## 8.3 DRBG Boundaries

As a convenience, this Standard uses the notion of a "DRBG boundary" to explain the operations of a DRBG and its interaction with and relation to other processes. The DRBG boundary is defined by the DRBG's public interfaces, which are made available to consuming applications.

Within a DRBG boundary,

1. The DRBG internal state and the operation of the DRBG functions **shall** only be affected according to the DRBG specification.
2. The DRBG internal state **shall** exist solely within the DRBG boundary.
3. Information about secret parts of the DRBG internal state and intermediate values in computations involving these secret parts **shall not** affect any information that leaves the DRBG boundary, except as specified for the DRBG pseudorandom bit outputs. The internal state **shall** be contained within the DRBG boundary and **shall not** be accessible from outside the boundary.

Each DRBG includes one or more cryptographic primitives (e.g., a hash function). Other applications may use the same cryptographic primitive as long as the DRBG's internal state and the DRBG functions are not affected.

A DRBG's functions may be contained within a single device, or may be distributed across multiple devices (see Figures 3 and 4). Figure 3 depicts a DRBG for which all functions are contained within the same device. In this case, there is a single DRBG boundary.

Figure 4 provides an example of DRBG functions that are distributed across multiple devices. In this case, each device has a DRBG boundary that contains the DRBG functions implemented on that device, and the "logical DRBG boundary" consists of the aggregation of boundaries providing the DRBG functionality. The use of distibuted DRBG boundaries may be convenient for restricted environments (e.g., smart card applications) in which the primary use of the DRBG does not require repeated use of the instantiation or reseeding functions.

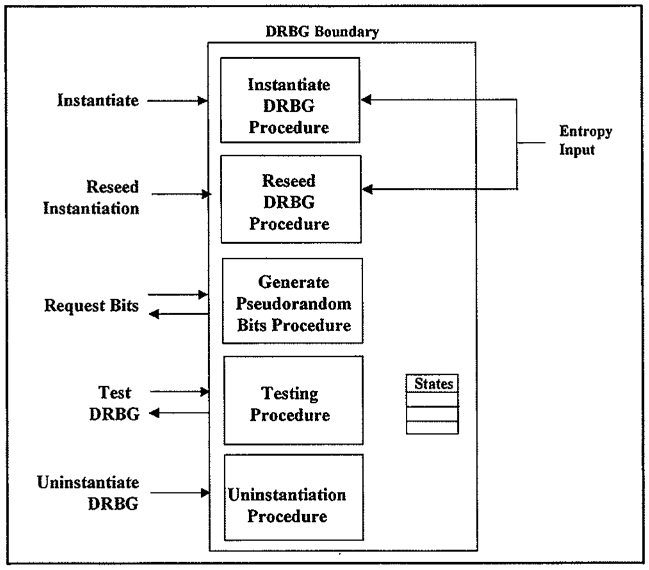

**Figure 3: DRBG Functions within a Single DRBG Boundary**

Each DRBG boundary **shall** contain a testing function to test the "health" of other DRBG functions within that boundary. Although the entropy input is shown in the figure as originating outside the DRBG boundary, it may originate from within the boundary. Part 4 discusses the construction of a full random bit generator that contains both the DRBG and its entropy input source.

Distributed DRBG boundaries **shall** be subject to the following:

1. Any DRBG boundary that includes an instantiate function **shall** include uninstantiate, generate and testing functions to allow health testing, although the generate function may not be the "primary" generate function for the DRBG. For example, for a smart card application, it may be necessary to distribute the DRBG functions so that the smart card contains only the generate function, along with its associated testing function. In this case, the instantiate function may reside on the system that initializes the smart card; the generate and uninstantiate functions are used on this system during the testing of the instantiate function.
2. A DRBG boundary containing a generate function **shall** include a testing function.
3. A DRBG boundary that contains a reseed function **shall** include generate and test functions to allow health testing, although the generate function may not be the "primary" generate function for the DRBG.

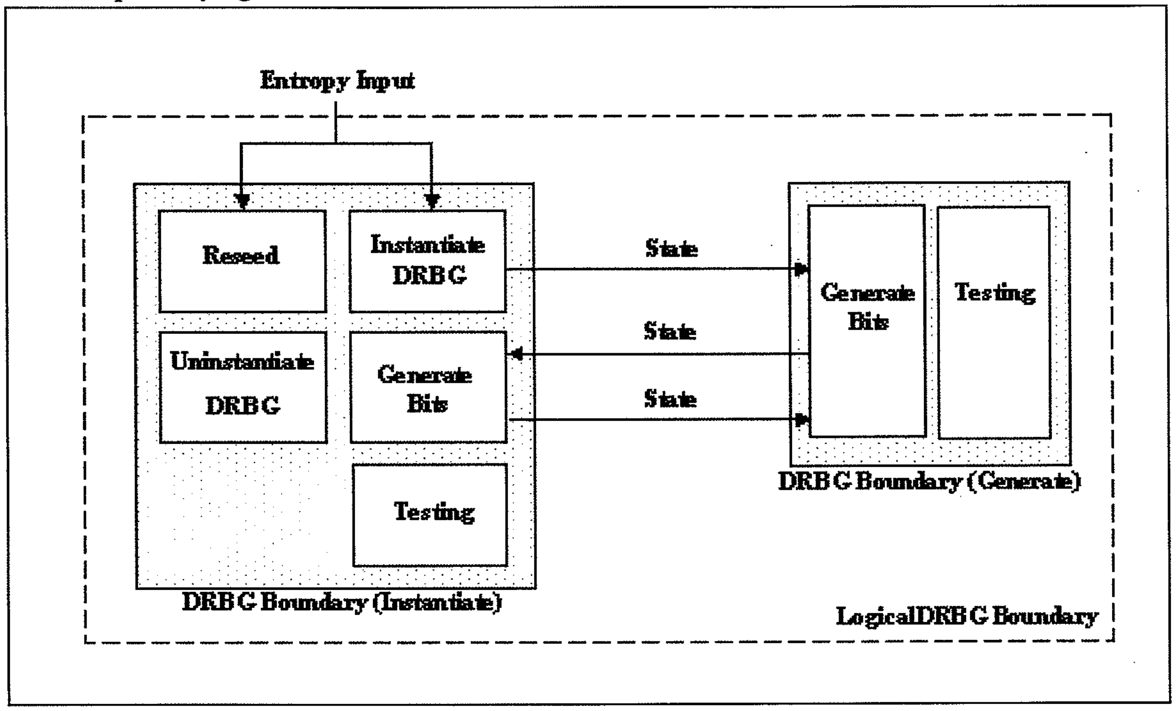

**Figure 4: Distributed DGBR Functions and Boundaries**

When DRBG functions are distributed, the DRBG functions are distributed among multiple DRBG boundaries, appropriate mechanisms **shall** be used to protect the confidentiality and integrity of the internal state when transferred between the distributed DRBG boundaries. The confidentiality and integrity mechanisms and security level **shall** be consistent with the data to be protected by the DRBG's consuming application (see SP 800-57).