. Due to the probabilistic nature of statistical testing, it is impossible to prescribe explicit conditions that shall be detected instantly. As a general rule, severe entropy defects can and shall be detected very quickly (i.e., with high probability). Smaller deviations from the acceptable behavior are more difficult to detect, but shall be detected after a reasonable amount of tested bits. Note that in contrast with an online test, a total failure test shall detected total failures of a noise source virtually immediately.

time-local stationarity (AIS 31-specific term that refers to the distribution of random numbers) A sequence of random variables $X_1, X_2, \ldots$ is considered to be 'time-local' stationarily distributed (often, loosely 'stationarily distributed' if the context is clear) if this sequence may be viewed as stationarily distributed at least over 'short' time-scales (in absolute time) that are yet 'large' compared to the sample size of the online tests and the evaluator tests.

total failure The noise source is broken and delivers no or at most a small fraction of its expected entropy.
Note 1: Depending on the concrete design and digitization, a total failure of the noise source may result in constant or short-period sequences of raw random numbers.
Note 2: It is possible that the raw random numbers still contain entropy due to noise from other components (e.g., an amplifier), but this scenario still constitutes a total failure..

total failure alarm Consequence of a failed total failure test.

total failure test A test that reliably detects total failures and prevents the output of low-entropy random numbers
Note: A total failure test is usually realized by physical measurements or by a statistical test. Due to the low entropy, a total failure can usually be detected very reliably, and the probability of a false positive is usually small.

true RNG A device or mechanism for which the output values depend on a noise source.

unbiased A random variable is said to be unbiased if all values of the finite sample space are chosen with the same probability. Contrast with biased.
Note: The terms unbiased and uniformly distributed are used synonymously.

uniformly distributed A random variable X with a finite range is considered to be uniformly distributed if X assumes each value with identical probability.
Note: The terms uniformly distributed and unbiased are used synonymously.

uninstantiation Uninstantiating an instance of a DRNG means that this instance no longer exists. In particular, the internal state and secret parameters are deleted.

widely recognized cryptographic primitive A cryptographic primitive is considered widely recognized if it has undergone diversified scientific review from many researchers and if the cryptographic community has no serious doubts concerning its strength in relevant operational circumstances.

with memory Property of a post-processing algorithm. It means that the post-processing is stateful, i.e., has a state that retains information from previous invocations or steps.

Acronyms

AES Advanced encryption standard.

ANSSI Agence nationale de la sécurité des systèmes d'information.

BSI Bundesamt für Sicherheit in der Informationstechnik.

das Digitized analog noise signal.

DRBG Deterministic random bit generator.

DRNG Deterministic RNG.

ECC Elliptic-curve cryptography.

ECDSA Elliptic Curve Digital Signature Algorithm.

iid Independent and identically distributed.

KAT Known-answer test.

LFSR Linear-feedback shift register.

NIST National Institute of Standards and Technology.

NPTRNG Non-physical true RNG.

OFB Output feedback.

PRNG Pseudorandom number generator.

PTRNG Physical true RNG.

RNG Random number generator.

RSA Rivest–Shamir–Adleman cryptosystem.

SHA Secure Hash Algorithm.

TRNG True RNG.

Abbreviations from Common Criteria

ADV Assurance Development.

AVA Assurance Vulnerability Analysis.

CC Common Criteria.

CEM Common Evaluation Methodology.

EAL Evaluation Assurance Level.

FCS Functional Class Cryptographic Support.

ITSEC Information Technology Security Evaluation Criteria.

ITSEM Information Technology Security Evaluation Manual.

PP Protection Profile.

SFR Security Functional Requirement.

ST Security Target.

TOE Target of Evaluation.

TSF TOE Security Functionality.

Symbols

$A \times B$ Cartesian product of the sets A and B.

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

$N(0, 1)$ Standard normal (Gaussian) distribution with mean 0 and variance 1.

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

P_λ Poisson distribution with parameter λ.

$X \| Y$ Concatenation of two strings X and Y. The strings X and Y are either both bit strings, or both byte strings.

$\Phi(\cdot)$ Cumulative distribution of the standard normal (Gaussian) distribution with mean 0 and variance 1; $\Phi(x) = \frac{1}{\sqrt{2pi}} \int_{-\infty}^{x} e^{-0.5t^2}\, dt$.

$\mathrm{Prob}(X = x)$ Probability that the random variable X assumes the value x.

$\mathrm{Prob}(x)$ Probability of the value x (short notation of $\mathrm{Prob}(X = x)$ if it is clear which random variable is concerned).

