10.1.2 Hash Function DRBG using SHA-1 (SHA1_Hash_DRBG)

10.1.2.1 Discussion

This DRBG was originally specified in ANSI X9.30, Part 1. A seed is used to initialize the instance of a generator. Note that *XKey* (from ANSI X9.30, Part 1) is called *V* in this specification, and *XSEED* is now *additional_input* in order to provide clarity and naming consistency.
The **SHA1_Hash_DRBG (...) shall not** be used in new applications. However, it may be used for compatibility with old applications.

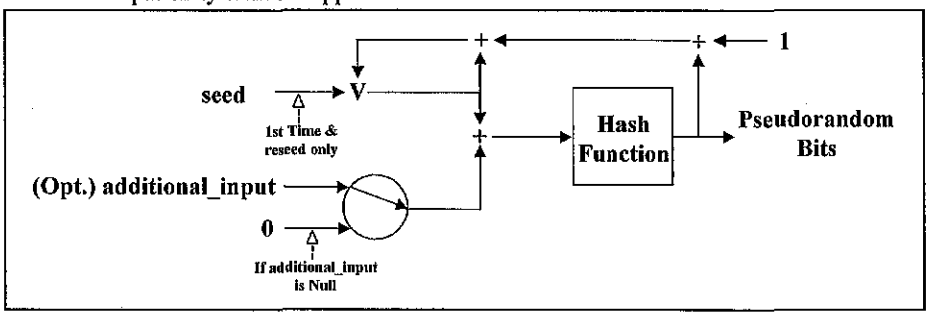

Figure 6: SHA1_Hash_DRBG (...)

Figure 6 depicts the **SHA1_Hash_DRBG (...)**. **SHA1_Hash_DRBG (...)** employs the SHA-1 hash function and produces a block of pseudorandom bits using a seed (*seed*) that determines the initial value of *V* during the first iteration of the algorithm; the *seed*'s role is thereafter performed by a function of the hash function's output. Additional input (*additional_input*) may be provided during each iteration of **SHA1_Hash_DRBG** (...); the size in bits of this input **should not** exceed the length of the seed, as the extra bits will be ignored.
This DRBG may be used by applications requiring 80 bits of security, requiring 128 bits of entropy for instantiation. The length of the *seed* (*seedlen*) for this DRBG **shall** be between 160 and 512 bits.

Figure 7 depicts the insertion of test input for the *seed* and the *additional_input*. The tests **shall** be run on the output of the generator. Validation and operational testing are discussed in Section 11. Detected errors **shall** result in a transition to the error state.
Note that the specifications for **SHA1_Hash_DRBG (...)** in the following sections do not specify a method for background reseeding (see Section 9.7) or the insertion of additonal entropy during the generation process except when pseudorandom bits are requested (see Section 9.8). Since the use of this DRBG is not allowed for new applications, but only only for compatibility with existing applications, these features have not been included.

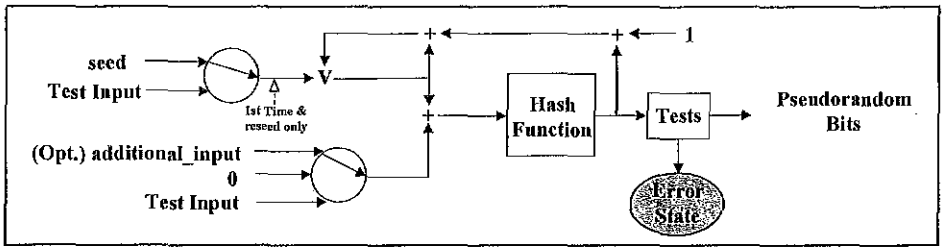

Figure 7: SHA1_Hash_DRBG (...) with Tests

10.1.2.2 Interaction with SHA1_Hash_DRBG (...)

10.1.2.2.1 Instantiating SHA1_Hash_DRBG (...)

Prior to the first request for pseudorandom bits, the **SHA1_Hash_DRBG (...) shall** be instantiated using the following call:

status = **Instantiate_SHA1_Hash_DRBG** ([*usage_class*], *requested_strength*, *prediction_resistance_flag*

as described in Section 9.6.1.

10.1.2.2.2 Reseeding a SHA1_Hash_DRBG (...) Instantiation

When a **SHA1_Hash_DRBG (...)** instantiation requires reseeding, the DRBG **shall** be re-instantiated (i.e., reseeded) using the following call:

status = **Reseed_SHA1_Hash_DRBG_Instantiation** ([*usage_class*]

as described in Section 9.7.2.

