7 DRBG Functional Model and Functional Objectives and Requirements

7.1 Functional Model and General Objectives and Requirements

Part 1 of this Standard provides a general functional model that is applicable to both NRBGs and DRBGs. Figure 1 provides a functional model for deterministic random bit generators (DRBGs) that particularizes the functional model of Part 1.

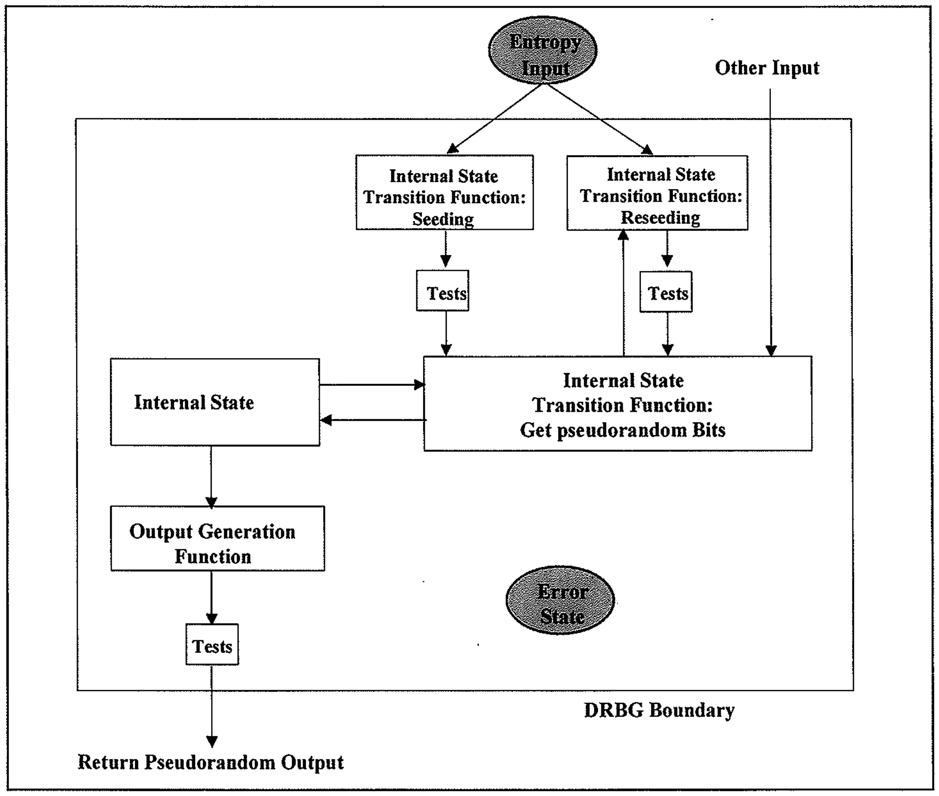
Entropy Input
Other Input
Internal State Transition Function: Seeding
Internal State Transition Function: Reseeding
Tests
Tests
Internal State
Internal State Transition Function: Get pseudorandom Bits
Output Generation Function
Error State
Tests
DRBG Boundary
Return Pseudorandom Output

Figure 1: DRBG Model

7.2 Functional Requirements

7.2.1 Introduction

Part 1 of this Standard provides general functional requirements for random bit generators. These requirements are specified below in italics, followed by a discussion about how they are satisfied by DRBGs in this part of the Standard.

7.2.2 General Functional Requirements

The following functional requirements apply to all random bit generators:

1. *The implementation **shall** be designed to allow validation, including specific design assertions about what the RBG is prohibited from doing. In the case of an NRBG, validation is intended to determine that the NRBG behaves as expected, not just during normal operation, but also at the boundaries of the intended operational conditions. Security-relevant branches in the code that govern behavior in exceptional conditions (e.g. initialization, failed health tests, etc.) **shall** be validated by forcing all error conditions to occur during validation testing. As many error conditions as can be feasibly created on the operational system should be tested; it is acceptable to use a special non-operational system that faithfully replicates nteh operational system, but also has capabilities that facilitate this testing.*
   Implementation validation is discussed in Section 11.2. Test points have been included in the functional model (see Figure 1) and in figures for each DRBG in Section 10.
2. ~~*The RBG **shall** satisfy all the appropriate top-level requirements.*~~
   ~~This has been addressed in Section 7.1.~~
3. *There **shall** be design documentation that describes how the RBG is intended to meet all security requirements, including protection from misbehavior.*
   This requirement **shall** be met for DRBGs claiming conformance to this Standard. For DRBGs, this is accomplished by incorporating the DRBG into a FIPS 140-2 cryptographic module. In addition, implementations that are submitted for validation testing **shall** include documentation describing the handling of data obtained from an entropy source. .

