NIST Internal Report
NIST IR 8446

Bridging the Gap Between Standards on Random Number Generation

Comparison of SP 800-90 Series and AIS 20/31

Elaine Barker
John Kelsey
Kerry McKay
Johannes Mittmann
Matthias Peter
Werner Schindler
Meltem Sönmez Turan

This publication is available free of charge from:
https://doi.org/10.6028/NIST.IR.8446

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
U.S. DEPARTMENT OF COMMERCE

NIST Internal Report
NIST IR 8446

Bridging the Gap Between Standards on Random Number Generation

Comparison of SP 800-90 Series and AIS 20/31

Elaine Barker [1], John Kelsey [1], Kerry McKay [1], Johannes Mittmann [2], Matthias Peter [2], Werner Schindler [2], Meltem Sönmez Turan [1]

[1] Computer Security Division, Information Technology Laboratories, NIST, Gaithersburg, Maryland, USA
[2] Bundesamt für Sicherheit in der Informationstechnik (BSI), Germany

This publication is available free of charge from:
https://doi.org/10.6028/NIST.IR.8446

January 2026

U.S. Department of Commerce
Howard Lutnick, Secretary

National Institute of Standards and Technology
Craig Burkhardt, Acting Under Secretary of Commerce for Standards and Technology and Acting NIST Director

NIST Technical Series Publications: https://www.nist.gov/nist-research-library/nist-publications

Reports on Computer Systems Technology

The Information Technology Laboratory (ITL) at the National Institute of Standards and Technology (NIST) promotes the U.S. economy and public welfare by providing technical leadership for the Nation's measurement and standards infrastructure. ITL develops tests, test methods, reference data, proof of concept implementations, and technical analyses to advance the development and productive use of information technology. ITL's responsibilities include the development of management, administrative, technical, and physical standards and guidelines for the cost-effective security and privacy of other than national security-related information in federal information systems.

References to Other Publications

There may be references in this publication to other publications currently under development by NIST in accordance with its assigned statutory responsibilities. The information in this publication, including concepts and methodologies, may be used by federal agencies even before the completion of such companion publications. Thus, until each publication is completed, current requirements, guidelines, and procedures, where they exist, remain operative. For planning and transition purposes, federal agencies may wish to closely follow the development of these new publications by NIST.

Organizations are encouraged to review all draft publications during public comment periods and provide feedback to NIST. Many NIST cybersecurity publications, other than the ones noted above, are available at https://csrc.nist.gov/publications

Non-Endorsement Disclaimer

Certain equipment, instruments, software, or materials, commercial or non-commercial, are identified in this paper in order to specify the experimental procedure adequately. Such identification does not imply recommendation or endorsement of any product or service by NIST, nor does it imply that the materials or equipment identified are necessarily the best available for the purpose.

Publication History

Approved by the NIST Editorial Review Board on February 4, 2025.

Suggested Citation

Elaine Barker, John Kelsey, Kerry McKay, Johannes Mittmann, Matthias Peter, Werner Schindler, Meltem Sönmez Turan (2026). Bridging the Gap Between Standards on Random Number Generation: Comparison of SP 800-90 Series and AIS 20/31. (National Institute of Standards and Technology, Gaithersburg, MD) NIST Internal Report (IR) 8446. DOI:10.6028/NIST.IR.8446

Author Names and ORCID Identifiers

Elaine Barker (0000-0003-0454-0461); John Kelsey (0000-0002-3427-1744); Kerry McKay (0000-0002-5956-587X); Johannes Mittmann (0000-0002-9307-1494); Matthias Peter (0000-0003-1080-3432); Werner Schindler (0000-0002-3073-0106); Meltem Sönmez Turan (0000-0002-1950-713).

Additional Information
Additional information about this publication is available at https://csrc.nist.gov/pubs/ir/8446/final, including related content, potential updates, and document history.

Contact Information: rbg_comments@nist.gov

All comments are subject to release under the Freedom of Information Act (FOIA)

Abstract

This report studies the cryptographic random number generation standards and guidelines written by BSI and NIST, namely AIS 20/31 and the SP 800-90 series. The aim of this report is to compare these publications, focusing on the similarities and differences of their terminology, assumptions, and requirements. This report also aims to improve the communications between all involved parties, promote a shared understanding, and reduce and resolve inconsistencies in related standards.

Keywords

cryptographic random number generation; entropy; terminology; validation.

Acknowledgments

The authors thank Hamilton Silberg (NIST), Jonas Fiege (BSI), and the members of the Cryptographic Module User Forum (CMUF) for fruitful discussions. The authors also wish to thank all individuals who submitted public comments for their valuable input.

Table of Contents

List of Figures

List of Tables

1. Introduction

The security of cryptographic mechanisms and protocols relies on the availability of high-quality random numbers (e.g., to generate cryptographic keys, initialization vectors, nonces, salts, and masking values). The generation of these random numbers and validating their quality are challenging tasks. There are multiple standards to provide guidelines on generating random numbers to be used in cryptography [1–11]. These standards may differ in their assumptions, requirements, and even the definitions that they use.

This report provides a study of the standards developed by the Bundesamt für Sicherheit in der Informationstechnik (BSI) and the National Institute of Standards and Technology (NIST) on random number generation. The terms random number generator (RNG) and random bit generator (RBG) are used interchangeably in this document.

The relevant standards developed by BSI are:

- **AIS 20**, *Functionality Classes and Evaluation Methodology for Deterministic Random Number Generators* [4]
- **AIS 31**, *Functionality Classes and Evaluation for Physical Random Number Generators* [5]
- **Mathematical-technical reference to AIS 20 and AIS 31** *A Proposal for Functionality Classes for Random Number Generators* [6] (previous version: [12])

The relevant standards developed by NIST are:

- **SP 800-90A**, *Recommendation for Random Number Generation Using Deterministic Random Bit Generators* [1]
- **SP 800-90B**, *Recommendation for the Entropy Sources Used for Random Bit Generation* [2]
- **SP 800-90C**, *Recommendation for Random Bit Generator Constructions* [3]

1.1. Random Number Generation Standards Developed by the BSI

AIS 20 and AIS 31 were developed by BSI and refer to a joint mathematical-technical reference. This NIST Interagency Report (IR) 8446 considers version 3.0 of the mathematical-technical reference [6]. AIS 20 specifies how deterministic RNGs will be evaluated in the German Common Criteria (CC) scheme, and it outlines an evaluation methodology for deterministic RNGs. Functionality classes with class-specific requirements are defined for different types of deterministic RNGs. AIS 31 specifies how physical RNGs are to be evaluated

in the German CC scheme, and it outlines an evaluation methodology for physical true RNGs (or simply, "physical RNGs"). Functionality classes with class-specific requirements are defined for different types of physical RNGs.

The *mathematical-technical reference to AIS 20 and AIS 31* [6] is intended for developers, evaluators, and certifiers. It specifies functionality classes for deterministic RNGs, physical true RNGs, and non-physical true RNGs. Furthermore, mathematical background is provided, and many examples are discussed in detail to explain the requirements of the functionality classes and the tasks of developers and evaluators. The mathematical-technical reference is often loosely referenced as AIS 20, AIS 31, or AIS 20/31, depending on the context.

A certification process according to the CC is carried out as follows:

- The applicant (usually the developer) delivers the following to the accredited evaluation lab: prototypes of the RNG (the RNG is usually a component of a larger target of evaluation), documentation and a description of the RNG, and evidence that the requirements of the claimed functionality class are fulfilled.
- The accredited lab evaluates the RNG according to AIS 20 or AIS 31 (and with regard to further criteria, such as implementation security), examines the documentation, and writes an evaluation report.
- The certification authority (e.g., BSI) checks the evaluation report and may demand additional evidence. If the certification authority is convinced that the (positive) evaluation result (of the RNG and the other evaluation aspects) is justified, a certificate is issued.

1.2. Random Number Generation Standards Developed by NIST

SP 800-90A, *Recommendation for Random Number Generation Using Deterministic Random Bit Generators* [1], specifies mechanisms for the generation of random bits using deterministic methods based on hash functions and block ciphers.

SP 800-90B, *Recommendation for the Entropy Sources Used for Random Bit Generation* [2], specifies the design principles and requirements for the entropy sources used by random bit generators and tests for the validation of entropy sources.

SP 800-90C, *Recommendation for Random Bit Generator Constructions*, specifies constructions for the implementation of random bit generators that include deterministic random bit generator mechanisms, as specified in SP 800-90A, and that use entropy sources, as specified in SP 800-90B.

The SP 800 90 series provides a basis for validation by the Cryptographic Algorithm Validation Program (CAVP) and Cryptographic Module Validation Program (CMVP) conducted by NIST and the Canadian Centre for CyberSecurity (CCCS) A submitting entity (e.g., a vendor) works with an accredited lab to submit their implementation for testing.

- For all SP 800-90A Deterministic Random Bit Generators (DRBGs), including their underlying cryptographic primitives (e.g., HMAC, hash functions, AES), and the vetted conditioning components, vector sets are generated by the testing laboratory, and the lab downloads them for provision to the submitter. The submitter generates the responses and sends them to the lab. The lab uploads the received responses to the CAVP and, upon successful validation, requests certification.
- For SP 800-90B entropy sources, the submitter generates data files for each defined operating environment. The files include at least one million samples, as required by SP 800-90B, along with the necessary restart data samples. The testing laboratory runs an entropy assessment tool and prepares an Entropy Assessment Report and Public Use Document that outline conformance to SP 800-90B and how the entropy source can be used within a cryptographic module. The entropy source documentation is then reviewed by the CMVP. Once any review concerns are resolved, the entropy source can be certified.
- For SP 800-90C RBG constructions, the testing laboratory prepares a Random Bit Generator Report that outlines conformance to SP 800-90C. The document is reviewed by the CMVP. Once any review concerns are resolved, the random bit generator can be certified.

1.3. Aim and Organization

The aim of this report is to compare standards and guidelines on cryptographic random number generation by focusing on the similarities and differences in their terminology, assumptions, and requirements. This report also aims to improve communications between all involved parties, promote a shared understanding, and reduce and resolve inconsistencies in related standards. In addition, this report is intended to assist in the validation and certification of a random number generator implementation in both the BSI and NIST validation/certification programs.

Section 2 discusses general validation and certification requirements related to conditioning, post-processing, computational security, entropy estimation, and health tests. Section 3 examines the similarities and differences between the functionality classes in AIS 20/31

and the constructions specified in SP 800-90C. Section 4 provides a comparison of the terminology used in the BSI and NIST standards.

2. General Validation/Certification Requirements

This section compares the BSI and NIST standards based on their various requirements, including conditioning, post-processing, computational security requirements, entropy estimation, and health tests.

2.1. Conditioning and Post-Processing

The outputs of noise sources may have statistical biases and can include dependencies. The terms *conditioning* and *post-processing* are used interchangeably to denote the process of reducing such bias and improving the statistical quality of these outputs. While these deterministic functions do not increase the amount of entropy, the entropy rate of the entropy source output (i.e., the amount of entropy per bit) can increase if the conditioning component compresses the output of the noise source.

Some examples of simple conditioning components are the *Von Neuman's method* [13], *Samuelson's method* [14], and *Peres' method* [15]. Conditioning components can also be cryptographic functions, such as the HMAC construction [16] using SHA-256. In addition to improving the statistical quality of the outputs, cryptographic post-processing/conditioning algorithms can provide additional security assurances when the underlying randomness source fails.

Non-cryptographic conditioning components should be selected based on the stochastic model of the noise or entropy source outputs. However, the stochastic model usually models the raw random numbers from the noise source, ideally providing a set of probability distributions that contains the true distribution of the raw random numbers. At a minimum, the stochastic model considers particular aspects that provide entropy. This suffices if it can be shown that the neglected effects (e.g., of any non-credited entropy contributions) do not lower the entropy.

On the basis of the stochastic model and depending on the post-processing algorithm used, a lower entropy bound for the output values is determined. In principle, a stochastic model can also provide distributions of the output values. However, unless a very simple algorithmic post-processing algorithm is applied (e.g., XORing non-overlapping pairs of binomially distributed raw random numbers), it is often infeasible to specify the distributions of the output bits after post-processing/conditioning.

In AIS 20/31, stochastic models are only required for physical noise sources. If the post-processing algorithm/conditioning component uses cryptographic algorithms, the guaranteed entropy bound of the random bits before conditioning is the most relevant.

2.2. Computational Security Requirements

The desired properties of random numbers used for cryptographic applications are that they be unpredictable, unbiased, and independent. Without knowing the output of the randomness source, the output of the random number generators should be difficult to distinguish from an ideal random sequence for an adversary with limited computational power. A practical RBG/RNG aims to approximate an ideal randomness source or ideal RNG.

The security of random number generators depends on the unpredictability of the noise source output (i.e., information-theoretic security quantified in entropy) and on algorithmic properties (i.e., computational security quantified as security strength). To achieve computational security, high-entropy seed material from a true RNG or from an entropy source is required, at least initially. For higher assurance, a periodic or continuous influx of high-entropy input may be desirable.
AIS 20/31 distinguishes between "additional input" and "high-entropy additional input." Additional input can be included in each request. The entropy of this data can be large, small, or even zero. In contrast, the term high-entropy additional input data contain enough entropy to ensure *enhanced forward secrecy*.

BSI and NIST documents require assurance that an adversary (i.e., an attacker) cannot use knowledge of recent values to determine earlier outputs. An RNG that conforms to AIS 20 provides *backward secrecy* and may provide *enhanced backward secrecy*, while an RBG that conforms to SP 800-90 has been specified to provide *backtracking resistance*. Backtracking resistance is a stronger security claim than enhanced backward secrecy. However, they are conceptually similar and can usually be considered equivalent in practice. The relationship between backward security requirements is depicted in Fig. 1.