$\lceil x \rceil$ Ceiling: the smallest integer greater than or equal to x, $\lceil x \rceil = \min\{n \in N \mid x \leq n\}$.

$\lfloor x \rfloor$ Floor: the largest integer less than or equal to x, $\lfloor x \rfloor = \max\{n \in N \mid n \leq x\}$.

$|X|$ For a finite set X the notation $|X|$ denotes its cardinality. If X is a string $|X|$ denotes its length.

$\mathbb{N}$ Set of natural numbers, $= \{1, 2, \ldots\}$.

$\mathbb{N}_0$ Set of natural numbers with zero, $= \{0, 1, 2, \ldots\}$.

$\mathbb{Z}$ Set of integers.

$\mathbb{Z}_n$ $\{0, 1, \ldots, n-1\}$.

$\oplus$ Addition in $\mathrm{GF}(2)$, $0 \oplus 0 = 0$, $0 \oplus 1 = 1$, $1 \oplus 0 = 1$, $1 \oplus 1 = 1$.

$\pi_w(x)$ The projection of a vector $x = (x_1, x_2, \ldots, x_n)$ onto the coordinates $w = \left\{i_1, i_{2,}, \ldots, i_{|w|}\right\} \subseteq \{1, \ldots, n\}$. That is, $\pi_w(x) = \left(x_{i_1}, x_{i_2}, \ldots, x_{i_{|w|}}\right)$.

$g \circ f$ Composition of mappings f and g.

References

[AIS20] BSI. *Funktionalitätsklassen und Evaluationsmethodologie für deterministische Zufallszahlengeneratoren*. Version 3.0 (15.05.2013). https://www.bsi.bund.de/dok/6618284.

[AIS2031An_11] W. Killmann and W. Schindler. *A proposal for: Functionality classes for random number generators*. Version 2.0 (18.09.2011), Mathematical-technical reference of [AIS20] and [AIS31], https://www.bsi.bund.de/dok/ais-20-31-appx-2011.

[AIS20An_99] 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/dok/ais-20-appx-1999.

[AIS31] BSI. *Funktionalitätsklassen und Evaluationsmethodologie für physikalische Zufallszahlengeneratoren*. Version 3 (15.05.2020). https://www.bsi.bund.de/dok/6618252.

[AIS31An_01] W. Killmann and W. Schindler. *A proposal for: Functionality classes and evaluation methodology for true (physical) random number generators*. Version 3.1 (25.09.2001), English translation. https://www.bsi.bund.de/dok/ais-31-appx-2001.

[AIS46_ECC] BSI. *AIS 46 — Minimum Requirements for Evaluating Side-Channel Attack Resistance of Elliptic Curve Implementations*. Version 2.0 (16.11.2016).

[ASPB+18] E.N. Allini et al. "Evaluation and Monitoring of Free Running Oscillators Serving as Source of Randomness". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2018.3 (2018), pp. 214–242.

[BeFV10] F. Bernard, V. Fischer, and B. Valtchanov. "Mathematical model of physical RNGs based on coherent sampling". In: *Tatra Mountains Mathematical Publications* 45.1 (2010), pp. 1–14. URL: https://tatra.mat.savba.sk/Full/45/01be-f-v.pdf.

[BuLu08] M. Bucci and R. Luzzi. "Fully Digital Random Bit Generators for Cryptographic Applications, IEEE Transactions on Circuits and Systems I". In: *Regular Papers* 5 (3 2008), pp. 861–875.

[BuLu16] M. Bucci and R. Luzzi. "A Fully-Digital Chaos-Based Random Bit Generator". In: *The New Codebreakers: Essays Dedicated to David Kahn on the Occasion of His 85th Birthday*. Ed. by P.Y.A. Ryan, D. Naccache, and J.-J. Quisquater. Vol. 9100. Lecture Notes in Computer Science. Springer, 2016, pp. 396–414.

[CFPZ09] C. Chevalier et al. "Optimal Randomness Extraction from a Diffie-Hellman Element". In: *Proceedings of the 28th Annual International Conference on Advances in Cryptology: the Theory and Applications of Cryptographic Techniques*. EUROCRYPT09. Cologne, Germany: Springer-Verlag, 2009, pp. 572–589.

[Cox62] David Roxbee Cox. *Renewal theory*. London: Methuen, 1962.

[DaRo87] W.B. Davenport Jr and W.L. Root. "An Introduction to the Theory of Random Signals and Noise (Institute of Electrical and Electronics Engineers, New York, 1987)". In: *Author Affiliations Kazumasa Takada, Akira Himeno, Ken-ichi Yukimatsu NTT Communications Switching Laboratories, Nippon Telegraph and Telephone Corporation* (), pp. 9–11.