Comment [ebb1]: Page: 32
This is a requirement on the validator, not the implementer. I don't think that this belongs here.

Comment [ebb2]: Page: 1
This is not a suitable requirement, since there are no top-level requirements any longer.

Optional attributes for the functions in an RBG are as follows:

1. *The RBG **should** be capable of supporting backtracking resistance. If this attribute is supported, it means that, from the perspective of a glass box, it **shall not** be feasible to compute or predict any prior output bit, up to the selected security level.*
   Backtracking resistance is provided for most DRBGs and is discussed in Section 8.7.
2. *The RBG **may** be capable of supporting prediction resistance. If this attribute is supported, it means that, from the perspective of a glass box at any particular point in time, it **shall not** be feasible to compute or predict any future output bit.*
   Prediction resistance is optional for a DRBG, and is discussed in Section 8.7 and specified in Section 10 for the DRBGs in this Standard.

7.2.3 Functional Requirements for Entropy Input

The entropy source is the source of digitized bits for the DRBG. At a minimum, this source **shall** provide the requested amount of entropy for a DRBG. Examples of appropriate entropy sources are an Approved NRBG as specified in Part 2 of this Standard, or an Approved DRBG or chain of DRBGs in which the first DRBG in the chain has an Approved NRBG as an entropy source. However, the use of other entropy sources are allowed. In this case, the entropy source requirements of Part 2 **shall** apply (see below).

Some of the DRBGs specified in Section 10 of this Standard allow for some bias in the entropy source. Whenever a bitstring containing entropy is required by the DRBG, a request is made to an entropy source that indicates the minimum amount of entropy to be returned, and the minimum and maximum length of the bitstring. The DRBG expects that, over the entire bitstring, the requested amount of entropy is provided, as a minimum. Additional entropy beyond the amount requested is not required, but is desirable. For those DRBGs for which the entropy source must provide full entropy, only a single length is allowed (i.e., the minimum and maximum lengths are the same).

An important use of the entropy source for DRBGs is the acquisition of entropy bits to create seeds. Seeds **shall** be obtained prior to requesting pseudorandom bits during each DRBG instance. Additional entropy **may** also be intoduced during a DRBG instance.

Part 1 of this Standard provides functional requirements for the entropy sources for random bit generators. These requirements are specified below. They are met, for example, when an entropy source that conforms to Part 2 of this Standard is used, and the interface between the entropy source and the DRBG is protected against influence, manipulation and observation. DRBGs and other sources that provide entropy **shall** also meet these requirements.

The requirements for the entropy source of an RBG are:

1. *The entrop yinput **shall** be based upon well-established physical principles, or extensively characterized behavior. These principles **shall** be documented.*
2. *The entropy rate **shall** be assessable, or the collection **shall** be self-regulating, so that the amount of entropy per collection unit or event will reliably obtain or exceed a designed lower bound. This aspect of the RBG design **shall** be documented.*
3. *The entropy input **shall** be designed so that direct manipulation (such as the ability to control the entropy input), predictable and controllable influence (such as the ability to bias entropy input), anddirect observationof the entropy input can be prevented, or at least detected. This aspect of the RBG design **shall** be documented.*
4. *Loss or severe degradation of an entropy input **shall** be detectable. This aspect of the design **shall** be documented.*

An optional feature is the following:

1. *The entropy input **may** be formed from multiple sources of entropy to improve resiliency to possible degradation or misbehavior This can help meet the requirement that the possibility of misbehavior is sufficiently small. It may be the case that an entropy source is already composed from multiple sources of entropy; this case is certainly allowed, although the entropy assessment of such an entropy source may be more complex.*

Figure 2 depicts the method that **shall** be used when the entropy input is not provided by an Approved NRBG or DRBG. In this case, a translation and smoothing function is used to translate the information from the entropy source to bits and possibly remove some of the bias from the entropy source, producing a pool of bits whose entropy is then assessed. When a pool of bits with sufficient entropy is available, the bits **shall** be processed by a derivation function (see Section 9.6.4) prior to passing the requested bits to the DRBG. Documentation **shall** describe all elements of this method.