Similarly, both sets of documents include requirements that are intended to limit an adversary's ability to determine future outputs. An RNG that conforms to AIS 20 provides *forward secrecy* and may provide a capability for *enhanced forward secrecy*, while an RBG that conforms to SP 800-90 has been specified to provide a capability for forward secrecy and may provide a capability for *prediction resistance*. Enhanced forward secrecy and prediction resistance are conceptually similar and are achieved by the insertion of sufficient amounts of fresh entropy. The relationship between forward security requirements is depicted in Fig. 2.

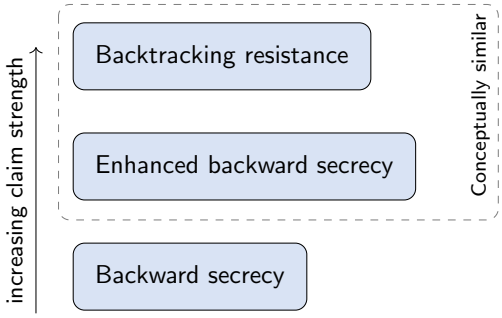


Fig. 1. Relationship of backward security claims

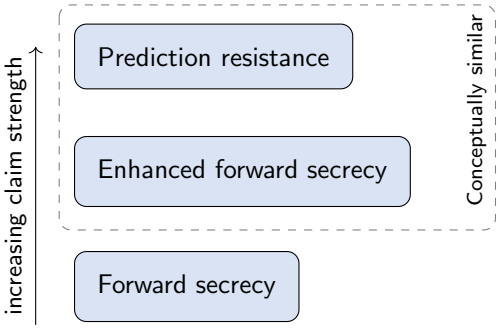


Fig. 2. Relationship of forward security claims

2.3. Entropy Estimation

A central goal of the security evaluation of a true RNG/RBG is the verification of a lower entropy bound for the random numbers that are output. For deterministic RNGs/RBGs, the notion of entropy is needed to quantify the randomness of seed material or high-entropy additional input.

Entropy estimation is a challenging problem because the distribution of the output values (i.e., the distribution of the underlying random variables) is a priori unknown. Trustworthy entropy estimation requires knowledge of the underlying nondeterministic process being used by the noise source, and black box statistical methods can only serve as a sanity check on that kind of estimate.

Two commonly used measures of entropy are *Shannon entropy* and *min-entropy*. To be more precise, *Shannon entropy* and *min-entropy* are the most important representatives of Rényi entropy. If a discrete random variable X takes outcomes $x_i \in \mathscr{A}$, each with probability $\Pr(X = x_j) = p_j$, then

$$H(X) = -\sum_j p_j \log_2(p_j) \qquad \text{(Shannon entropy)},$$

$$H_\infty(X) = \min_j(-\log_2(p_j)) \qquad \text{(min-entropy)}.$$

A value x that is assumed by a random variable is called a realization of X. The min-entropy is determined by the largest probability with which a value is assumed by X. This value is the most promising single guess. While min-entropy considers the worst case, Shannon entropy aims at the average case.

In the context of true random number or bit generators, sequences $x_1, x_2, \ldots$ of random numbers are considered, with each element interpreted as a realization of the corresponding random variable $X_1, X_2, \ldots$. Unless these random variables are independent and identically distributed (iid), the central task in evaluating a generator is to derive a trustworthy lower entropy bound per output bit. For physical RNGs, this requires formulating, justifying, and analyzing the stochastic model with a focus on entropy. The technical reference [6] contains several examples that deal with entropy calculation. Both the SP 800-90 series and AIS 20/31 use min-entropy. However, for all but one functionality class, AIS 20/31 alternatively allows the use of Shannon entropy under suitable conditions.

2.4. Health Tests

Health tests must be performed on the nondeterministic components of a randomness source and any deterministic components within the RBG/RNG, including within an SP 800-90B entropy source and AIS 20/31 physical true RNGs (PTRNGs) and non-physical true RNGs (NPTRNGs). AIS 20/31 uses the terms “start-up tests,” “online tests,” and “total failure tests” instead of “health tests.”

For nondeterministic components (i.e., noise sources), the process of extracting randomness from nondeterministic events is fragile and can fail due to various factors, such as environmental conditions, manufacturing tolerances and defects, or aging. Health tests on the noise sources are used to avoid these failures and check that the generator continues to perform as desired. The generator indicates an error condition (e.g., triggers a noise alarm) when abnormal behavior is detected. Health tests are typically designed as statistical tests, and the number of false positive alarms are limited by type I error probability.

For deterministic components, known answer tests are typically conducted at RBG/RNG start-up to ensure the correct operation of a device, algorithm, or function before its first use. They may also be performed on demand.

2.4.1. SP 800-90B Health Tests on Noise Sources

SP 800-90B requires three types of tests on the noise source output within an entropy source (after digitization but before any conditioning is performed):

- *A start-up test* is performed every time the entropy source is initialized or powered up. This test is carried out on the noise source output before any output is released from the entropy source.

- *Continuous tests* are performed within an entropy source on the output of its noise source in order to gain some level of assurance that the noise source is working correctly prior to producing each output from the entropy source.
- *On-demand tests* are a type of health test that is available to be run whenever a user or a relying component requests it.

SP 800-90B defines two continuous health tests that target generic failure conditions: the *repetition count test* and the *adaptive proportion test*. Developers may define their own tests that detect the same failures, additional failures, or both.

2.4.2. AIS 20/31 Health Tests on Noise Sources

For PTRNGs, AIS 31 requires three types of tests:

- A *start-up test* is performed after the RNG has been started. Its task is to detect a total failure of the noise source and severe statistical weaknesses. No random numbers are output before the start-up test has been successfully completed.
- A *total failure test* detects the occurrence of a total failure of the noise source during PTRNG operation. It prevents the output of random numbers that have small or even no entropy due to the total failure of the noise source.
- An *online test* checks the quality of the raw random numbers produced by the noise source while the RNG is in operation. The online test is intended to quickly detect non-tolerable entropy defects of the raw random numbers.

There are no approved tests specified, but the applicant for a certificate (usually the developer) has to provide evidence that the selected tests perform their tasks. Usually, start-up tests and online tests are realized by statistical tests or by a test procedure that applies several statistical tests, and total failure tests typically apply statistical tests or physical measurements. In particular, for physical RNGs, the total failure test should be based on a sound failure analysis of the physical noise source, and the online test has to be tailored to the stochastic model of the noise source.

For NPTRNGs, the raw random numbers are also tested. The aims are similar to those for physical RNGs but, in particular, the requirements on the online test are lower since no stochastic model for the noise source is required. For NPTRNGs, AIS 20/31 simply speaks of “testing” but does not apply the same terms as are used for physical RNGs. Testing is needed to confirm the entropy claim. The results of testing may affect the (heuristic) entropy counter.

3. Functionality Classes vs. RBG Constructions

Section 3 provides information to support developers in constructing designs that can be successfully validated under both the BSI and NIST/CSE programs. The functionality classes from AIS 20/31 and the constructions from SP 800-90C are briefly explained in Sec. 3.1 and 3.2. Section 3.3 discusses the similarities and differences between corresponding functionality classes and constructions.

3.1. Functionality Classes of BSI

Rather than specifying approved RNG designs, AIS 20/31 defines seven functionality classes with security requirements that an RNG has to fulfill in order to comply with a targeted functionality class. This section explains the main features of the functionality classes, and application notes in AIS 20/31 illustrate these security requirements.

The functionality classes are listed below in the same order as they appear in AIS 20/31. Functionality classes DRG.2, DRG.3, DRG.4, and DRT.1 define requirements for deterministic RNGs (DRNGs). The objects of functionality class DRT.1 are whole DRNG trees rather than individual DRNGs. Functionality classes PTG.2 and PTG.3 are physical true RNGs (PTRNGs), and functionality class NTG.1 describes non-physical true RNGs (NPTRNGs).

Figure 3 shows the functionality classes in AIS 20/31 and the relationship between them. Several arrows point from one functionality class to another (e.g., from DRG.2 to DRG.3), whereby the functionality class to which the arrow points has stronger requirements. Figure 3 illustrates the hierarchical order. Not all functionality classes are comparable, but PTG.3 is the strongest functionality class. The arrow from DRT.1 to DRG.3 is dotted, and functionality class DRT.1 is placed slightly below DRG.3; cf. the explanations to functionality class DRT.1 below.

The functionality classes DRG.3, DRG.4, DRT.1, PTG.2, PTG.3, and NTG.1 have equivalents in the SP 800-90C RBG constructions. Similarities and differences between the functionality classes and the RBG constructions are explained in Sec. 3.3.

DRG.2. Functionality class DRG.2 ensures backward secrecy and forward secrecy. Since AIS 20/31 does not specify approved designs, a security proof is required.

The *effective internal state* must have at least 246 bits. The min-entropy requirement of the effective internal state for seeding and reseeding is at least 240 bits. In the case of reseeding, the requirement for at least 240 bits of fresh entropy is intended to prevent an adversary who knows the previous internal state from predicting future outputs after sufficient fresh entropy is inserted into the DRNG.

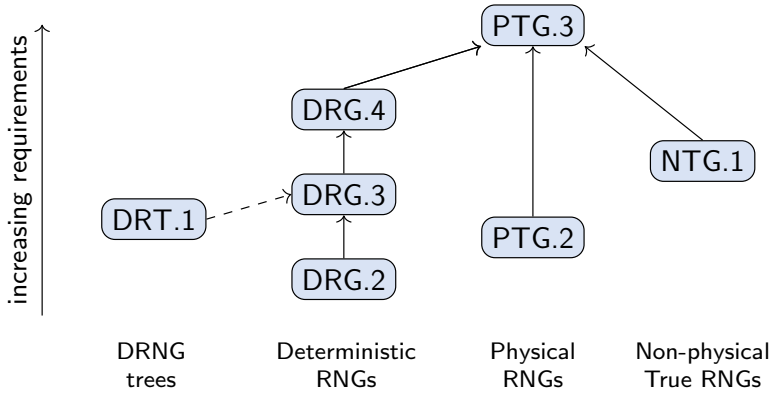


Fig. 3. AIS 20/31 functionality classes

Under suitable conditions, namely when a PTG.2- or PTG.3-compliant PTRNG (or more generally, a PTRNG generating time-local stationarily distributed raw random numbers) is used as the randomness source, an alternative entropy condition (i.e., ≥ 250 bits of Shannon entropy for the effective internal state) can be applied. Seeding and reseeding use a true RNG (with a physical or non-physical noise source) as the randomness source or, alternatively, a DRG.4 compliant DRNG. In the latter case, enhanced forward secrecy must be ensured by the insertion of sufficient fresh entropy into the DRG.4-compliant DRNG before generating the seed material for the DRG.2 implementation.

Usually, a true RNG belongs to the functionality class PTG.2, PTG.3, or NTG.1. This is the easiest way for the applicant to meet the evaluation requirements, as it saves providing proofs for the entropy of its output data (i.e., proofs about the entropy of the seed material). In principle, true RNGs that are not compliant with one of these classes are also possible but require additional security proofs.

DRG.3. Compared with DRG.2, functionality class DRG.3 additionally ensures enhanced backward secrecy. This requires an additional security proof. The requirements on the size and on the min-entropy for the effective internal state are identical to class DRG.2.

Usually, the seeding and reseeding of functionality class DRG.3 use a true RNG (with a physical or non-physical noise source) as the randomness source. Like the DRG.2 functionality class, the randomness source is (but is not limited to) functionality classes PTG.2, PTG.3, NTG.1, and DRG.4 (under the condition that fresh entropy is inserted into the DRG.4-compliant DRNG before it generates the seed material).

DRG.4. Compared with DRG.3, functionality class DRG.4 additionally ensures the capability of providing enhanced forward secrecy and requires an additional security proof. Enhanced forward secrecy can be provided by reseeding or by the insertion of high-entropy additional

input. In both cases, the effective internal state shall have at least 240 bits of min-entropy (relative to an adversary who knows the previous internal state). Under suitable conditions, at least 250 bits of Shannon entropy is also possible. This prevents an adversary who knows the previous internal state from determining the next outputs (i.e., internal random numbers) with practical computational effort.

The requirements on the size (at least 246 bits) and on the min-entropy requirement (at least 240 bits) for the effective internal state are identical to functionality classes DRG.2 and DRG.3. The min-entropy requirement also applies to the insertion of high-entropy additional input.

The DRNG must be able to trigger reseeding or acquire high-entropy additional input to provide enhanced forward secrecy. This may be done on demand, in response to some condition, or after a certain time span has elapsed (non-exclusive options). In any case, enhanced forward secrecy always has to be ensured by the addition of sufficient fresh entropy each time that 2^{17} internal random number bits have been generated, although an ongoing generate request need not be interrupted.

Seeding, reseeding, and the insertion of high-entropy additional input require the use of a suitable physical RNG. In contrast to functionality classes DRG.2 and DRG.3, the use of non-physical true RNGs are not allowed for functionality class DRG.4. In particular, the randomness source for functionality class DRG.4 is (but is not limited to) a PTG.2- or PTG.3-compliant physical RNG. Again, it is possible to use a physical RNG that is not compliant with PTG.2 or PTG.3, but this requires additional security proofs.

DRT.1. In software implementations, there is often a need to seed and reseeed DRNGs by other DRNGs because no true RNG is available. The DRT.1 functionality class defines requirements for deterministic RNG trees or "DRNG trees." For this functionality class, it is permitted to seed a DRNG (except the root DRNG) by another DRNG under certain conditions. Unlike for the other functionality classes, the objects of functionality class DRT.1 are not individual DRNGs but the whole DRNG tree. This is due to the fact that in a DRNG tree, security cannot be guaranteed locally by evaluating a particular DRNG and the DRNG serving as its randomness source. DRNG trees can be static or dynamic. The latter means that new DRNGs can be instantiated and that existing DRNGs can be uninstantiated during the lifetime of the DRNG tree.

