The NIST SP 800-90 Deterministic Random Bit Generator Validation System (DRBGVS)

October 30, 2007

Timothy A. Hall

National Institute of Standards and Technology

Information Technology Laboratory

Computer Security Division

TABLE OF CONTENTS

1 Introduction

This document, *The NIST SP 800-90 Deterministic Random Bit Generator Validation System (DRBGVS),* specifies the procedures involved in validating implementations of the Deterministic Random Bit Generator mechanisms approved in NIST SP 800-90, *Recommendation for Random Number Generation Using Deterministic Random Bit Generators (Revised) (March 2007)* [1].

The *DRBGVS* is designed to perform automated testing on Implementations Under Test (IUTs). This document provides the basic design and configuration of the *DRBGVS*. It defines the purpose, the design philosophy, and the high-level description of the validation process for DRBG. The requirements and procedures to be followed by those seeking formal validation of an implementation of DRBG are presented. The requirements described include the specification of the data communicated between the IUT and the DRBGVS, the details of the tests that the IUT must pass for formal validation, and general instruction for interfacing with the DRBGVS.

2 Scope

This document specifies the tests required to validate IUTs for conformance to the NIST SP 800-90 [1]. When applied to IUTs that implement DRBG, the DRBGVS provides testing to determine the correctness of the algorithm contained in the implementation. The DRBGVS consists of a single test that determines if the DRBG implementation produces the expected random bit output given a set of entropy and other inputs.

3 Conformance

The successful completion of the tests contained within the DRBGVS is required to be validated as conforming to the DRBG. Testing for the cryptographic module in which the DRBG is implemented is defined in FIPS PUB 140-2, *Security Requirements for Cryptographic Modules*[2].

4 Definitions and Abbreviations

4.1 Definitions

DEFINITION	**MEANING**
CMT laboratory	Cryptographic Module Testing laboratory that operates the DRBGVS
Deterministic Random Bit Generator	The algorithms specified in NIST SP 800-90, *Recommendations for Random Number Generation Using Deterministic Random Bit Generators (DRBG)* for generating random bits.

4.2 Abbreviations

ABBREVIATION	MEANING
DRBG	Deterministic Random Bit Generator
DRBGVS	Deterministic Random Bit Generator Validation System
IUT	Implementation Under Test

5 Design Philosophy of the Deterministic Random Bit Generation Validation System

The DRBGVS is designed to test conformance to NIST SP 800-90 rather than provide a measure of a product's security. The validation tests are designed to assist in the detection of accidental implementation errors, and are not designed to detect intentional attempts to misrepresent conformance. Thus, validation should not be interpreted as an evaluation or endorsement of overall product security.

The DRBGVS has the following design philosophy:

1. The DRBGVS is designed to allow the testing of an IUT at locations remote to the DRBGVS. The DRBGVS and the IUT communicate data via *REQUEST (.req)* and *RESPONSE (.rsp)* files.

2. The testing performed within the DRBGVS uses statistical sampling (i.e., only a small number of the possible cases are tested); hence, the successful validation of a device does not imply 100% conformance with the standard.

6 DRBGVS Test

The DRBGVS for DRBG consists of a single test. The DRBGVS requires the vendor to select the mechanism or mechanisms, hash-based, block cipher, or dual elliptic curve mechanisms, and options (e.g., which SHA algorithm is used for hashing functions). Separate files will be generated for each mechanism.

6.1 Configuration Information

To initiate the validation process of the DRBGVS, a vendor submits an application to an accredited laboratory requesting the validation of its implementation of DRBG. The vendor's implementation is referred to as the Implementation Under Test (IUT). The request for validation includes background information describing the IUT along with information needed by the DRBGVS to perform the specific tests. More specifically, the request for validation includes:

1. Vendor Name;
2. Product Name;
3. Product Version;
4. Implementation in software, firmware, or hardware;
5. Processor and Operating System with which the IUT was tested if the IUT is implemented in software or firmware;
6. Brief description of the IUT or the product/product family in which the IUT is implemented by the vendor (2-3 sentences);
7. The DRBG mechanisms and options (e.g., SHA sizes, block cipher algorithms) supported by the IUT.