7.2.4 Functional Requirements for Non-secret Inputs

Other information is required by a DRBG as input during the generation process. This information includes the input parameters when the DRBG is called by the consuming application and any additional input that may be public (e.g., information provided by a user). Input information **shall** be checked for validity when possible. Depending on the DRBG, time variant information **may** also be required, e.g., a counter or a date/time value.
A counter used by a deterministic RBG **shall not** repeat during a DRBG instance. When a DRBG instantiation is seeded or reseeded, the counter **may** be set to a fixed value (e.g., set to 1), but **shall** be updated for each state of a DRBG instance. See Section 8.3 for a discussion of DRBG instantiation and instances.
A date/time value used by a DRBG instantiation **shall not** repeat. Whenever a date/time value is requested by a DRBG, either a different date/time value **shall** be used than was previously used for the DRBG instantiation, or another technique **shall** supplement the date/time value to provide uniqueness (e.g., a counter is concatenated to the date/time value).

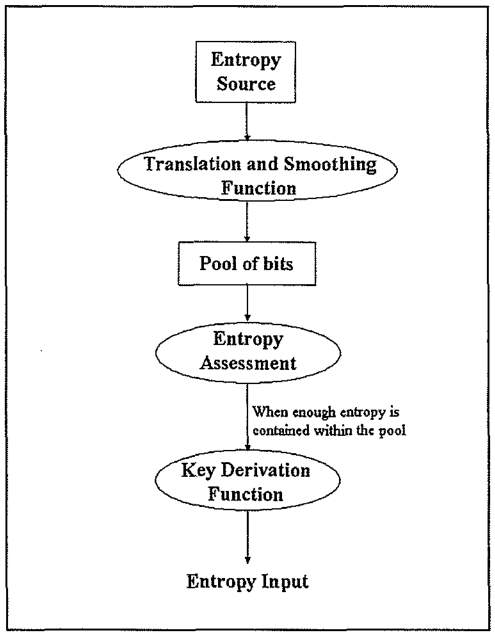
Entropy Source
Translation and Smoothing Function
Pool of bits
Entropy Assessment
When enough entropy is contained within the pool
Key Derivation Function
Entropy Input

Figure 2: Method for Obtaining Entropy Input Directly from an Entropy Source

7.2.5 Functional Requirements for the Internal State

The internal state is the memory of the DRBG and consists of all of the parameters, variables and other stored values that the DRBG uses or acts upon. The internal state consists of both an administrative portion and a working portion. The administrative portion includes information about a DRBG instantiation; this information is used, for example, to select the correct internal state when multiple DRBG instantiations have been created. The working portion of the internal state includes information that is used or modified during pseudorandom bit generation, such as keys and data that is modified during each request. The type of data in the internal state is dependent on the specific DRBG and includes all information that is required to produce the pseudorandom bits from one request to the next. Some portion of the working portion of the internal state **shall** be changed by the internal state transition function during each iteration of the DRBG.

Comment [ebb3]: Page: 1
Not really sure if we really need to make the distinction between the administrative portion and the working portion. If we don't end up really using it, then it should be removed.

Part 1 of this Standard provides requirements on the internal state of a random bit generator. The requirements for the internal state of a RBG are:

1. *The internal state **shall** be protected in a manner that is consistent with the use and sensitivity of the output.*

A DRBG and its state(s) **shall** be contained within a DRBG boundary (see Section 8.2). The state **shall** be protected at least as well as the intended use of the output bits by the consuming application (see Section 8.3).

2. *The internal state **shall** be functionally maintained properly across power failures, reboots, etc. or regain a secure condition before any output is generated (i.e., either the integrity of the internal state **shall** be assured, or the internal state **shall** be re-initialized).*
   The fulfillment of this requirement is dependent on the physical embodiment of the DRBG and how it has been designed. When a DRBG is validated, documentation **shall** be provided and testing **shall** be conducted to establish that this requirement is fulfilled.

Comment [ebb4]: Page: 1
Reflect this in Section 11.2.

3. *The state elements that accumulate or carry entropy for the RBG **shall** have at least x* bits of entropy, *where x is the desired security level expressed in bits of security.*
   For DRBGs, the seed used to determine the internal state shall have entropy that is at least 128 bits or the desired security strength for the DFRBG, whichever is greater (i.e., *seed_entropy* ≥ **max** (128, *security_strength*)) (see Section 8.4).

4. *The secret portion of the internal state **shall** have a specified finite cryptoperiod, after which the RBG **shall** either cease operation or have sufficient additional entropy, as specified by the security level desired. This cryptoperiod may be large, depending on the design. Operations using an old secret portion of the internal state **shall** cease after the specified cryptoperiod elapses; note, however, that it is a good practice to combine the value of the old seed with the value of the new seed, so this specific activity is not prevented, rather it is encouraged. The maximum cryptoperiod to be used for a particular design **shall** be specified and documented; the actual cryptoperiod in use may be smaller than the maximum.*
   Cryptoperiods for each DRBG are discussed in Section 10. A capability for specifiying the maximum number of state updates has also been incorporated into each DRBG specification.