Using the left drawing in Fig. 4 as an example, DRNG A is the root of the tree. DRNG B is a direct seed successor of DRNG A if DRNG B has been seeded by DRNG A. DRNG A is the direct seed predecessor of DRNG B. This definition refers to the seeding procedure when DRNG B was instantiated but not necessarily to later reseeding procedures. A crucial security goal is

to prevent seed loops. To form a DRT.1-compliant DRNG tree, several conditions must be met:

- All nodes of the DRNG tree must fulfill the algorithmic requirements of functionality class DRG.3.
- The root DRNG can be seeded by a PTRNG, an NPTRNG, or a DRNG that provides enhanced forward secrecy immediately before the seed material is generated. In particular, this comprises (but is not limited to) the functionality classes PTG.2, PTG.3, NTG.1, and DRG.4 (under the condition that fresh entropy is inserted into the DRG.4-compliant DRNG before it generates the seed material).
- ('sibling rule') Apart from the root, each DRNG in the DRNG tree can only be reseeded by its direct seed predecessor or by a sibling of its direct seed predecessor. In the left drawing of Fig. 4, the direct seed predecessor of DRNG C is DRNG B. DRNG F is a sibling of DRNG B because they have the same direct seed predecessor (i.e., DRNG A). Therefore, DRNG F can be used to reseed DRNG C. Even if a sibling DRNG of its direct seed predecessor generates the seed material for reseeding DRNG C, the role of DRNG B as the direct seed predecessor of DRNG C remains unchanged, even if it has already been uninstantiated.
- The RNG that generates seed material for the root DRNG (i.e., the initial randomness source) as well as all DRNGs of the DRNG tree shall be implemented and operated inside the same security boundary. These requirements are intended to limit the potential security risks of the DRNG tree. The DRNG tree shall not be distributed over multiple computing platforms that belong to different operators with different (incompatible) security boundaries, and the seed material for the DRNGs shall not be generated outside of the security boundary.

Figure 4 shows two examples of DRNG trees. The relevant seeding and reseeding information of a DRNG tree can be stored and visualized as a *seed graph* — a colored, directed graph. The DRNGs (physical objects) in the DRNG tree are identified as nodes in the seed graph. When a new DRNG in the DRNG tree is instantiated, a node is added to the seed graph. Furthermore, a directed edge (e.g., directed line, directed link, arrow) is drawn from the direct seed predecessor of the instantiated DRNG to the instantiated DRNG in black. When a DRNG of the DRNG tree is uninstantiated, the corresponding node remains in the seed graph but is grayed out. However, the edges from and to this node remain in black in the seed graph.

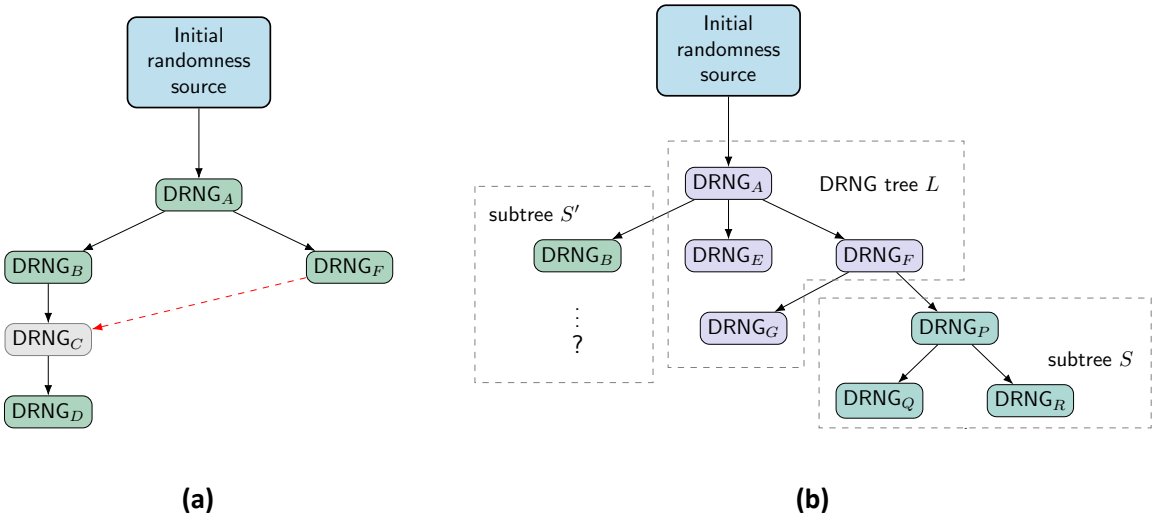


Fig. 4. Examples of seed graphs used by DRT.1-compliant DRNG trees. (a) DRNG$_C$ was reseeded by DRNG$_F$, the sibling of its direct seed predecessor DRNG$_B$, before it was uninstantiated. (b) Subtree S is attached to DRNG tree L, which was previously shown to comply with DRT.1 (e.g., by previous certification).

Finally, suppose that a DRNG generates seed material for another DRNG in the DRNG tree that is not its direct successor, as shown for DRNG$_F$ in Fig. 4a. In this case, a red edge is drawn from the DRNG that generates the seed material to the DRNG that receives the seed material unless such an edge (black or red) already exists. As shown in Fig. 4a, DRNG$_C$ has been uninstantiated. However, before being uninstantiated, DRNG$_C$ received seed material for reseeding from DRNG$_F$, which is a sibling of its direct seed predecessor, DRNG$_B$.

It is not necessary to explicitly maintain a seed graph during the lifetime of a DRNG tree. Nevertheless, a seed graph can be a useful tool for an evaluation to illustrate the seeding and reseeding information of a DRNG tree, especially when compositions of DRNG trees with DRNG subtrees are concerned. During the lifetime of the DRNG tree, it suffices to keep "local information," namely knowledge of the direct seed predecessor of a DRNG and the siblings of its direct seed predecessor.

In Fig. 4b, a subtree S (consisting of DRNG$_P$, DRNG$_Q$, and DRNG$_R$) is to be attached. It is already known (e.g., by a previous certification process) that DRNG$_A$, DRNG$_E$, DRNG$_F$, and DRNG$_G$ form a DRNG tree L that complies with functionality class DRT.1. Furthermore, a subtree S' that consists of DRNG$_B$ and its seed successors has been attached to DRNG tree L (or could be attached in the future). Little is known about the structure of subtree S', apart from the fact that no DRNG of S' generates seed material for any DRNG in tree L or subtree S. Subtree S' may violate the reseeding rule formulated above, but even if

this is the case, S' does not affect DRNG tree L or the composition of L with subtree S. In particular, the composition L with S satisfies the reseeding rule.

Algorithmically, the security of DRNGs in DRNG trees should be similar to that of DRG.3-compliant DRNGs. However, more non-algorithmic security threats exist for DRNG trees. This is because in DRNG trees, security cannot be verified locally by evaluating the individual DRNGs and their direct seed predecessors. For these reasons, in Fig. 3, functionality class DRT.1 is placed a little below functionality class DRG.3, and the arrow is dotted.

PTG.2. The PTG.2 functionality class includes a physical noise source (including a digitization mechanism) that produces raw random numbers. Raw random numbers are discrete values (e.g., bits, bit strings, integers) that are obtained by the digitization mechanism. The raw random numbers can be interpreted as realizations of random variables that are time-locally stationary distributed (i.e., time-local stationarity).

Principally, post-processing is optional but may be necessary for concrete designs to satisfy the PTG.2-specific entropy requirements. The applicant for a certificate can select between three entropy claims in which the entropy per output bit (i.e., internal random number bit) is:

(a) ≥ 0.9998 Shannon entropy,

(b) ≥ 0.98 min-entropy, or

(c) ≥ 0.9998 Shannon entropy *and* ≥ 0.98 min-entropy.

The entropy boundaries for class PTG.2 (0.9998 and 0.98) are fixed PTG.2-specific values that cannot be adjusted to the PTRNG. They are not far from the NIST definition of full entropy, which corresponds to min-entropy $\geq 1 - 2^{-32}$; cf. functionality class PTG.3.

An online test, total failure test, and start-up test are mandatory. A stochastic model of the noise source is the central part of the evaluation, and the effectiveness of the online test and the total failure test has to be verified. 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. The verification of the claimed stochastic model usually can be supported by statistical tests and predictors that are tailored to this stochastic model.

Furthermore, the evaluator has to apply a specified black box test suite T_{irn} on the internal random numbers (i.e., on the random numbers after post-processing). The test suite T_{irn} includes four statistical tests, two of which use predictors. While applying black box tests and black box predictors cannot be used to verify any entropy claim, they can possibly

falsify it. Since the class requirements allow developers to choose the physical noise source, a black box test suite for the raw random numbers is not specified.

Although the entropy per output bit is large, the output bits can be biased and dependent. Therefore, a PTG.2-compliant PTRNG should not be directly used to generate sensitive data (e.g., keys, signature parameters, nonces). Instead, PTG.2-compliant PTRNGs should be used and are appropriate to seed and reseed DRNGs, provide high-entropy additional input, and serve as the core of a PTG.3 implementation.

PTG.3. The PTG.3 functionality class is the strongest class in AIS 20/31. It defines hybrid physical true RNGs by combining a physical noise source (including a digitization mechanism) with a DRG.3-compliant cryptographic post-processing algorithm (with memory) such that the output rate of the post-processing algorithm is no greater than the rate of its input from the noise source. Like the PTG.2 functionality class, the raw random numbers produced by the noise source can be interpreted as realizations of random variables that are time-locally stationarily distributed.

Usually, a PTG.2-compliant physical RNG provides the data that are input into the cryptographic post-processing algorithm (e.g., as seed material for reseeding, as high-entropy additional input), which are called intermediate random numbers in this context. This is a minimum requirement for compliance to class PTG.3. Recall that the cryptographic post-processing algorithm does not increase the size of the intermediate data.

At the cost of a higher compression rate, it is also possible to use an appropriate physical RNG for which the entropy per output bit is smaller than the PTG.2-specific min-entropy (or Shannon entropy) bound. For constructions with a PTG.2-compliant core PTRNG, a device-specific entropy claim is optional but stronger than the above-mentioned minimum requirement. For other PTG.3-compliant constructions, a device-dependent entropy claim is mandatory.

Additionally, a certificate applicant can make the following device-dependent entropy claims, which are denoted by v_S (Shannon entropy claim) and v_m (min-entropy claim):

(a) Shannon entropy per output bit is $\geq v_S$ for some $v_S \in [0.9998, 1 - 2^{-32}]$;

(b) min-entropy per output bit is $\geq v_m$ for some $v_m \in [0.98, 1 - 2^{-32}]$; or

(c) Shannon entropy per output bit i $\geq v_S$ for some $v_S \in [0.9998, 1 - 2^{-32}]$, *and* the min-entropy per output bit is $\geq v_m$ for some $v_m \in [0.98, 1 - 2^{-32}]$.

The lower entropy bounds coincide with the class-specific values defined in class PTG.2. The upper entropy bound per output bit $1 - 2^{-32}$ coincides with the NIST definition of full entropy.

An online test, total failure test, and start-up test are also mandatory for functionality class PTG.3. A stochastic model of the noise source is the central part of the evaluation, and this stochastic model shall be justified and analyzed for functionality class PTG.2. The effectiveness of the online test and the total failure test must be verified.

Entropy claims greater than 0.9998 (Shannon entropy) and 0.98 (min-entropy) per output bit cannot be verified by the stochastic model alone but require additional data compression.

NTG.1. The NTG.1 functionality class defines hybrid non-physical true RNGs (NPTRNGs) by combining (non-physical) noise sources (including the digitization mechanisms) with a DRG.3-compliant cryptographic post-processing algorithm. While the noise sources are typically classified as non-physical, physical noise sources are also permitted.

Class NTG.1 only allows min-entropy claims per output bit within the range $[0.98, 1 - 2^{-32}]$. Shannon entropy claims are not permitted. The entropy claim has to be justified. However, unlike functionality classes PTG.2 and PTG.3, a stochastic model is not required.

At start-up, the non-physical true RNG (NPTRNG) shall not generate any random numbers until the entropy pool has collected entropy from at least two different noise sources that each contribute at least 240 bits of min-entropy and employ different principles for providing randomness.

After start-up, it is desirable but not required that the entropy is generated from more than one noise source. The raw random numbers are tested when the RNG is in operation.

3.2. NIST RBG Constructions

SP 800-90C specifies approved RBG designs called *RBG constructions*. RBG constructions include:

(*i*) A DRBG (mechanism) from SP 800-90A that ensures that the outputs of the RBG are indistinguishable from the ideal distribution by a computationally bounded adversary

(*ii*) A randomness source that is either an entropy source that generates truly random bits in compliance with SP 800-90B or another RBG construction that complies with SP 800-90C

Entropy sources. SP 800-90B entropy sources obtain entropy from one or more noise sources that may rely on physical or non-physical phenomena. If multiple noise sources are used in an entropy source, one noise source is designated as the *primary* noise source. Only entropy provided by the primary noise source is credited as providing entropy during validation, even though other noise sources may contribute entropy as well. If the primary

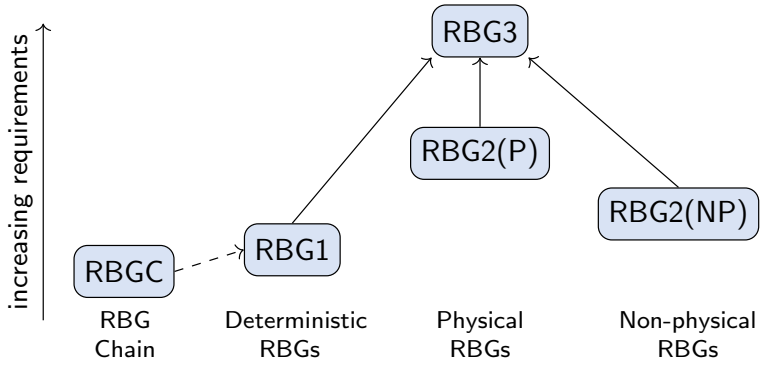


Fig. 5. SP 800-90C RBG constructions

noise source is physical, the entropy source is classified as a physical entropy source. Otherwise, the entropy source is classified as a non-physical entropy source. There is no hard limit on the minimum entropy per output bit. Health tests are performed while the entropy source is in operation.