10.1.2.2.3 Generating Pseudorandom Bits Using SHA1_Hash_DRBG (...)

An application may request the generation of pseudorandom bits by **SHA1_Hash_DRBG (...)** using the following call:

(*status*, *pseudorandom_bits*) = **SHA1_Hash_DRBG** ([*usage_class*,] *requested_no_of_bits*, *requested_strength*, *additional_input_flag*)

as discussed in Section 9.8.2.

10.1.2.3 Specifications

10.1.2.3.1 General

The instantiation of **SHA1_Hash_DRBG (...)** consists of obtaining a *seed* with the appropriate amount of entropy, which is used to define the initial *state* of the DRBG. The *state* consists of:

1. (Optional) The *usage_class* for the DRBG instantiation (if the DRBG is used for multiple *usage_classes*, requiring multiple instantiations, then the *usage_class* parameter **shall** be present, and the implementation **shall** accommodate multiple

states simultaneously; if the DRBG will be used for only one *usage_class*, then the *usage_class* parameter **may** be omitted),

2. The value (*V*) that is updated during each call to the DRBG,
3. The initial value (*t*) for the hash function for the indicated *usage_class*,
4. The length of the *seed* (*seedlen*),
5. A *prediction_resistance_flag* that indicates whether or not prediction resistance is required by the DRBG,
6. A counter (*ctr*) that indicates the number of states that have been used by the DRBG instantiation, and
7. (Optional) A transformation of the *seed* using a one-way function for later comparison with a new *seed* when the DRBG is reseeded; this value **shall** be present if the DRBG will potentially be reseeded; it **may** be omitted if the DRBG will not be reseeded.

The variables used in the description of **SHA1_Hash_DRBG (...)** are:

additional_input	Optional additional input.
additional_input_flag	A flag that indicates whether or not additional input is to be requested (see Section 10.8.3); its values are as follows: 0 = Do not request *additional_input*. Set *additional_input* = 0. 1 = Request *additional_input*, but return 0 if no input is available.
ctr	A temporary counter value.
data	The data to be hashed.
Get_entropy (128, 160, 512)	A function that acquires a string of bits from an entropy source. 128 indicates the minimum amount of entropy to be provided in the returned bits; 160 indicates the minimum number of bits to be returned; 512 indicates the maximum number of bits to be returned. See Section 9.6.2.
Get_additional_input ( )	Returns a value for *additional_input*. This routine is left to the implementer. See Section 9.8.3.
i	A temporary value used as a loop counter.
m	The number of iterations of the hash function that are required to generate the requested number of pseudorandom bits.
M	The padded *data* to be hashed.
max_updates	The maximum number of updates of *V* for the DRBG.
old_seedlen	The *seedlen* from the previous instantiation.
old_transformed_seed	The *transformed_seed* from the previous instantiation
old_V	The value of *V* from the previous instantiation.
prediction_resistance_flag	A flag indicating whether or not prediction resistance is required by the instantiation. *prediction_resistance_flag* = 1 = yes, 0 = no.

pseudorandom_bits	The string of pseudorandom bits that are generated during a single "call" to the **SHA1_Hash_DRBG (...)** process.
requested_no_of_bits	The number of bits requested from the DRBG.
requested_strength	The requested security strength for the pseudorandom bits obtained from the DRBG.
returned_bits	The 160-bit value that is generated at each iteration of the hash function.
seed_material	The seed for this instance of the **SHA1_Hash_DRBG(...).**
seedlen	The length of the *seed.*
state	The state of **SHA1_Hash_DRBG (...)** that is carried between calls to the DRBG. In the following specifications, the entire state is defined as {[*usage_class,*] *V, t, seedlen, prediction_resistance_flag, ctr* [, *transformed_seed*]}. A particular element of the *state* is specified as *state.element*, e.g., *state.V.*
status	The *status* returned from a function call, where *status* = "Success" or an indication of a failure. Failure messages are: 1. Invalid *requested_strength.* 2. Failure indication returned by the entropy source. 3. State not available for the indicated *usage_class.* 4. Entropy source failure. 5. Invalid *additional_input_flag* value. 6. Failure from request for *additional_input.* 7. *additional_input* too large.
t	The initial value of the hash function. See Annex E.
temp	A temporary value.
transformed_seed	A one-way transformation of the *seed* for the **SHA1_Hash_DRBG(...)** instance.
usage_class	The purpose(s) of a DRBG instance.
V	A value that is initially derived from the *seed,* but assumes new values during subsequent calls to the **SHA1_Hash_DRBG (...)** process, based on the current value of *V* and the output of the hash function. The last value of *V* from one call to the function is the new value for the next call to the function.