Comment [ebb5]: Page: 1
Need to make sure that this is actually true for each DRBG.

5. *A specific internal state **shall not** be <u>deliberately</u> reused, although this might occur by chance. In the case of a DRBG, this implies that the same seed **shall not** be deliberately used to seed multiple instantiations.*
   Section 8.4 states that a seed that is used to initialize one instantiation of a DRBG **shall not** be intentionally be used to reseed the same instantiation or as a seed for another DRBG instantiation.

An optional feature for the internal state of an RBG is:

1. *The internal states used to produce public data, such as nonces and initialization vectors, **should** be fully independent from the states used to produce secret data, such as cryptographic keys. There may be more separation of internal states to meet application requirements, such as separating the internal states for symmetric keys from asymmetric keys, signature keys from key establishment keys, MAC keys from data encryption keys, etc.*
   This Standard recommends, but does not require, different instantiations (i.e., seed separation) for different types of random data, including using different

seeds for the generation of public data than are used to generate secret data (see Section 8.4).

7.2.6 Functional Requirements for the Internal State Transition Function

The internal state transition function uses the internal state and one or more Approved algorithms to produce pseudorandom bits. During this process, the working portion of the internal state is altered. The algorithms used and the method of altering the internal state depends on the specific DRBG.
The DRBGs in this Standard have three separate state transition functions:

1. During the initial instantiation of the DRBG, seed material is obtained, and all initial input is determined. The initial input is used as all or part of the initial state of the DRBG.

2. Each request for pseudorandom bits produces the requested bits using the current internal state and determines a new internal state that is used for the next request of bits.
3. When an application determines that reseeding of the DRBG is required, a new instance is created by a reseeding function that obtains new seed material, combines it with the internal state values of the prior DRBG instance, and determines a new internal state for the next request for pseudorandom bits. By combining the new seed material with the internal state from the prior DRBG instance, the entropy available from the prior instance is not lost, but is enhanced by the entropy of the new seed material. See Section 8.3 for a discussion on DRBG instances.

Part 1 of this Standard provides requirements on the internal state transition function of a random bit generator. The requirements for the internal state transition functions of an RBG are:

1. *The deterministic elements of the internal state transition function* ***shall*** *be verifiable via known-answer testing during the startup and periodic health tests.*
   A DRBG module **shall** perform self-tests to ensure that the DRBG continues to function properly. Self tests of the DRBG functionality **shall** be performed when the DRBG module is powered up, on demand, at periodic intervals, and under various conditions (see Section 11.3).

2. *The internal state transition function* ***shall****, over time, depend on all the entropy carried by the internal state. That is, added entropy* ***shall*** *affect the internal state. This aspect of the design* ***shall*** *be documented.*
   This objective is fulfilled by the design of the DRBGs specified in Section 10.

3. *The internal state transition functions* ***shall*** *resist observation and analysis via power consumption, timing, radiation emissions, or other side channels as appropriate, depending on the access by an observer who could be an adversary. This aspect of the design* ***shall*** *be documented. For example, if an adversary would have access to the power consumption, then ways to address this concern need to be considered in the design and documented, but if an adversary would not have access to the power consumption, then this assumption* ***should*** *be stated in the design documentation. Note that timing information may leak across*

communication networks, while power usage and radiation fluctuation almost certainly will not.
Fulfillment of this requirement will depend on the embodiment of the DRBG, although when incorporated into a validated FIPS 140-2 cryptomodule, many of these concerns may be addressed. When an implementation is validated, documentation **shall** be provided that indicates how this requirement is satisfied; testing will be conducted to provide assurance that this requirement has been met (see Section 11.2).

Optional features of the internal state transistion function are:

1. *The Internal State Transition Function* ***may*** *enable the RBG to recover from the compromise of the internal state at a particular time through periodic incorporation of entropy appropriate for the desired security level. Note that an NRBG will always have this property.*
   DRBGs may be instantiated to provide prediction resistance, which requires the insertion of sufficient additional entropy to meet the desired security strength prior to generating pseudorandom bits (see Section 8.7). Prediction resistance will isolate prior internal states from the next internal state.

2. *It* ***should not*** *be feasible (either intentionally and unintentionally) to cause the Internal State Transition Function to return to a prior state in normal operation (this excludes testing and authorized verification of the RBG output), except possibly by chance (depending on the specific design)~~, unless dealing with a "grandfathered" design that is allowed only for compatibility purposes~~. This includes the requirement for re-initialization of a DRBG before a full cycle of values is output. Meeting this implies that the RBG has volatile memory.*

   This requirement is fulfilled by the design of each DRBG in Section 10 during one instance of the DRBG and requirements for reseeding at the end of the instance's seedlife.