Entropy sources can also use optional conditioning components (i.e., deterministic functions that are responsible for reducing bias and/or increasing the entropy rate of the resulting output bits). The conditioning components can be designed in various ways. SP 800-90B provides a list of vetted conditioning components, namely HMAC with any NIST-approved hash function, CMAC and CBC-MAC with the AES block cipher, any NIST-approved hash function, and `Hash_df` and `Block_Cipher_df`, as specified in SP 800-90A. SP 800-90B allows the use of non-vetted conditioning components with some restrictions. However, to generate full-entropy outputs, vetted conditioning components must be used.

DRBG (mechanisms). SP 800-90A specifies several approved Deterministic Random Bit Generator mechanisms (DRBGs), namely, the `Hash_DRBG`, `HMAC_DRBG`, and `CTR_DRBG`. These DRBGs are based on approved cryptographic algorithms that, once provided with seed material containing sufficient randomness, can be used to generate random bits suitable for cryptographic applications.

RBG constructions. SP 800-90C defines four RBG constructions: RBG1, RBG2, RBG3, and RBGC. Each contains a DRBG mechanism from SP 800-90A. All RBG constructions provide backtracking resistance.

RBG1. This construction is suitable for an application in which no internal randomness source is available within its security boundary. An RBG1 construction is instantiated once in its lifetime over a physical secure channel from an external RBG with appropriate security properties, including the use of a physical entropy source (i.e., an RBG2(P) construction, an RBG3 construction, or the root RBGC construction with an RBG3 construction or Full

Entropy Source serving as the initial randomness source; see below). An RBG1 construction does not have access to a randomness source after instantiation so cannot be reseeded to provide prediction resistance.

RBG2. This construction includes one or more SP 800-90B-compliant entropy sources within its security boundary that are used to instantiate and reseed the DRBG within the construction. This construction can be reseeded to provide prediction resistance when sufficient entropy is available (e.g., in a pool) or can be obtained from the RBG's entropy sources when a reseed is requested. There are two types of RBG2 constructions, depending on the type of the underlying entropy source:

(*i*) RBG2(P) construction if the entropy is only credited when provided by a physical entropy source

(*ii*) RBG2(NP) construction if the entropy is credited from non-physical entropy sources or from both non-physical and physical entropy sources

An RBG2 construction allows only designs approved in SP 800-90A for the DRBG mechanism. The construction provides backtracking resistance and can provide prediction resistance when sufficient fresh entropy is inserted by reseeding using the entropy sources.

RBG3. This construction is designed to provide output with a security strength equal to the requested length of its output by producing outputs that have *full entropy*. Only entropy provided by physical entropy sources is credited. This construction continually provides prediction resistance and has two types:

(*i*) RBG3(XOR) construction combines the output of one or more validated entropy sources with the output of an instantiated, approved DRBG using an exclusive-or (XOR) operation.

(*ii*) RBG3(RS) construction uses one or more validated entropy sources to provide randomness input for the DRBG by continuously reseeding.

RBGC. This construction allows a DRBG to seed and (optionally) reseed another DRBG. A DRBG tree consists of only RBGC constructions on the same platform (e.g., a computer). The initial RBGC construction in the chain is called the root RBGC construction. It accesses an initial randomness source, which may be an RBG2 or RBG3 construction or a Full Entropy Source. Each RBGC construction (after the root) has only one parent (i.e., a direct predecessor) but may have multiple children (i.e., direct successors), thus forming a tree of RBGC constructions. A non-root RBGC construction may be reseeded by its parent or an alternative randomness source (i.e., a sibling of the parent, an ancestor RBGC construction,

or the initial randomness source). A DRBG tree works in the same manner as the DRNG tree described in Sec. 3.1.

3.3. Comparison of Functionality Classes and RBG Constructions

This section provides an overview of the similarities and differences between the functionality classes of BSI and the RBG constructions of NIST. Table 1 provides mappings between the functionality classes and the RBG constructions. Table 2 gives additional information for the remaining classes and constructions. The aim of this section is to promote an understanding of AIS 20/31 and the SP 800-90 series. This, of course, cannot replace a detailed study of both standards for concrete questions about specific designs.

Sections 3.3.1 through 3.3.6 discuss the similarities and differences between corresponding functionality classes and RBG constructions. They provide the information necessary for a design to be successfully validated under both the BSI and NIST/CSE programs. Furthermore, additional evaluation tasks are mentioned. Section 3.3.7 considers the DRG.2 functionality class and RBG3(XOR) constructions for which no natural equivalents in the other scheme exist.

Table 1. Mapping between BSI functionality classes and NIST RBG constructions that are most similar.

Functionality Class		RBG Construction
DRG.3	$\leftrightarrow$	RBG1
DRT.1	$\leftrightarrow$	DRBG tree (RBGC)
DRG.4	$\leftrightarrow$	RBG2(P)
PTG.2	$\leftrightarrow$	physical entropy source
PTG.3	$\leftrightarrow$	RBG3(RS)
NTG.1	$\leftrightarrow$	RBG2(NP)

Table 2. Additional mapping information. DRG.2 does not map to any RBG constructions. Parts of the RBG3(XOR) construction may map to functionality classes.

Functionality Class		RBG Construction
DRG.2		–
PTG.2 + DRG.3, possibly PTG.3	$\leftarrow$	RBG3(XOR)

3.3.1. PTG.2 and an Entropy Source

The PTG.2 functionality class is similar to an SP 800-90B physical entropy source. Both apply similar tests during operation; post-processing/conditioning is optional; and neither is intended for direct use but are relevant for other functionality classes and constructions, either as a component or as an entropy provider.

Validating PTG.2 as an entropy source

- [design] The PTG.2 implementation must contain the necessary health tests. In particular, on-demand and continuous health tests must be added if they are not present.
- [design] During evaluation, output values have to successfully pass SP 800-90B compliance tests (i.e., all tests and predictors that are required for entropy-source validation).

Certifying an entropy source as PTG.2

- [design] The noise source used in the entropy source must be physical.
- [design] The output of the noise source (i.e., the raw random numbers) must follow a time-locally stationary distribution.
- [design] The entropy per output bit must satisfy the PTG.2 entropy requirements that are described in Sec. 3.1. The PTG.2 entropy requirements can be achieved through additional conditioning if necessary.
- [evaluation] A stochastic model for the noise source must be provided, and the output entropy source will have to pass the black box test suite T_{irn} (see Sec. 3.1).
- [evaluation] The effectiveness of the online and total failure tests must be verified.

3.3.2. DRG.3 and RBG1

The DRG.3 functionality class and the RBG1 construction resemble each other, as shown in Fig. 6. Neither of them contains an internal source of randomness, so both must be seeded from an external source.

This section considers only single DRNGs / DRBGs. DRT.1-compliant seed trees (composed of DRG.3 DRNGs) and DRBG chains (RBGC constructions) are compared in Sec. 3.3.3.

Validating DRG.3 as RBG1

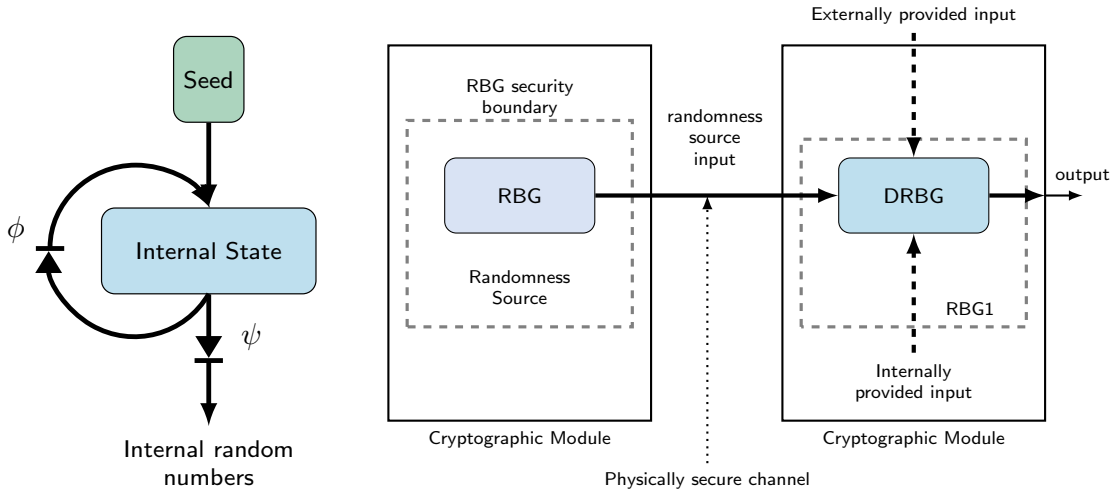
In order for a DRG.3 implementation to be compliant with an RBG1 construction, several design aspects must be present:

- [design] The algorithmic part of the implementation must be an approved DRBG mechanism from SP 800-90A.
- [design] For seeding, a randomness source that conforms to functionality class PTG.3 or DRG.4 must be used if they also satisfy the design requirements of an RBG3(RS) or RBG2(P) construction, respectively (see Sec. 3.3.6 and 3.3.4). Reseeding is not permitted. If a DRG.4-compliant DRNG is used for seeding, enhanced forward secrecy shall be provided by reseeding to the DRG.4-compliant DRNG before it generates the seed material; cf. the description of functionality class DRG.4 for more details. Seeding with an NTG.1-compliant NPTRNG is not permitted.
- [design] The seed material produced by a PTG.3 or DRG.4 must meet the RBG1 entropy requirements, and the implementation must have known answer tests.

Certifying RBG1 as DRG.3

An RBG1 implementation can be validated as compliant with functionality class DRG.3 if the following requirements of class DRG.3 are verified:

- [design] The RBG1 construction needs to satisfy the algorithmic requirements of the DRG.3 functionality class. The algorithmic requirements for DRG.3 include, for example, that the effective internal state is at least 246 bits.
- [design, evaluation] Seeding can be done with one of the following functionality classes: PTG.3, PTG.2, NTG.1, or DRG.4. Other TRNGs are also permitted as randomness sources, such as RBG3(RS), RBG3(XOR), or entropy sources. Alternatively, the seed material for the RBG1 construction can be generated by a DRG.4-compliant DRNG or an RBG2(P) construction serving as the randomness source. In these cases, enhanced forward secrecy or prediction resistance must be ensured by the randomness source before the seed material is generated; cf. the description of functionality class DRG.4 for details. If the RNG that generates the seed material has not been certified, evidence for the claimed entropy is necessary.
- [evaluation] The algorithmic and non-algorithmic requirements (e.g., that the min-entropy of the effective internal state after seeding is at least 240 bits) of functionality class DRG.3 need to be verified.
- [evaluation] Algorithmic verification is waived for Hash_DRBG and HMAC_DRBG due to existing conformity proofs in AIS 20/31.



(a) DRG.3 functionality class **(b)** RBG1 construction

Fig. 6. Neither the DRG.3 functionality class (a) nor the RBG1 construction (b) contain an internal randomness source.

3.3.3. DRT.1 and DRBG Tree (RBGC construction)

The DRT.1 functionality class and DRBG tree are rather similar. Fig. 7 illustrates the concept of a DRNG tree.

In SP 800-90C, the DRBG tree is realized through the RBGC construction. The root RBGC construction consists of an initial randomness source and a DRBG. An RBGC construction can serve as the randomness source for other RBGC constructions and becomes part of them (see Fig. 8 and 9).

Validating a DRT.1-compliant DRNG tree as a DRBG chain (RBGC construction)

In order for a DRT.1-compliant DRNG tree implementation to be compliant with an SP 800-90C DRBG tree, several design aspects must be present:

- [design] The algorithmic parts of the DRNGs must be an approved DRBG mechanism from SP 800-90A.
- [design] The randomness source used to generate the seed material for the root RBGC construction must be compliant with an RBG3(RS), RBG3(XOR), RBG2(P), or RBG2(NP) construction or a Full Entropy Source. This is particularly the case if a PTG.3-compliant PTRNG meets the requirement of an RBG3(RS) construction, if a PTG.2-compliant PTRNG with external conditioning function meets the requirements of a Full Entropy Source, if an NTG.1 NPTRNG meets the requirement of an RBG2(NP) construction, or if a DRG.4 meets the requirement of an RBG2(P) construction. In

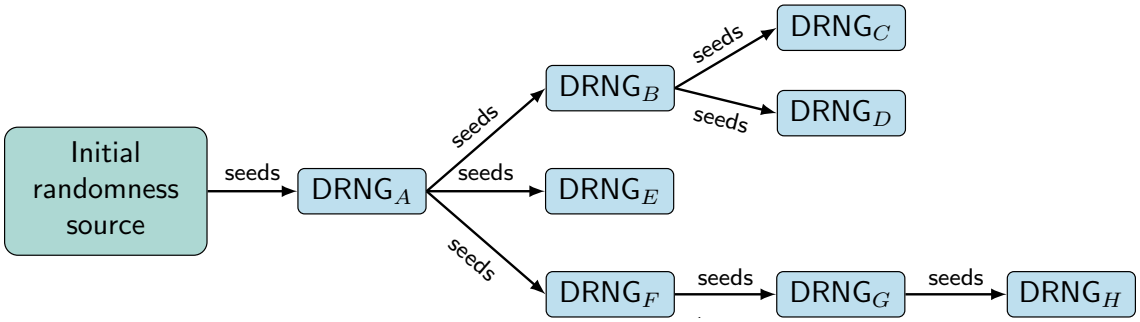


Fig. 7. Illustrating example of an AIS 20/31 DRNG tree. The initial randomness source provides entropy for the whole DRNG tree. Arrows point to the direct seed successor (child node in the DRNG tree). DRNG$_A$ is the root DRNG.