6.2 The Deterministic Random Bit Generator Test

The DRBGVS consists of a single test where the vendor is required to execute a series of functions or commands defined in NIST SP 800-90: `INSTANTIATE`, `GENERATE`, `RESEED`, and `UNINSTANTIATE`. CAVS generates a separate request (.req) file for each type of DRBG mechanism used; the Hash_DRBG (10.1.1) and HMAC_DRBG (10.1.2) mechanisms are treated together as hash-based mechanisms. An example of a hash-based mechanism sample file follows:

```
#  CAVS 6.0
#  "DRBG800-90" information for "aaadrbg800-90"
#  Generated on Thu Oct 18 15:26:10 2007


[Hash-based options: Hash SHA-1 :: Hash SHA-384 :: HMAC SHA-224]

[Hash SHA-1]

INSTANTIATE Hash-based with PREDICTION RESISTANCE ENABLED
Entropy input = f9d183efce3692605d518698ddb6bc45
Nonce = f4d870b3e9712109
Personalization string = <None>

GENERATE 80 bits -- PREDICTION RESISTANCE ENABLED
Additional input = <None>
Entropy input (for reseed) = 95f2345c9ff79a49d2d0525a8f2d87d0
Returned bits = ?

GENERATE 160 bits -- PREDICTION RESISTANCE ENABLED
Additional input = e721338d6df3e63074645a2468f4f418
Entropy input (for reseed) = e84f02a5df135df6ed163697466c419b
```

```
Returned bits = ?

GENERATE 240 bits -- PREDICTION RESISTANCE ENABLED
Additional input = <None>
Entropy input (for reseed) = 521ec92de73e7fa8cf8f29d27d0f2b83
Returned bits = ?

GENERATE 320 bits -- PREDICTION RESISTANCE ENABLED
Additional input = b5a7a35b9b017f0c77c5c036dbe924e8
Entropy input (for reseed) = 45446a9267ecd34ff31df1848c1cbd83
Returned bits = ?

UNINSTANTIATE


INSTANTIATE Hash-based with NO PREDICTION RESISTANCE
Entropy input = be3d2a8e819d5c5b96549f6863a72e87
Nonce = a2cc6ee336023336
Personalization string = 8c8d091bdf40baa7f6c5c9f861a65995

GENERATE 480 bits -- NO PREDICTION RESISTANCE
Additional input = <None>
Returned bits = ?

GENERATE 560 bits -- NO PREDICTION RESISTANCE
Additional input = 288933722fc873abdf62b8f2f6286463
Returned bits = ?

RESEED
Entropy input = 3a777680904d0fc269cb80b75ccd785f
Additional input = 0c15ac6ccf39cd128e4b6c2d471b5374

GENERATE 640 bits -- NO PREDICTION RESISTANCE
Additional input = <None>
Returned bits = ?

GENERATE 720 bits -- NO PREDICTION RESISTANCE
Additional input = b8a7b605d7fd56fca88885704ecd9358
Returned bits = ?

UNINSTANTIATE

.  .  . <other options follow>
```

Following the three-line header is a line in square brackets listing the options to be tested for this DRBG mechanism. In the example above, three options will be tested: Hash_DRBG with SHA-1, Hash_DRBG with SHA-384, and HMAC_DRBG with SHA-224. The next line in square brackets indicates that Hash_DRBG with SHA-1 is the option that immediately follows.

Next is a sequence of commands with inputs generated by CAVS (see below for the bit length of these inputs) and the corresponding values for the vendor to fill in. The four commands correspond to the four DRBG functions defined in NIST SP 800-90 Section 9, "DRBG Mechanism Functions." They are:

`INSTANTIATE`. For the instantiate function, CAVS supplies values for the entropy input, nonce, and, if non-null, the personalization string used to instantiate the DRBG instance. `<None>` indicates a zero-length bitstring, i.e., a Null value. The instantiate command also indicates whether or not prediction resistance is enabled or not. If it is, then all generate commands will be called with the prediction resistance flag = true. Nothing is required from the vendor for this command.

`RESEED`. The reseed function has two parameters. Entropy input is required and is supplied by CAVS. Additional input is optional, and the CMT lab specifies whether the implementation supports null (zero-length) additional input, non-null additional input, or both in the CAVS user interface. If no additional input is used, the line in the sample file reads `Additional input = <None>`, as shown above. If additional input is used, then CAVS supplies a value for it. Nothing is required from the vendor for this command.

`GENERATE`. This function is called requesting a certain number of returned bits that the vendor must provide. It also indicates whether prediction resistance is enabled or not and should always match what is indicated in the instantiate command for this instantiation. Additional input is an optional parameter; if the implementation supports both Null and non-Null additional inputs, then CAVS alternates between the two, as shown in the example above. Otherwise, the additional input is always Null or non-Null. For the generate command, the vendor is required to supply the returned bits in hexadecimal in place of the `?` is in `Returned bits = ?` in the example above.

`UNINSTANTIATE`. This function takes no input and produces no output. The DRBG instance should empty the internal state. CAVS does not check that the internal state has been emptied.

6.3 Input values

Prediction resistance: if an implementation supports prediction resistance, the CMT lab should check the prediction resistance box for each mechanism tested. If the prediction resistance supported box is checked, CAVS will generate test values with and without using prediction resistance. This is illustrated in the example file listed in 6.2. If the box is not checked, CAVS will only generate values without using prediction resistance.

Derivation function (df) for CTR_DRBG: counter-mode (CTR) block cipher mechanism DRBGs are defined in NIST SP 800-90 for use with a derivation function (df) and with no df. One or both of these options may be checked on the CTR_DRBG tab, whichever one(s) the implementation uses.

CAVS has default bit lengths for the inputs it provides. If the implementation can support these bit lengths, then do not change them. If an implementation does not support one of the defaults, the bit lengths can be edited by pushing the “Edit Input Lengths” button. The defaults and restrictions on each of the input lengths are as follows:

Entropy input: the default bit length is the maximum security strength supported by the mechanism/option. This is the minimum bit length CAVS will accept, as CAVS tests all DRBGs at their maximum supported security strength. Longer entropy inputs are permitted, with the following exception: for CTR_DRBG with no df, the bit length **must** equal the seed length.

Nonce: the default nonce bit length is one-half the maximum security strength supported by the mechanism/option. Longer nonces are permitted. CTR_DRBG with no df does not use a nonce.

Personalization string: CAVS has two default bit lengths for personalization string, 0 and maximum supported security strength, except in the case of CTR_DRBG with no df, where the second length must be <= seed length. If the implementation only supports one personalization string length, then set both numbers equal to each other. If the implementation does not use a personalization string, set both numbers to 0 (zero).

Additional input: the additional input bit lengths have the same defaults and restrictions as the personalization string lengths.

Appendix A References

[1] *Recommendation for Random Number Generation Using Deterministic Random Bit Generators (Revised)*, NIST SP 800-90, National Institute of Standards and Technology, March 2007.

[2] *Security Requirements for Cryptographic Modules*, FIPS Publication 140-2, National Institute of Standards and Technology, May 2001.