7.2.7 Functional Requirements for the Output Generation Function

The output generation function of a DRBG produces pseudorandom bits that are a function of the working portion of the internal state of the DRBG and any input that is introduced while the internal state transition function is operating. These pseuodorandom bits output bits are deterministic with respect to the input information. Any formatting of the output bits prior to output is determined by a particular implementation.
Part 1 of this Standard provides requirements for the output generation function of a random bit generator. The functional requirements for the output generation function are:

1. *The output generation function* ***shall*** *be deterministic (given all inputs) and* ***shall*** *allow known-answer testing when requested.*
   Operational testing, as specified in Section 10.3, **shall** be performed on demand and **shall** include known-answer testing. If an implementation is validated, known-answer testing will be performed, and the implementation will be examined to ensure that known-answer testing will be performed during normal operations.

2. *The output **shall** be inhibited until the internal state exhibits/obtains sufficient assessed entropy.*
   Section 8.4 states that a DRBG **shall not** provide output until a seed is available. The entropy source for a seed **shall** provide the required amount of entropy for that seed in order to support the appropriate security level for a consuming application.
   Section 8.5 states that a DRBG requiring a key(s) **shall not** provide output until the key(s) is available. The entropy for the key(s) is provided in the seed.
   When an implementation is validated, documentation **shall** be provided, and testing will be conducted to establish that the DRBG does not provide output until sufficient entropy has been acquired in the seed.
3. *Once a particular internal state has been used for output, the internal state **shall** be changed in order to produce more output. This aspect of the design **shall** be documented.*
   The specifications in Section 10 for each DRBG include an update of the internal state prior to returning the requested pseudorandom bits to the consuming application.
4. *Test output from a known-answer test **shall** be separated from operational output (e.g., random output that is used for a cryptographic purpose).*
   Section 11.3.1 states that all data output from the DRBG module **shall** be inhibited while tests operational tests are performed. The results from known-answer tests **shall not** be output as random bits during normal operation.
5. *The output generation function **shall** protect the internal state, so that analysis of RBG outputs does not reveal useful information (from the point of view of compromise) about the internal state.*
   The DRBGs specified in Section 10 have been designed to fulfill this requirement. If an implementation is validated (see Section 11.2), testing will be conducted to provide assurance that the implementation conforms to the DRBG specification.

6. *The output generation function **shall** use information from the internal state that contains sufficient entropy to support the required security level. This aspect of the design **shall** be documented.*

   Providing that the seed used to initialize the DRBG contains the appropriate amount of entropy for the required security level, the output generation function in the DRBGs in this Standard have been designed to fulfill this requirement.

7. *The output generation function **shall** resist observation and analysis via power consumption, timing, radiation emissions, or other side channels as appropriate. This aspect of the design shall be documented.*
   This requirement depends on the embodiment of the DRBG. If an implementation is validated (see Section 11.2), design evidence **shall** be provided that will provide assurance that this requirement is fulfilled.

Comment [ebb6]: Page: 38
Or a different part of the state shall be used ?

7.2.8 Functional Requirements for Support Functions

The support functions for a DRBG are concerned with assessing and reacting to the health of the DRBG. The functional requirement for support functions in Part 1 is as follows.

*Any support functions in an RBG output **shall** be documented regarding their purpose and the principles used in their design.*

The required support funtions for DRBGs are discussed in this part of the Standard. Any additional support functions used by an implementation **shall** be documented as stated in the above requirement.

Additional support function requirements for DRBGs are as follows (see Section 11):

1. A DRBG **shall** be designed to permit testing that will ensure that the generator is correctly implemented and continues to operate correctly. A test function **shall** be available for this purpose. The test function **shall** also allow the insertion of predetermined values for the input information in order to test for expected results. If any test fails, the DRBG **shall** enter an error state and output an error indicator. The DRBG **shall not** perform any operations while in an error state. All output **shall** be inhibited when an error state exists.
2. Error states may include "hard" errors that indicate an equipment malfunction that may require maintenance, service, repair or replacement of the DRBG, or may include recoverable "soft" errors that may require re-instantiation of the DRBG. Recovery from error states **should** be possible except for those caused by hard errors that require maintenance, service, repair or replacement of the DRBG. [Editor's note: We probably need to include more advice on recovering from errors.]
3. Optional implementation validation is discussed in Section 11.2. Operational testing **shall** be implemented in accordance with the tests specified in Section 11.3.