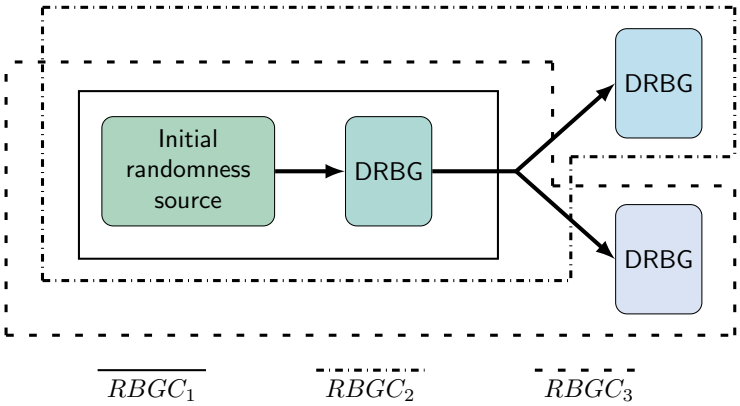


Fig. 8. Two RBGC constructions seeded by the same root RBGC

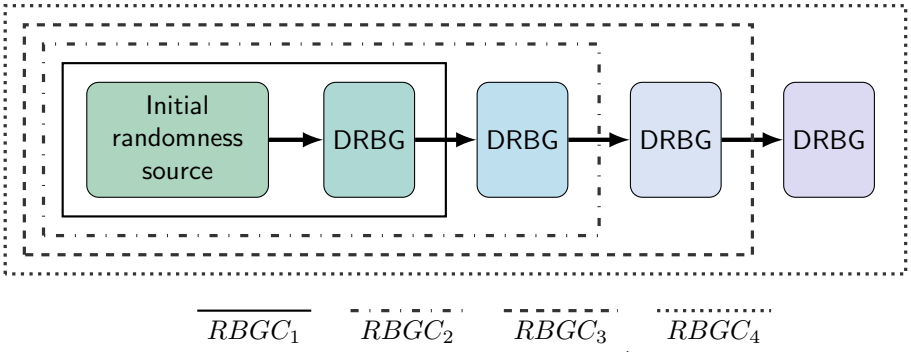


Fig. 9. A DRBG chain within a DRBG tree: each RBGC construction consists of a DRBG and RBGC ancestors up to the root RBGC construction $RBGC_1$.

the latter case, enhanced forward secrecy shall be provided to the DRG.4-compliant DRNG by reseeding before it generates the seed material; cf. the description of functionality class DRG.4 for more details.

- [design] The seed material for the root RBGC construction must be at least $3s/2$ bits long with at least s bits of randomness when the randomness source is compliant with an RBG2 or RBG3 construction or must provide $3s/2$ bits of entropy if the randomness source is a full-entropy source.
- [design] The implementation must have known answer tests.
- [design] The initial randomness source and all DRNGs of the DRNG tree shall be implemented and operated inside the same platform (e.g., a computer).

Certifying a DRBG tree as a DRT.1-compliant DRNG tree

A DRBG tree implementation can be validated as an AIS 20/31-compliant DRNG tree if the following requirements of functionality class DRT.1 are verified:

- [design] The RBGC construction needs to satisfy the algorithmic requirements of the DRG.3 functionality class.
- [design, evaluation] Seeding the root DRNG can be done with a randomness source that is compliant with one of the following functionality classes: PTG.3, PTG.2, NTG.1, or DRG.4. Other RNGs are also permitted as a randomness source, such as RBG3(RS), RBG3(XOR), Full Entropy Sources, or RBG2(P). Evidence for the claimed randomness is necessary. If a DRG.4-compliant DRNG or a RBG2(P) is used, enhanced forward secrecy or prediction resistance must be ensured by the randomness source before the seed material is generated.
- [evaluation] The algorithmic and non-algorithmic requirements (e.g., that the min-entropy of the effective internal state after seeding is at least 240 bits) of functionality class DRG.3 need to be verified.
- [evaluation] Algorithmic verification is waived for Hash_DRBG and HMAC_DRBG due to existing conformity proofs in AIS 20/31.

3.3.4. DRG.4 and RBG2(P)

The RBG2(P) construction is most similar to the DRG.4 functionality class, as shown in Fig. 10. Both provide strong security in the backward direction, are capable of providing strong security in the forward direction, and have access to an internal physical randomness source that can be used for seeding and reseeding. For a DRG.4 implementation, the randomness

source must be a PTRNG. In particular, PTRNGs that are compliant with functionality class PTG.2 or PTG.3 are considered appropriate. For an RBG2(P) construction, the randomness source must be an SP 800-90B entropy source using a physical noise source as its primary noise source.

Validating DRG.4 as RBG2(P)

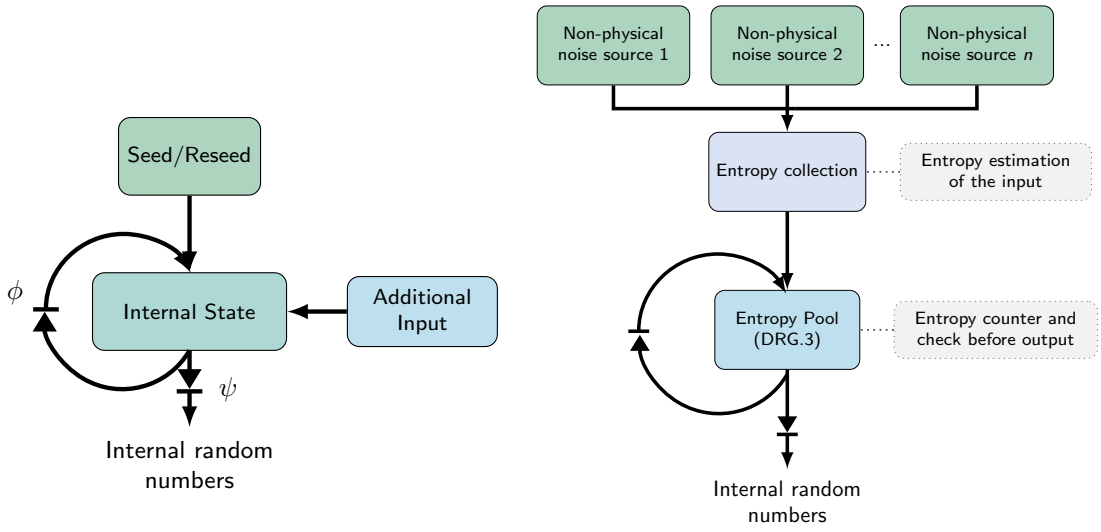
- [design] A DRG.4 implementation that is compliant with an RBG2(P) construction must use an approved DRBG mechanism from SP 800-90A.
- [design] A physical RNG compliant with classes PTG.2 or PTG.3 may be used for seeding and reseeding, along with a min-entropy claim for that source.
- [design] Prediction resistance is achieved only by reseeding the DRNG with sufficient entropy, not by additional input. However, the insertion of entropy in the additional input is allowed.
- [design] The seed material produced by the physical RNG must meet the RBG2(P) entropy requirements, and known answer tests must be included in the design.

Certifying RBG2(P) as DRG.4

- [design] The RBG2(P) construction needs to satisfy the algorithmic requirements of the DRG.4 functionality class. The algorithmic requirements for DRG.4 include, for example, that the effective internal state is at least 246 bits.
- [evaluation] The algorithmic and non-algorithmic requirements of class DRG.4 need to be verified (e.g., that the min-entropy of the effective internal state after seeding and reseeding is at least 240 bits of fresh entropy).
- [evaluation] Algorithmic verification is waived for Hash_DRBG and HMAC_DRBG due to existing conformity proofs in AIS 20/31.
- [evaluation] A stochastic model is required for the noise source of the PTRNG or entropy source that is used within the RBG2(P) construction, and it has to be ensured that the PTRNG or entropy source is working properly when generating the seed material. These requirements are satisfied if the PTRNG or entropy source is compliant to class PTG.2 or PTG.3.

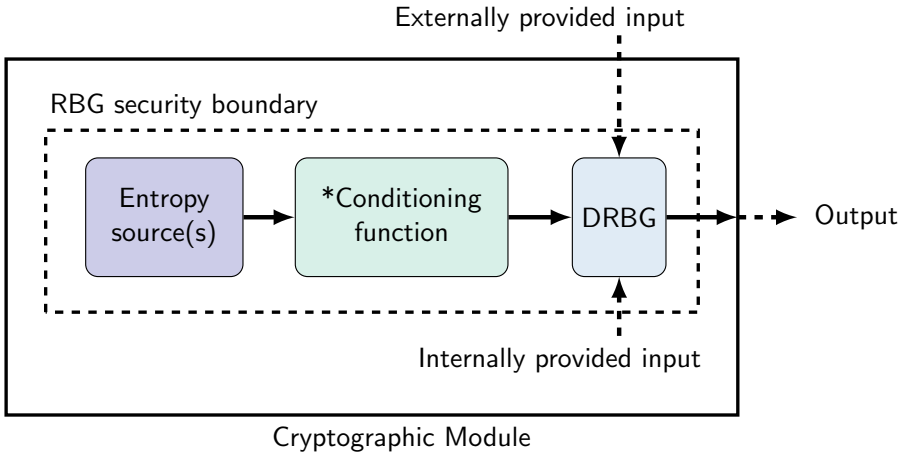
3.3.5. NTG.1 and RBG2(NP)

The AIS 20/31 NTG.1 functionality class and the SP 800-90C RBG2(NP) construction rely on the use of non-physical noise sources, as shown in Fig. 10. Class NTG.1 describes true RNGs and is the AIS 20/31 non-physical counterpart to PTG.3. In SP 800-90C, the requirements for



(a) DRG.4 functionality class

(b) NTG.1 functionality class.



(c) RBG2 construction

Fig. 10. The DRG.4 functionality class (a) and RBG2 construction (c) are similar when a physical noise source is used. When the noise source is non-physical, the RBG2 construction is most similar to the NTG.1 functionality class (b).

an RBG2(NP) construction fall between the RBG1 and RBG2(P) constructions (see Sec. 3.3.2 and 3.3.4).

In general, the evaluation of physical noise sources provides greater assurance than that of non-physical noise sources. One reason is that physical noise sources allow stochastic models. Furthermore, the environment in which non-physical noise sources are being operated is usually not under the control of the developer or evaluator.

Validating NTG.1 as RBG2(NP)

- [design] The NTG.1 implementation must use a DRBG mechanism from SP 800-90A for post-processing.
- [design] After instantiation, fresh entropy can only be credited if it is introduced into the DRBG state through reseeding.
- [design] The noise sources in the NTG.1 implementation (including the digitization mechanism) need to conform to an SP 800-90B entropy source (see Sec. 3.3.5). In particular, the noise source needs to pass SP 800-90B-compliant tests and predictors.
- [design] A known-answer test that tests the DRG.3 component must be present.

Certifying RBG2(NP) as NTG.1

- [design] Fresh entropy must constantly be introduced into the DRBG state through reseeding such that the the min-entropy per output bit is greater than or equal to some lower bound v_m in $[0.98, 1 - 2^{-32}]$.
- [design] Before the RNG outputs the first random numbers, at least two noise sources that employ different principles to provide randomness must generate at least 240 bits of min-entropy.
- [design, evaluation] The algorithmic requirements of class DRG.3 need to be verified (waived for the `Hash_DRBG` and the `HMAC_DRBG`).

3.3.6. PTG.3 and RBG3(RS)

The AIS 20/31 PTG.3 functionality class and an SP 800-90C RBG3(RS) construction have the strongest requirements for their respective models and provide the strongest security assurances. They require cryptographic post-processing with memory that (viewed as a deterministic RNG/RBG) provides enhanced backward secrecy and backtracking resistance. Fresh entropy is incorporated constantly from the noise sources within the security boundary to ensure that the entropy per RNG/RBG output bit is close to 1. A PTG.3-compliant PTRNG requires a physical noise source. For an RBG3(RS)-construction, only the entropy

from physical entropy sources is credited. In particular, the continuous incorporation of fresh entropy guarantees enhanced forward secrecy and prediction resistance.

An RBG3(RS) construction is comparable to functionality class PTG.3 (see Fig. 11).

Validating PTG.3 as RBG3(RS)

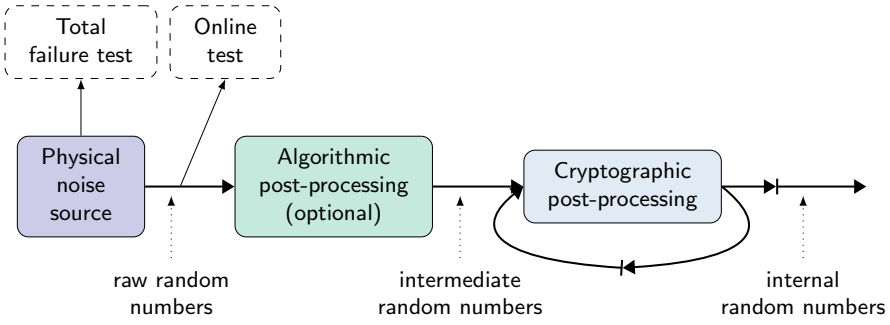
- [design] The post-processing algorithm in the PTG.3 implementation must be compliant with a DRBG approved in SP 800-90A.
- [design] The fresh entropy has to be included by reseeding the post-processing algorithm. When more entropy is needed than can be supplied during a reseed, additional entropy may be acquired directly from the noise sources and inserted as additional input. The min-entropy per output bit must be $\geq 1 - 2^{-32}$, which corresponds to full entropy.
- [design] An on-demand health test, a continuous health test, and a known-answer test for the post-processing algorithm must be added if they are not present. During evaluation, the output of the noise source (i.e., raw random numbers) and the input data of the cryptographic post-processing algorithm (i.e., the output data of an inner PTG.2, if it exists) will have to successfully pass SP 800-90B compliance tests.
- [design] A known-answer test that tests the DRG.3 component must be present.