[DiBi07] M. Dichtl. "Bad and Good Ways of Post-processing Biased Physical Random Numbers". In: *Fast Software Encryption, 14th International Workshop, FSE 2007, Luxembourg, Luxembourg, March 26-28, 2007, Revised Selected Papers*. Ed. by A. Biryukov. Vol. 4593. Lecture Notes in Computer Science. Springer, 2007, pp. 137–152. DOI: 10.1007/978-3-540-74619-5_9. URL: https://doi.org/10.1007/978-3-540-74619-5_9.

[Fell65] W. Feller. *An Introduction to Probability Theory and Its Application (Vol. 2)*. Wiley, 1965.

[FiBB19] V. Fischer, F. Bernard, and N. Bochard. "Modern random number generator design - Case study on a secured PLL-based TRNG". In: *it – Information Technology* 61.1 (2019), pp. 3–13. DOI: 10.1515/itit-2018-0025.

[FiDr02] V. Fischer and M. Drutarovsky. "True Random Number Generator Embedded in Reconfigurable Hardware". In: *Cryptographic Hardware and Embedded Systems - CHES 2002*. Vol. 2523. LNCS. Redwood Shores, CA, USA. Springer Verlag, 2002, pp. 415–430.

[Flod89] P. Flajolet and A. Odlyzko. "Random mapping statistics". In: *Advances in cryptology – EUROCRYPT'89*. Springer. 1990, pp. 329–354.

[GaSt77] P. Gänssler and W. Stute. *Wahrscheinlichkeitstheorie*. Springer, 1977.

[Geor15] H.-G. Georgii. *Stochastik — Einführung in die Wahrscheinlichkeitstheorie und Statistik*. 5th. De Gruyter, 2015.

[Golo64] S.W. Golomb. "Random permutations". In: *Bull. Amer. Math. Soc* 70 (1964).

[HaLP34] G.H. Hardy, J.E. Littlewood, and G. Pòlya. *Inequalities*. 1934.

[HoRo48] W. Hoeffding and H. Robbins. "The central limit theorem for dependent random variables". In: *Duke Math. J.* 15 (1948), pp. 773–780.

[ISO_11770-3] *ISO / IEC 11770-3: Information security — Key management — Part 3: Mechanisms using asymmetric techniques*. 2021.

[ISO_18031] *ISO / IEC 18031: Information technology – Security Techniques. Random Bit Generation. 2011 / Cor 1: 2014 / A1: 2017.*

[ISO_20543] *ISO / IEC 20543: Information technology – Security Techniques. Test and Analysis Methods for Random Bit Generators within ISO / IEC 19790 and ISO / IEC 15408*. 2019.

[Jitter-RNG] Stephan Müller. *CPU Time Jitter Based Non-Physical True Random Number Generator*. (July 2022). https://www.chronox.de/jent/doc/CPU-Jitter-NPTRNG.pdf.

[Jone04] G.L. Jones. "On the Markov Chain Central Limit Theorem". In: *Probability Surveys* 1 (2004), pp. 299–320.

[KaTa75] S. Karlin and H.M. Taylor. *A First Course in Stochastic Processes.* 2nd edition. London: Academic Press, 1975.

[Kels23] J Kelsey. "Prediction, Backtracking, and Enhanced Backtracking Resistance in HMAC_ DRBG." In: *to be submitted* 2023 (2023).

[KiSc08] W. Killmann and W. Schindler. "A Design for a Physical RNG with Robust Entropy Estimators." In: *CHES 2008.* Ed. by E. Oswald and P. Rohatgi. Vol. 5154. Lecture Notes in Computer Science. Springer, 2008, pp. 146–163.

[KoSC78] V.F. Kolchin, B.A. Sevast'yanov, and V.P. Chistyakov. *Random Allocations.* V.H. Winston, 1978.

[Lach08] P. Lacharme. "Post-Processing Functions for a Biased Physical Random Number Generator". In: *Fast Software Encryption, 15th International Workshop, FSE 2008, Lausanne, Switzerland, February 10-13, 2008, Revised Selected Papers.* Ed. by K. Nyberg. Vol. 5086. Lecture Notes in Computer Science. Springer, 2008, pp. 334–342. DOI: 10.1007/978-3-540-71039-4_21. URL: https://doi.org/10.1007/978-3-540-71039-4_21.

