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.

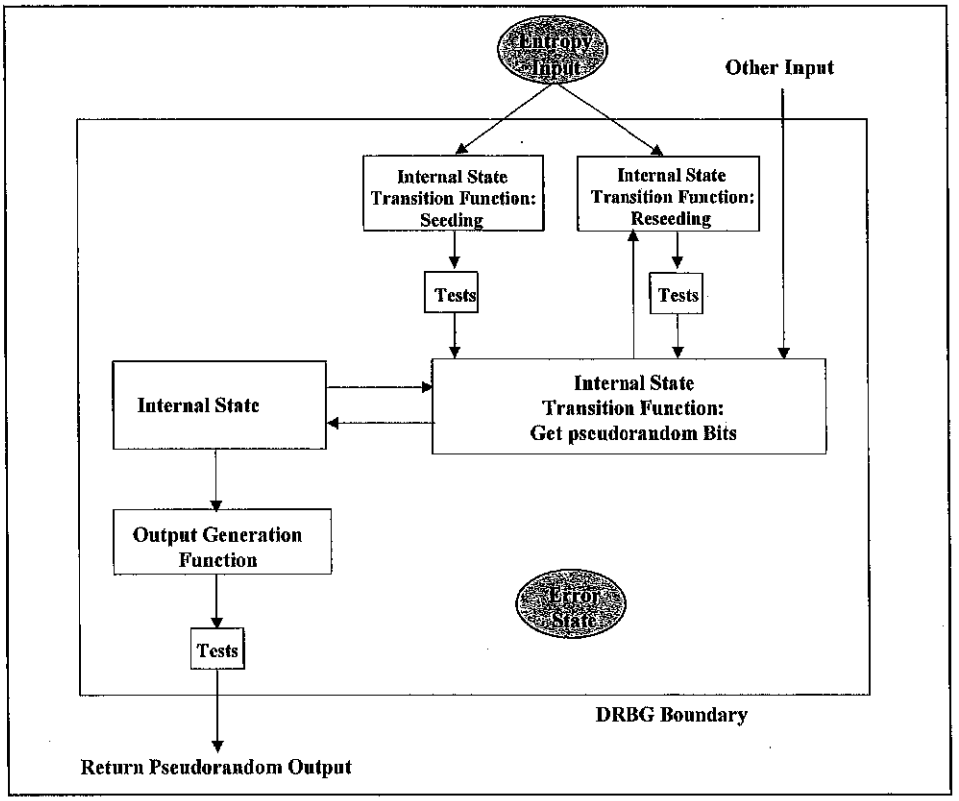

Figure 1: DRBG Model

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:

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.

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.
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.
5. *A specific internal state **shall not** be deliberately 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

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

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

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.

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.