Certifying RBG3(RS) as PTG.3

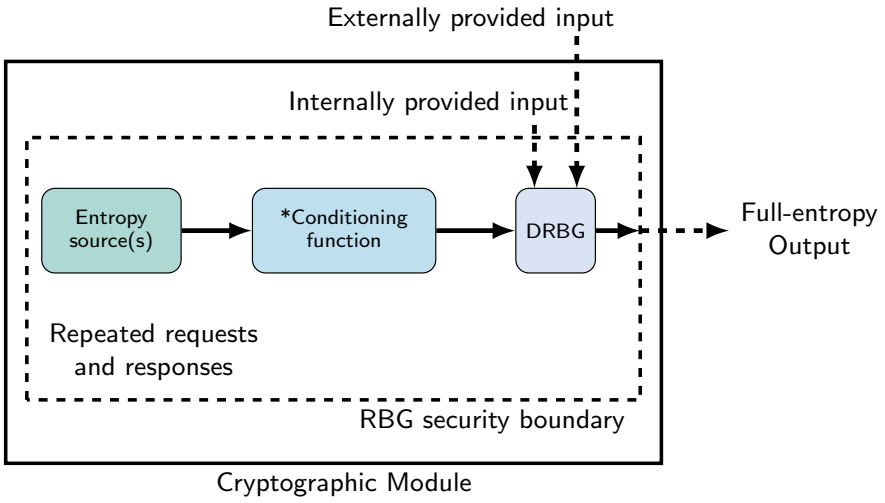
- [design] The distribution of the output data of the noise source (i.e., raw random numbers) must be time-locally stationarily distributed.
- [evaluation] It needs to be verified that the DRBG part of the RBG3(RS) construction satisfies the algorithmic requirements of functionality class DRG.3 (waived for the Hash_DRBG and the HMAC_DRBG) (see Sec. 3.3.2).
- [evaluation] A stochastic model for the noise source in the entropy source must be provided, and the effectiveness of the online test and the total failure tests must be verified.

3.3.7. Other Classes (DRG.2) and Constructions (RBG3(XOR))

Functionality class DRG.2 defines requirements for deterministic RNGs that ensure backward secrecy and forward secrecy but does not provide enhanced backward secrecy. SP 800-90 does not have a construction that is comparable to the DRG.2 functionality class. DRG.2-compliant RNGs can be an option for resource-constrained devices.



(a) PTG.3 functionality class



(b) RBG3(RS) construction

Fig. 11. The PTG.3 functionality class and RBG3(RS) construction

The RBG3(XOR) construction has no functionality class in AIS 20/31 that is directly comparable. However, several options are discussed below.

Certifying RBG3(XOR) as PTG.2 and DRG.3

If the following two conditions are met, the RBG3(XOR) construction is compliant to both functionality class PTG.2 and functionality class DRG.3:

- The physical entropy source in the RBG3(XOR) construction (plus the external conditioning function, if it exists) is compliant with class PTG.2 (see Section 3.3.1).
- The DRBG in the RBG3(XOR) construction is compliant with class DRG.3 (see Sec. 3.3.2).

However:

- (i) If only the first condition is fulfilled, the RBG3(XOR) construction is compliant with the PTG.2 functionality class but not with class DRG.3.
- (ii) If only the second condition is fulfilled, the RBG3(XOR) construction is compliant with the DRG.3 functionality class but not with class PTG.2.
- (iii) If the DRBG does not influence the physical entropy source, then both branches of the RBG3(XOR) construction, the physical entropy source plus (optional) conditioning function, and the DRBG can be evaluated separately.

If the RBG3(XOR) construction satisfies the first condition and a stronger version of the second condition above, namely that the DRBG is compliant with class DRG.4 (see Sec. 3.3.4), then the previous assertions remain valid for DRG.4 in place of DRG.3. The RBG3(XOR) construction is then compliant to both functionality class PTG.2 and functionality class DRG.4.

Certifying RBG3(XOR) as PTG.3

If the following two conditions are satisfied, the RBG3(XOR) construction is compliant with functionality class PTG.3:

- The physical entropy source (plus the conditioning function, if it exists) is compliant with class PTG.3 (see Sec. 3.3.6).
- The physical entropy source (plus the conditioning function, if it exists) and the DRBG are independent.

3.3.8. Rough Comparison Between Noise Source Health Tests in SP 800-90B and AIS 20/31

Section 2.4.1 specifies the health tests in SP 800-90 (i.e., start-up tests, continuous tests, on-demand tests), while Sec. 2.4.2 explains the concepts of start-up tests, online tests, and total failure tests in AIS 20/31. The main difference is that in AIS 20/31, the developer has to select tests of their choice and give evidence that the selected tests are effective.

4. Terminology Comparison

This section compares the terminology used in the BSI and NIST standards on random number generation. The definitions provided below are taken from the corresponding glossaries of the BSI and NIST standards. Certain definitions have been revised to enhance consistency and alignment. When the definition of a term is not available in a standard, it is represented by the "—" symbol.

Each term is accompanied by a symbol that indicates the level of alignment between its definitions in BSI and NIST standards.

- The symbol ● indicates that the meaning of the term is identical in both standards, although the wording may not be exactly the same.
- The symbol ◐ indicates that there are minor differences in the meaning of the term.
- The symbol ○ is used when the term is defined or used in only one of the standards.
- *Additional notes* provide information about relevant terms.

additional input ○

Source	*Definition*
BSI	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.
NIST	Optional additional information that could be provided in a generate or reseed request by a consuming application.

Additional notes:

Personalization strings that are provided during `instantiation` can also be considered as additional input.

adversary ●

Source	*Definition*
BSI	A malicious entity whose goal is to determine, to guess, or to influence the output of an RNG. The term attacker is used synonymously.
NIST	A malicious entity whose goal is to determine, guess, or influence the output of an RBG.

Additional notes:
RNG and RBG are used interchangeably.

algorithmic post-processing ○

Source	*Definition*
BSI	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 (in contrast to cryptographic post-processing). Algorithmic post-processing can be stateful (i.e., with memory) or stateless. *Note 2:* Typical examples of algorithmic post-processing algorithms: XORing bits or binary vectors, modular addition, linear feedback shift registers.
NIST	—

Additional notes:
The SP 800-90 series uses the term *conditioning component* or *conditioning function*.

alternative randomness source ○

Source	*Definition*
BSI	—
NIST	A sibling of the parent randomness source, an ancestor of the RBGC construction to be reseeded, or the initial randomness source.

ancestor (randomness source) ○

Source	*Definition*
BSI	—
NIST	A parent, grandparent, or other direct RBGC predecessor of an RBGC construction.

approved ○

Source	*Definition*
BSI	—
NIST	An algorithm or technique for a specific cryptographic use that is specified in a FIPS or NIST recommendation, adopted in a FIPS or NIST recommendation, or specified in a list of NIST-approved security functions.

Additional notes:
AIS 20/31 does not specify any BSI-approved RNG designs or BSI-approved online tests.

attacker ○

Source	*Definition*
BSI	Synonym for adversary.
NIST	—

backtracking resistance ○

Source	*Definition*
BSI	—
NIST	A property of a DRBG that provides assurance that compromising the current internal state of the DRBG does not weaken previously generated outputs. See SP 800-90A for a more complete discussion. Contrast with prediction resistance.

Additional notes:
Backtracking resistance corresponds to *enhanced backward secrecy*, as used in AIS 20/31.

backward secrecy ○

Source	*Definition*
BSI	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.
NIST	—

biased ●

Source	*Definition*
BSI	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.
NIST	A random variable is said to be biased if values of the finite sample space are selected with unequal probability. Contrast with unbiased.

bitstring ●

Source	*Definition*
BSI	A finite sequence of ones and zeroes.
NIST	An ordered sequence (string) of 0s and 1s. The leftmost bit is the most significant bit.

Additional notes:
This term is used in AIS 20/31 as two words (bit string) but as a single word (bitstring) in the SP 800-90 series.

block cipher ○

Source	*Definition*
BSI	—
NIST	A parameterized family of permutations on bitstrings of a fixed length; the parameter that determines the permutation is a bitstring called the key.

Additional notes:
Although not defined in BSI standards, block cipher is a well-established term in cryptographic literature.

compression rate ○

Source	*Definition*
BSI	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.
NIST	—

computational security ○

Source	*Definition*
BSI	Security against an adversary with bounded computing power. Quantified by the security level (of cryptographic mechanisms).
NIST	—

computing platform ○

Source	*Definition*
BSI	—
NIST	A system's hardware, firmware, operating system, and all applications and libraries executed by that operating system. Components that communicate with the operating system through a peripheral bus or a network, either physical or virtual, are not considered to be part of the same computing platform.

conditioning (of noise source output) ○

Source	*Definition*
BSI	—
NIST	A method of processing the raw data to reduce bias and/or ensure that the entropy rate of the conditioned output is no less than some specified amount.

Additional notes:
Conditioning within an entropy source may be either algorithmic or cryptographic and may be memoryless or stateful. A list of vetted conditioning functions has been provided in SP 800-90B.

conditioning function (external) ○

Source	*Definition*
BSI	—
NIST	As used in SP 800-90C, a deterministic function that is used to produce a bitstring with full entropy or to distribute entropy across a bitstring.

Additional notes:
AIS 20/31 uses the term *post-processing*.

consuming application ●

Source	*Definition*
BSI	An application that uses random outputs from an RNG.
NIST	An application that uses random outputs from an RBG.

cryptographic boundary ○

Source	*Definition*
BSI	—
NIST	An explicitly defined physical or conceptual perimeter that establishes the physical and/or logical bounds of a cryptographic module and contains all of the hardware, software, and/or firmware components of a cryptographic module.

cryptographic module ○

Source	*Definition*
BSI	—
NIST	The set of hardware, software, and/or firmware that implements cryptographic functions (including cryptographic algorithms and key generation) and is contained within the cryptographic boundary.

cryptographic post-processing ○

Source	*Definition*
BSI	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 AIS 20/31, cryptographic post-processing is always stateful. Cryptographic constructions without memory (i.e., which do not grant DRNG security properties) are denoted as stateless post-processing with cryptographic functions.
NIST	—

Additional notes:
In SP 800-90, cryptographic post-processing falls under the umbrella of *conditioning*. It can be applied to either noise source output within an entropy source or to the output of the entropy source (external conditioning).

deterministic random bit generator (DRBG) / deterministic random number generator (DRNG) ●

Source	*Definition*
BSI	DRNG: An RNG that produces random numbers from a secret initial value called a seed or seed material by applying a deterministic algorithm. *Note:* A deterministic RNG, at least initially, has access to a randomness source.
NIST	DRBG: An RBG that produces random bitstrings by applying a deterministic algorithm to seed material.

Additional notes:
A DRBG has access to a randomness source, at least initially.
The seed material is secret.
An SP 800-90 DRBG corresponds to an AIS 20/31 DRNG.

digitization ●

Source	*Definition*
BSI	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 can 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.
NIST	The process of generating raw discrete digital values from non-deterministic events (e.g., analog noise sources) within a noise source.

direct seed predecessor ○

Source	*Definition*
BSI	A DRNG from which a (non-root) DRNG in a DRNG tree receives initial seed material.
NIST	—

Additional notes:
This corresponds to *parent randomness source* in SP 800-90.

DRBG chain ○

Source	*Definition*
BSI	—
NIST	A chain of DRBGs in which one DRBG is used to provide seed material for another DRBG.

DRG.2 ○

Source	*Definition*
BSI	A DRNG construction that provides assurance of backward secrecy and forward secrecy. *Note:* The DRNG may be hybrid but need not.
NIST	—

DRG.3 ○

Source	*Definition*
BSI	A DRNG construction that provides assurance of backward secrecy, enhanced backward secrecy, and forward secrecy. *Note:* The DRNG may be hybrid but need not.
NIST	—

DRG.4 ○

Source	*Definition*
BSI	A DRNG construction that provides assurance of backward secrecy, forward secrecy, and enhanced backward secrecy and can also provide enhanced forward secrecy. *Note:* The DRNG must be hybrid to provide enhanced forward secrecy.
NIST	—

DRNG tree ●

Source	*Definition*
BSI	A set of DRNGs of which one DRNG is and remains distinguished (“root DRNG’’). This includes the information about the direct seed predecessors. The set of DRNGs can be static or dynamic in the sense that DRNGs can be uninstantiated and new DRNGs can be instantiated. All DRNGs except the root DRNG receive seed material from another DRNG from this set. The root DRNG receives seed material from an initial randomness source. The initial randomness source generates seed material to which entropy can be assigned. *Note:* Functionality class DRT.1 formulates requirements for DRNG trees.
NIST	A set of DRBGs within RBGC constructions that originate with the DRBG in a root RBGC construction. The root obtains seed material from an initial randomness source, but all other DRBGs receive seed material from another DRBG in the tree.

DRT.1 ○

Source	*Definition*
BSI	A DRNG tree construction that provides assurance of backward secrecy, enhanced backward secrecy, and forward secrecy for all DRNGs in the tree.
NIST	—

effective internal state ○

Source	*Definition*
BSI	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.
NIST	—

enhanced backward secrecy ○

Source	*Definition*
BSI	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 3:* A term related to enhanced backward secrecy is backtracking resistance (from SP 800-90[A,B,C]).
NIST	—

enhanced forward secrecy ○

Source	*Definition*
BSI	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 SP 800-90[A,B,C]).
NIST	—

entropy ●

Source	*Definition*
BSI	A measure of disorder, randomness, or variability in a closed system. *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 AIS 20/31, Shannon entropy and min-entropy are used, depending on the context. *Note 3:* Min-entropy is the measure used in SP 800-90.
NIST	A measure of disorder, randomness, or variability in a closed system.

entropy rate ○

Source	*Definition*
BSI	—
NIST	The validated rate at which an entropy source provides entropy in terms of bits per entropy-source output (e.g., five bits of entropy per eight-bit output sample).

entropy source ●

Source	*Definition*
BSI	—
NIST	The combination of a noise source, health tests, and an optional conditioning component that produce bitstrings containing entropy. A distinction is made between entropy sources having physical noise sources and those having non-physical noise sources.