[Lall86] S.P. Lalley. "Renewal Theorem for a Class of Stationary Sequences". In: *Probab. Th. Rel. Fields* 72 (1986), pp. 195–213.

[Linux_RNG_2016] *S. Müller and G. Krummeck and M. Romsey: Dokumentation und Analyse des Linux-Pseudozufallszahlengenerators. Version 5.5 (includes the Linux kernels 3.2.0 and 3.5 to 4.8), report, produced by atsec information security GmbH, by order of Bundesamts für Sicherheit in der Informationstechnik (BSI), last updated: November 2016.* https://www.bsi.bund.de/LinuxRNG.

[Linux_RNG_2020] *S. Müller: Documentation and Analysis of the Linux Random Number Generator. Version 3.6 (includes the Linux kernels 4.9 to 5.5), report, produced by atsec information security GmbH, by order of Bundesamts für Sicherheit in der Informationstechnik (BSI), last updated: July 2020.* www.bsi.bund.de/LinuxRNG.

[Linux_RNG_2022] *S. Müller: Documentation and Analysis of the Linux Random Number Generator. Version 4.11 (includes the Linux kernels 5.6 to 5.17), report, produced by atsec information security GmbH, by order of Bundesamts für Sicherheit in der Informationstechnik (BSI), last updated: April 2022.* www.bsi.bund.de/LinuxRNG.

[Linux_RNG_overview] *Overview of Linux kernels with NTG.1- or DRG.3-compliant random number generator /dev/random, Bundesamts für Sicherheit in der Informationstechnik (BSI), last updated: April 2022.* www.bsi.bund.de/LinuxRNG.

[Maur92] U. Maurer. "A universal statistical test for random bit generators". In: *J. Cryptol.* 5.2 (1992), pp. 89–105.

[Neue04] D. Neuenschwander. *Probabilistic and Statistical Methods in Cryptology, An Introduction by Selected Topics*. Vol. 3028. Lecture Notes in Computer Science. Springer, 2004.

[Plia99] J. Pliam. "The Disparity between Work and Entropy in Cryptology". In: (1998). Appeared in the THEORY OF CRYPTOGRAPHY LIBRARY and has been included in the ePrint Archive. pliam\spacefactor\@m{}ima.umn.edu. 10500 received November 9th, 1998. Revised, February 1st, 1999. https://eprint.iacr.org/1998/024.

[PuWi68] Purdom, P.W. and Williams, J.H. "Cycle Length in a Random Function". In: *Transactions of the American Mathematical Society* 2.133 (547-551), pp. 88–100.

[RNG_virtual_env] *S. Müller and G. Krummeck and H. Kurth: Analysis of random number generation in virtual environments. Report, produced by atsec information security GmbH, by order of Bundesamts für Sicherheit in der Informationstechnik (BSI), Version 1.0, October 21, 2016.* https://www.bsi.bund.de/dok/randomness-in-vms.

[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, 2010 revisions dated May 15, 2001).*

[SP800-90A] *Recommendation for Random Number Generation Using Deterministic Random Bit Generators, NIST Special Publication 800-90A (June 2015).* https://doi.org/10.6028/NIST.SP.800-90Ar1.

[SP800-90B] *Recommendation for the Entropy Sources Used for Random Bit Generation, NIST Special Publication 800-90A (January 2018).* https://doi.org/10.6028/NIST.SP.800-90B.

[SP800-90C] *Recommendation for Random Bit Generator (RBG) Constructions, NIST Special Publication 800-90C (Third Draft) (September 2022).* https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90C.3pd.pdf.

[Schi01] W. Schindler. "Efficient Online Tests for True Random Number Generators." In: *CHES 2001*. Ed. by Ç.K. Koç, D. Naccache, and C. Paar. Vol. 2162. Lecture Notes in Computer Science. Springer, 2001, pp. 103–117.

[Schi09a] W. Schindler. "Random Number Generators for Cryptographic Applications". In: *Cryptographic Engineering*. Ed. by Ç.K. Koç. Springer, 2009, pp. 5–23.

[Schi09b] W. Schindler. "Evaluation Criteria for Physical Random Number Generators". In: *Cryptographic Engineering*. Ed. by ¸C.K. Ko¸c. Springer, 2009, pp. 25–54.

[Shev11] I. Shevtsova. "On the absolute constants in the Berry Esseen type inequalities for identically distributed summands". In: *arXiv:1111.6554* (2011).

[TR-02102] *BSI, TR-02102, Technical Guideline Cryptographic Mechanisms, (current version, courtesy translation),* *https://bsi.bund.de/TR-02102*.