Additional notes:
A PTG.2-compliant PTRNG can be viewed as a (coarse) equivalent of a physical entropy source that generates random numbers whose entropy per bit is very close to 1.

external random number ○

Source	*Definition*
BSI	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 be cut off. *Note 2:* (PTRNG): If the PTRNG runs continuously, many internal random numbers might never be output.
NIST	—

false positive ○

Source	*Definition*
BSI	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.
NIST	—

Additional notes:
Also known as *type I error*, which is used in the NIST documents.

forward secrecy ○

Source	*Definition*
BSI	Assurance that knowledge about subsequent output values cannot be derived with practical computational effort from the knowledge of current or previous output values. *Note 1:* "Deriving knowledge" means gaining significant advantage over blind guessing. *Note 2:* TRNGs can ensure forward secrecy by entropy.
NIST	—

fresh entropy ●

Source	*Definition*
BSI	A random bit string recently generated by a noise source. In particular, "fresh" means that the bit string has not been previously used to generate output or has otherwise been made externally available. *Note:* The randomness source should be compliant with PTG.2, PTG.3, or NTG.1.
NIST	A bitstring that is output from a non-deterministic randomness source that has not been previously used to generate output or has otherwise been made externally available.

Additional notes:
For the SP 800-90 series, the randomness source should be an entropy source or RBG3 construction.

fresh randomness ○

Source	*Definition*
BSI	—
NIST	A bitstring that is output from a randomness source that has not been previously used to generate output or has not otherwise been made externally available.

full-entropy bitstring ○

Source	*Definition*
BSI	—
NIST	A bitstring with ideal randomness (i.e., the amount of entropy per bit is equal to 1). The SP 800-90 series assumes that a bitstring has full entropy if the entropy rate is at least $1 - \varepsilon$, where ε is at most 2^{-32}.

Additional notes:
AIS 20/31 does not actively use the term "full entropy," but the min-entropy claim of $1 - 2^{-32}$ bits per output bit is the maximum for the functionality classes PTG.3 and NTG.1.

full-entropy source ○

Source	*Definition*
BSI	—
NIST	An SP 800-90B-compliant entropy source that has been validated as providing output with full entropy or the combination of a validated SP 800-90B-compliant entropy source and an external conditioning function that provides full-entropy output.

granularity level ○

Source	*Definition*
BSI	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.
NIST	—

hash function ○

Source	*Definition*
BSI	—
NIST	A (mathematical) function that maps values from a large (possibly very large) domain into a smaller range. The function satisfies the following properties: 1) (One-way) It is computationally infeasible to find any input that maps to any pre-specified output; 2) (Collision-free) It is computationally infeasible to find any two distinct inputs that map to the same output.

Additional notes:
NIST-approved hash functions are specified in [17–19].

health testing ○

Source	*Definition*
BSI	—
NIST	Testing within an implementation immediately prior to or during normal operations to obtain assurance that the implementation continues to perform as implemented and validated.

Additional notes:
In the terminology of AIS 20/31, health tests are comprised of start-up tests, online tests, and total failure tests. In the terminology of SP 800-90[A,B,C], health tests are comprised of continuous tests and start-up tests.

hybrid DRNG ○

Source	*Definition*
BSI	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.
NIST	—

hybrid PTRNG ○

Source	*Definition*
BSI	A hybrid true RNG with a physical noise source; see also hybrid true RNG.
NIST	—

hybrid RNG ○

Source	*Definition*
BSI	An RNG that uses design elements from DRNGs and TRNGs. *Note:* This is an intuitive but rough characterization.
NIST	—

hybrid true RNG ○

Source	*Definition*
BSI	A true RNG with cryptographic post-processing. 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 (entropy extraction). *Note:* Cryptographic post-processing may be viewed as an additional security anchor for the case where the entropy per output bit is smaller than assumed.
NIST	—

ideal randomness source/ideal RNG ●

Source	*Definition*
BSI	Ideal RNG: A mathematical construct that generates independent and uniformly distributed random numbers.
NIST	Ideal randomness source: The source of an ideal random sequence of bits. Each bit of an ideal random sequence is unpredictable and unbiased, with a value that is independent of the values of the other bits in the sequence. Prior to an observation of the sequence, the value of each bit is equally likely to be 0 or 1, and the probability that a particular bit will have a particular value is unaffected by knowledge of the values of any or all of the other bits. An ideal random sequence of n bits contains n bits of entropy.

independent entropy sources ○

Source	*Definition*
BSI	—
NIST	Two entropy sources are *independent* if knowledge of the output of one entropy source provides no information about the output of the other entropy source.

independent and identically distributed (iid) ○

Source	*Definition*
BSI	—
NIST	A quality of a sequence of random variables for which each element of the sequence has the same probability distribution as the other values, and all values are mutually independent.

information-theoretic security ○

Source	*Definition*
BSI	Security against an adversary with unlimited computing power. Requires fresh entropy.
NIST	—

initial randomness source ●

Source	*Definition*
BSI	The randomness source for the root DRNG of a DRNG tree; cf. randomness.
NIST	The randomness source for the root RBGC construction in a DRBG tree of RBGC constructions.

instantiate ○

Source	*Definition*
BSI	—
NIST	The process of initializing a DRBG with sufficient randomness to generate pseudorandom bits at the desired security strength.

intermediate random number ○

Source	*Definition*
BSI	(PTG.3- and NTG.1-specific term) Input data for cryptographic post-processing. Example: Consider a PTG.3-compliant RNG 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.
NIST	—

internal random number ○

Source	*Definition*
BSI	Final stage of the random numbers of an RNG that are ready to be output. Compare to external random numbers.
NIST	—

internal state ●

Source	*Definition*
BSI	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.
NIST	The collection of all secret and non-secret information about an RBG or entropy source that is stored in memory at a given point in time.

Kerckhoffs's principle ○

Source	*Definition*
BSI	A security analysis is conducted under the basic assumption that the design and public keys of a cryptosystem are known to an adversary. Only secret keys and seed material are assumed to be unknown to an adversary.
NIST	—

known-answer test/known answer test ●

Source	*Definition*
BSI	Known-answer test: A test that uses a fixed input/output pair to test the correctness of a deterministic mechanism.
NIST	Known answer test: A test that uses a fixed input/output pair to detect whether a deterministic component was implemented correctly or continues to operate correctly.

min-entropy ●

Source	*Definition*
BSI	A measure of entropy based on the minimal (worst-case) gain of information from an observation.
NIST	A lower bound on the entropy of a random variable. The precise formulation for min-entropy is $(-\log_2 \max p_i)$ for a discrete distribution having probabilities $p_1, \ldots, p_k$. Min-entropy is often used as a measure of the unpredictability of a random variable.

multi-target attack ○

Source	*Definition*
BSI	A scenario in which an adversary applies guesses or the results of a precomputation to attack many instances of the same cryptosystem at once in hope that at least one instance succumbs to the attack.
NIST	—

must ○

Source	*Definition*
BSI	—
NIST	A requirement that may not be testable by a CMVP testing lab. Note: **Must** may be coupled with **not** to become **must not**.

noise alarm ○

Source	*Definition*
BSI	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.
NIST	—

noise source ●

Source	*Definition*
BSI	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.
NIST	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.

non-physical entropy source ○

Source	*Definition*
BSI	—
NIST	An entropy source whose primary noise source is non-physical.

non-physical noise source ●

Source	*Definition*
BSI	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 of system data: RAM data, system time of a PC, or the output of API functions. Examples of interaction: key strokes, mouse movement, etc.
NIST	A noise source that typically exploits system data and/or user interaction to produce digitized random data.

non-physical true RNG (NPTRNG) ○

Source	*Definition*
BSI	A true RNG with a non-physical noise source.
NIST	—

non-validated entropy source ○

Source	*Definition*
BSI	—
NIST	An entropy source that has not been validated by the CMVP as conforming to SP 800-90B.

NTG.1 ○

Source	*Definition*
BSI	An NPTRNG construction that provides assurance that the entropy per bit is above a class-specific bound. Heuristic assessments and total failure tests detect non-tolerable weaknesses of the random numbers. Furthermore, it includes cryptographic post-processing with memory for which the input rate is (significantly) larger than the output rate.
NIST	—

null string ○

Source	*Definition*
BSI	—
NIST	An empty bitstring.

one-way function ○

Source	*Definition*
BSI	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.
NIST	—

online test ○

Source	*Definition*
BSI	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."
NIST	—

parent randomness source ○

Source	*Definition*
BSI	—
NIST	The randomness source used to seed a non-root RBGC construction during the instantiation of its DRBG.

Additional notes:
Corresponds to *direct seed predecessor* in AIS 20/31.

personalization string ●

Source	*Definition*
BSI	An optional input value to a DRNG during instantiation to make one RNG instance behave differently from other instances. *Note:* Can be a secret parameter or public parameter.
NIST	An optional input value to a DRBG during instantiation to make one DRBG instantiation behave differently from other instantiations.

physical entropy source ○

Source	*Definition*
BSI	—
NIST	An entropy source whose primary noise source is physical.

physical noise source ●

Source	*Definition*
BSI	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 1:* 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 2:* Physical noise sources are used by physical true RNGs (PTRNGs).
NIST	A noise source that exploits physical phenomena (e.g., thermal noise, shot noise, jitter, metastability, radioactive decay) from dedicated hardware designs (e.g., using diodes, ring oscillators) or physical experiments to produce digitized random data.

physically secure channel ○

Source	*Definition*
BSI	—
NIST	A physical trusted and safe communication link that is established between an implementation of an RBG1 construction and its randomness source to securely communicate unprotected seed material without relying on cryptography. A physically secure channel protects against eavesdropping as well as physical or logical tampering by unwanted operators/entities, processes, or other devices between the endpoints.

physical true RNG (PTRNG) ○

Source	*Definition*
BSI	A TRNG that uses a physical noise source. *Note 1:* The abbreviated version "physical RNG" is used instead of "physical true RNG" because all physical RNGs are, by definition, true RNGs. *Note 2:* The abbreviation "PTRNG" is used instead of "PRNG" to avoid confusion with pseudorandom number generators.
NIST	—

post-processing ○

Source	*Definition*
BSI	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:* AIS 20/31 distinguishes between algorithmic post-processing, cryptographic post-processing (i.e., with memory), and (in general: stateless) post-processing with cryptographic functions. *Note 3:* Post-processing is related to the term conditioning function in SP 800-90.
NIST	—

prediction resistance ○

Source	*Definition*
BSI	—
NIST	A property of a DRBG that provides assurance that compromising the current internal state of the DRBG does not allow future DRBG outputs to be predicted past the point where the DRBG has been reseeded with sufficient entropy. See SP 800-90A for a more complete discussion. Contrast with backtracking resistance.

Additional notes:
Prediction resistance (as used in the SP 800-90 series) corresponds to enhanced forward secrecy (as used in AIS 20/31).

predictor ○

Source	*Definition*
BSI	An (r,s)-predictor makes a prediction about the next s random numbers on the basis of the previous r random numbers ($r \in \mathbb{N}_0 \cup \{\infty\}$, $s \in \mathbb{N}$).
NIST	—

pseudorandom number generator ●

Source	*Definition*
BSI	Another term for DRNG.
NIST	Another term for DRBG.

PTG.2 ○

Source	*Definition*
BSI	A PTRNG construction that provides assurance that the entropy per bit is above a class-specific bound close to 1 (Shannon entropy). Alternatively or additionally, a class-specific min-entropy bound (close to 1) can also be claimed. A start-up test, a total failure test, and an online test detect non-tolerable weaknesses of the random numbers. Usually, no cryptographic post-processing is applied. For the evaluation, a stochastic model of the noise source is required. *Note:* A PTG.2-compliant PTRNG is a coarse equivalent to a physical entropy source in the sense of SP 800-90B. One difference is that entropy sources ensure individual entropy bounds per bit that can be (much) smaller than the PTG.2-specific min-entropy bound 0.98.
NIST	—

PTG.3 ○

Source	*Definition*
BSI	A PTRNG construction that provides assurance that the entropy per bit is above an entropy bound close to 1. A start-up test, a total failure test, and an online test detect non-tolerable weaknesses of the random numbers. Furthermore, it includes cryptographic post-processing with memory for which the input rate is larger than or equal to the output rate. For data-compressing post-processing algorithms, very small entropy defects (min-entropy) are possible. For the evaluation, a stochastic model of the noise source is required.
NIST	—

pure DRNG ○

Source	*Definition*
BSI	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 (by itself) a reseeding procedure.
NIST	—

pure PTRNG ○

Source	*Definition*
BSI	A PTRNG in which any post-processing 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 results in weak pseudorandom output if simple (non-cryptographic) algorithmic post-processing is implemented.
NIST	—

random bit generator (RBG)/random number generator (RNG) ●

Source	*Definition*
BSI	RNG: A group of components or an algorithm that outputs sequences of discrete values (usually represented as bit strings called internal random numbers).
NIST	RBG: A device or algorithm that outputs a random sequence that is effectively indistinguishable from statistically independent and unbiased bits.

randomness ◐

Source	*Definition*
BSI	An intuitive characterization of the unpredictability of a bit string. If the bit string is generated by a true RNG, its unpredictability is quantified by entropy. If the bit string is generated by a DRNG, its unpredictability is based on the unpredictability of the seed material of the generating DRNG, on the secrecy of its internal state, and on the difficulty for an adversary to break the cryptographic guarantees of the DRNG (in particular, forward secrecy, backward secrecy, and enhanced backward secrecy).
NIST	The unpredictability of a bitstring. If the randomness is produced by a nondeterministic source (e.g., an entropy source or RBG3 construction), the unpredictability is dependent on the quality of the source. If the randomness is produced by a deterministic source (e.g., a DRBG), the unpredictability is based on the capability of an adversary to break the cryptographic algorithm for producing the pseudorandom bitstring.

randomness source ○

Source	*Definition*
BSI	—
NIST	A source of randomness for an RBG. The randomness source may be an entropy source or an RBG construction.

raw random number ○

Source	*Definition*
BSI	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.
NIST	—

Additional notes:
The raw random numbers correspond to the output of the noise source.

RBG1 construction ○

Source	*Definition*
BSI	—
NIST	An RBG construction with the DRBG and the randomness source in separate cryptographic modules.

Additional notes:
An RBG1 construction corresponds to the BSI's DRG.3 functionality class.

RBG2 construction ○

Source	*Definition*
BSI	—
NIST	An RBG construction with one or more entropy sources and a DRBG within the same cryptographic module. This RBG construction does not provide full-entropy output.

Additional notes:
An RBG2 construction has two variants: RBG2(P) and RBG2(NP).

RBG2(NP) construction ○

Source	*Definition*
BSI	—
NIST	A non-physical RBG2 construction. An RBG2 construction that obtains entropy from one or more validated non-physical entropy sources and possibly from one or more validated physical entropy sources. This RBG construction does not provide full-entropy output.

Additional notes:
An RBG2(NP) construction corresponds to BSI's NTG.1 functionality class.

RBG2(P) construction ○

Source	*Definition*
BSI	—
NIST	A physical RBG2 construction. An RBG construction that includes a DRBG and one or more entropy sources in the same cryptographic module. Only the entropy from validated physical entropy sources is counted when fulfilling an entropy request within the RBG. This RBG construction does not provide full-entropy output.

Additional notes:
An RBG2(P) construction corresponds to BSI's DRG.4 functionality class.

RBG3 construction ○

Source	*Definition*
BSI	—
NIST	An RBG construction that includes a DRBG and one or more entropy sources in the same cryptographic module. When working properly, bitstrings that have full entropy are produced. Sometimes called a *non-deterministic random bit generator* (NRBG) or true random number (or bit) generator.

Additional notes:
The RBG3 construction has two variants: RBG3(XOR) and RBG3(RS).

RBGC construction ○

Source	*Definition*
BSI	—
NIST	An RBG construction used within a DRBG tree in which one DRBG is used to provide seed material for another DRBG. The construction does not provide full-entropy output.

reseed ●

Source	*Definition*
BSI	To refresh the internal state of a DRNG with seed material. The seed material should contain sufficient entropy (sufficient randomness if generated by a DRNG) to allow recovery from a possible compromise. *Note:* A reseeding procedure shall utilize the previous internal state. Even if an adversary knew the seed material, it shall not be significantly easier to determine the new internal state than the previous one.
NIST	To refresh the internal state of a DRBG with seed material from a randomness source.

root RBGC construction ○

Source	*Definition*
BSI	—
NIST	The first RBG construction in a DRBG tree of RBGC constructions.

sample space ○

Source	*Definition*
BSI	—
NIST	The set of all possible outcomes of an experiment.

secret parameter ○

Source	*Definition*
BSI	An optional input value to the seeding procedure or reseeding procedure of a DRNG or to the initialization of the cryptographic post-processing algorithm of a PTRNG to achieve additional security against adversaries who are not in possession of this value.
NIST	—

security boundary ●

Source	*Definition*
BSI	A physical or conceptual perimeter that confines the secure domain that an adversary cannot observe or influence in a malicious way (according to the chosen threat model).
NIST	For an entropy source: A conceptual boundary that is used to assess the amount of entropy provided by the values output from the entropy source. The entropy assessment is performed under the assumption that any observer (including any adversary) is outside of that boundary during normal operation. For a DRBG: A conceptual boundary that contains all of the DRBG functions and internal states required for a DRBG. For an RBG: A conceptual boundary that is defined with respect to one or more threat models that includes an assessment of the applicability of an attack and the potential harm caused by the attack.

security strength ●

Source	*Definition*
BSI	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.
NIST	A number associated with the amount of work (i.e., the number of basic operations of some sort) that is required to "break" a cryptographic algorithm or system in some way. In the SP 800-90 series, the security strength is specified in bits and is a specific value from the set 128, 192, 256. If the security strength associated with an algorithm or system is s bits, then it is expected that (roughly) 2^s basic operations are required to break it. Note: This is a classical definition that does not consider quantum attacks.

seed ●

Source	*Definition*
BSI	To initialize the internal state of a DRNG with seed material. The seed material should contain sufficient entropy (sufficient randomness if generated by a DRNG) to meet the security requirements.
NIST	To initialize the internal state of a DRBG with seed material. The seed material should contain sufficient randomness to meet security requirements.

seed material ●

Source	*Definition*
BSI	A bit string containing entropy (containing randomness if generated by a DRNG) that is used to seed or reseed a DRNG. *Note:* This definition also applies to the cryptographic post-processing algorithm (with memory) of a TRNG.
NIST	An input bitstring from a randomness source that provides an assessed minimum amount of randomness (e.g., entropy) for a DRBG.

seedlife ●

Source	*Definition*
BSI	The period between (re-)seeding the internal state of a DRNG and subsequent reseeding with new seed material, or between (re-)seeding the internal state of a DRNG and uninstantiating the DRNG.
NIST	The period between instantiating or reseeding a DRBG with seed material and reseeding the DRBG with seed material containing fresh randomness or uninstantiation of the DRBG.

shall ○

Source	*Definition*
BSI	—
NIST	The term used to indicate a requirement that is testable by a testing lab. **Shall** may be coupled with **not** to become **shall not**. See *testable requirement.*

Shannon entropy ○

Source	*Definition*
BSI	A measure of entropy based on the expected (average) gain of information from an observation.
NIST	—

should ○

Source	*Definition*
BSI	—
NIST	The term used to indicate an important recommendation. Ignoring the recommendation could result in undesirable results. Note: **Should** may be coupled with **not** to become **should not**.

sibling (randomness source) ○

Source	*Definition*
BSI	—
NIST	A sibling of the parent randomness source for a non-root RBGC construction (i.e., the sibling can be considered as the "aunt" or "uncle" in "human family" terms). The "grandparent" of the non-root RBGC construction is the parent of both the parent randomness source and the parent's sibling.

start-up test/start-up testing ●

Source	*Definition*
BSI	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.
NIST	Start-up testing: A suite of health tests that are performed every time the entropy source is initialized or powered up. These tests are carried out on the noise source before any output is released from the entropy source.

state handle ○

Source	*Definition*
BSI	—
NIST	A pointer to the internal state information for a particular DRBG instantiation.

stationarily distributed ○

Source	*Definition*
BSI	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.
NIST	—

statistical inconspicuousness ○

Source	*Definition*
BSI	The application of standard statistical tests does not distinguish the generated random numbers from ideal random numbers.
NIST	—

stochastic model ●

Source	*Definition*
BSI	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 the 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 can suffice to model parts of the entropy contributions if it can be shown that the neglected effects do not decrease the entropy.
NIST	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. Formally, a stochastic model consists of a family of probability distributions that contains the true distribution of the noise source output or suitably defined auxiliary random variables during the lifetime of the entropy source containing that noise source, even if the quality of the digitized data goes down. The stochastic model is based on and justified by the understanding of the noise source.

subordinate DRBG (sub-DRBG) ○

Source	*Definition*
BSI	—
NIST	A DRBG that is instantiated by an RBG1 construction and contained within the same security boundary as the RBG1 construction.

support a security strength (by a DRBG) ○

Source	*Definition*
BSI	—
NIST	The DRBG has been instantiated at a security strength that is equal to or greater than the security strength requested for the generation of random bits.

targeted security strength ○

Source	*Definition*
BSI	—
NIST	The security strength that is intended to be supported by one or more implementation-related choices (e.g., algorithms, cryptographic primitives, auxiliary functions, parameter sizes, and/or actual parameters).

testable requirement ○

Source	*Definition*
BSI	—
NIST	A requirement that can be tested for compliance by a testing lab via operational testing, a code review, or a review of relevant documentation provided for validation. A testable requirement is indicated using a **shall** statement.

threat model ○

Source	*Definition*
BSI	—
NIST	A description of a set of security aspects that need to be considered. A threat model can be defined by listing a set of possible attacks along with the probability of success and the potential harm from each attack.

time-local stationarity ○

Source	*Definition*
BSI	A sequence of random variables $X_1, X_2, \ldots$ is called "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, however, "large" compared to periods needed to generate samples for the online tests and the evaluator tests.
NIST	—

total failure ○

Source	*Definition*
BSI	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., from an amplifier), but this scenario still constitutes a total failure.
NIST	—

total failure test ○

Source	*Definition*
BSI	A test that shall reliably detect total failures and prevent 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.
NIST	—

true RNG ○

Source	*Definition*
BSI	A device or mechanism for which each output value depends on newly generated data from a noise source.
NIST	—

unbiased ●

Source	*Definition*
BSI	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.
NIST	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.

uninstantiate ●

Source	*Definition*
BSI	Uninstantiating an instance of a DRNG means that this instance no longer exists. In particular, the internal state and secret parameters are destroyed (i.e., securely deleted).
NIST	The termination of a DRBG instantiation.

validated entropy source ○

Source	*Definition*
BSI	—
NIST	An entropy source that has been successfully validated by the CAVP and CMVP for conformance to SP 800-90B.

widely recognized cryptographic primitives ○

Source	*Definition*
BSI	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.
NIST	—

with memory ○

Source	*Definition*
BSI	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.
NIST	—

References

[1] Barker EB, Kelsey JM (2015) SP 800-90A Recommendation for Random Number Generation Using Deterministic Random Bit Generators (National Institute of Standards and Technology), DOI:10.6028/NIST.SP.800-90Ar1

[2] Sönmez Turan M, Barker EB, Kelsey JM, McKay KA, Baish ML, Boyle M (2018) SP 800-90B Recommendation for the Entropy Sources Used for Random Bit Generation (National Institute of Standards and Technology), DOI:10.6028/NIST.SP.800-90B

[3] Barker EB, Kelsey JM, McKay KA, Roginsky A, Sönmez Turan M (2025) SP 800-90C Recommendation for Random Bit Generator (RBG) Constructions (National Institute of Standards and Technology), DOI:10.6028/NIST.SP.800-90C

[4] BSI (2025) AIS 20: Funktionalitätsklassen und Evaluationsmethodologie für deterministische Zufallszahlengeneratoren (Version 4) (Bundesamt für Sicherheit in der Informationstechnik (BSI)), Report. Available at https://www.bsi.bund.de/dok/ais-20-31.

[5] BSI (2025) AIS 31: Funktionalitätsklassen und Evaluationsmethodologie für physikalische Zufallszahlengeneratoren (Version 4) (Bundesamt für Sicherheit in der Informationstechnik (BSI)), Report. Available at https://www.bsi.bund.de/dok/ais-20-31.

[6] Peter M, Schindler W (2024) A Proposal for Functionality Classes for Random Number Generators (Version 3.0) (Bundesamt für Sicherheit in der Informationstechnik (BSI)), Report. Available at https://www.bsi.bund.de/dok/ais-20-31-appx-2024.

[7] ISO Central Secretary (2012) ISO/IEC 19790:2012 Information technology — Security techniques — Security requirements for cryptographic modules (International Organization for Standardization, Geneva, CH), Standard ISO/IEC 19790:2012. Available at https://www.iso.org/standard/52906.html.

[8] ISO Central Secretary (2015) ISO/IEC 15408-1:2009 Information technology — Security techniques — Evaluation criteria for IT security — Part 1: Introduction and general model (International Organization for Standardization, Geneva, CH), Standard ISO/IEC 15408-1:2009. Available at https://www.iso.org/standard/50341.html.

[9] ISO Central Secretary (2011) ISO/IEC 18031:2011 Information technology — Security techniques — Random bit generation (International Organization for Standardization, Geneva, CH), Standard ISO/IEC 18031:2011. Available at https://www.iso.org/standard/54945.html.

[10] ISO Central Secretary (2019) Information technology — Security techniques — Test and analysis methods for random bit generators within ISO/IEC 19790 and ISO/IEC 15408 (International Organization for Standardization, Geneva, CH), Standard ISO/IEC 20543:2019. Available at https://www.iso.org/standard/68296.html.

[11] Rukhin A, Soto J, Nechvatal J, Smid M, Barker E, Leigh S, Levenson M, Vangel M, Banks D, Heckert N, Dray J, Vo S, Bassham L (2010) SP 800-22 Rev. 1a A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications (National Institute of Standards and Technology), Available at https://doi.org/10.6028/NIST.SP.800-22r1a.

[12] Killmann W, Schindler W (2011) A proposal for: Functionality classes for random number generators (Version 2.0) (Bundesamt für Sicherheit in der Informationstechnik (BSI)), Report. Available at https://www.bsi.bund.de/dok/ais-20-31-appx-2011.

[13] von Neumann J (1951) Various techniques used in connection with random digits. *Monte Carlo Method*, eds Householder A, Forsythe G, Germond H (National Bureau of Standards Applied Mathematics Series, 12, Washington, D.C.: U.S. Government Printing Office), pp 36–38.

[14] Samuelson PA (1968) Constructing an unbiased random sequence. *Journal of the American Statistical Association* 63(324):1526–1527. DOI:10.1080/01621459.1968.10480945. Available at https://www.tandfonline.com/doi/abs/10.1080/01621459.1968.10480945

[15] Peres Y (1992) Iterating Von Neumann's Procedure for Extracting Random Bits. *The Annals of Statistics* 20(1):590 – 597. DOI:10.1214/aos/1176348543. Available at https://doi.org/10.1214/aos/1176348543

[16] National Institute of Standards and Technology (2008) The Keyed-Hash Message Authentication Code (HMAC), (Department of Commerce, Washington, DC), Federal Information Processing Standards Publication (FIPS) NIST FIPS 198-1. DOI:10.6028/NIST.FIPS.198-1.

[17] National Institute of Standards and Technology (2012) Secure Hash Standard (SHS), (Department of Commerce, Washington, DC), Federal Information Processing Standards Publication (FIPS) NIST FIPS 180-4. DOI:10.6028/NIST.FIPS.180-4.

[18] National Institute of Standards and Technology (2015) SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions, (Department of Commerce, Washington, DC), Federal Information Processing Standards Publication (FIPS) NIST FIPS 202. DOI:10.6028/NIST.FIPS.202.

[19] Sönmez Turan M, McKay KA, Kang J, Kelsey J, Chang D (2025) NIST SP 800-232 Ascon-Based Lightweight Cryptography Standards for Constrained Devices: Authenticated Encryption, Hash, and Extendable Output Functions (National Institute of Standards and Technology), DOI:10.6028/NIST.SP.800-232