10.1.2.3.2 Instantiation of SHA1_Hash_DRBG(...)

The following process or its equivalent **shall** be used to initially instantiate the **SHA1_Hash_DRBG (...)** process in Section 10.1.2.3.4:

Instantiate_SHA1_Hash_DRBG (...):

Input: integer ([*usage_class*], *requested_strength, prediction_resistance_flag.*

Output: string *status.*

Process:

1. If (*requested_strength* > 80), then **Return** ("Invalid *requested_strength*").
2. (*status, seed_material*) = **Get_entropy** (128, 160, 512).

3. If (*status* = "Failure"), then **Return** ("Failure indication returned by the entropy source").
4. *seedlen* = ||*seed_material*||.
5. (Optional) Get additional input and combine with the *seed_material*.
   5.1 (*status*, *additional_input*) = **Get_additional_input**( ).
   5.2 If (*status* = "Failure"), then **Return** ("Failure from request for additional input").
   5.3 *seed_material* = *seed_material* || *additional_input*.
   Comment: Perform a one-way function on the *seed* for later comparison during reseeding.
6. (Optional) *transformed_seed* = **SHA1** (*seed_material*).
7. Set up *t* for the indicated *usage_class*. Comment: See Annex E.
8. *ctr* = 1.
9. *V* = **SHA1_df** (*seed_material*, *seedlen*). Comment: Ensure that the entropy in the seed material is distributed throughout *V*. See Section 9.6.3.2.
10. *state* = {[*usage_class*,] *V*, *t*, *seedlen*, *prediction_resistance_flag*, *ctr* [, *transformed_seed*]}.
11. **Return** ("Success").

Note that multiple *state* storage is required if the DRBG is used for multiple *usage_classes*.

10.1.2.3.3 Reseeding a SHA1_Hash_DRBG(...) Instantiation

The following or an equivalent process **shall** be used to explicitly reseed the **SHA1_Hash_DRBG** (...) process:

Reseed_SHA1_Hash_DRBG_Instantiation (...):

Input: integer ([*usage_class*].

Output: string *status*.

Process:

1. If a *state* is not available for an indicated *usage_class*, then **Return** ("State not available for the indicated *usage_class*").
2. Get the appropriate *state* values for the indicated *usage_class*, e.g., *old_V* = *state.V*, *old_seedlen* = *state.seedlen*, *old_transformed_seed* = *state.transformed_seed*.
3. Perform steps 1 to 8 of **Instantiate_SHA1_Hash_DRBG (...)**.
   3.1 (*status*, *seed_material*) = **Get_entropy** (128, 160, 512).
   3.2 If (*status* = "Failure"), then **Return** ("Failure indication returned by the entropy source").
   3.3 *seedlen* = ||*seed_material*||.
   3.4 (Optional) Get additional input and combine with the *seed_material*.
      3.4.1 (*status*, *additional_input*) = **Get_additional_input**( ).
      3.4.2 If (*status* = "Failure"), then **Return** ("Failure from request for additional input").
      3.4.3 *seed_material* = *seed_material* || *additional_input*.
      Comment: Perform a one-way function on the *seed* for later comparison during reseeding.

3.5 (Optional) *transformed_seed* = **SHA1** (*seed_material*).
3.6 Set up *t* for the indicated *usage_class*. Comment: See Annex E.
3.7 *ctr* = 1.

4. If (*old_transformed_seed* = *transformed_seed*), then **Return** ("Entropy source failure").
5. *seedlen* = **max** (*seedlen*, *old_seedlen*). Determine the larger of the seed sizes so that entropy is not lost.
   Comment: Combine the *new_seed* with the current value of *V* to derive the new initial *V* (the new *seed*).
6. *V* = **SHA1_df** ((*old_V* || *seed_material*), *seedlen*).
7. Update the appropriate *state* values for the *usage_class*.
   7.1 *state.V* = *V*.
   7.2 *state.seedlen* = *seedlen*.
   7.3 *state.ctr* = *ctr*.
   7.4 *state.transformed_seed* = *transformed_seed*.
8. **Return** ("Success").

10.1.2.3.4 Generating Pseudorandom Bits Using SHA1_Hash_DRBG(...)

The following process or an equivalent **shall** be used to generate pseudorandom bits:

SHA1_Hash_DRBG(...):

Input: integer ([*usage_class*], *requested_no_of_bits*, *requested_strength*, *additional_input_flag*).

Output: string (*status*, *pseudorandom_bits*).

Process:

1. If ((*requested_strength* >80), then **Return** ("Invalid *requested_strength*", Null).
2. If ((*additional_input_flag* < 0) or (*additional_input_flag* > 1)), then **Return** ("Invalid *additional_input_flag* value", Null).
3. If a *state* for the indicated *usage_class* is not available, then **Return** ("State not available for the indicated *usage_class*", Null).
4. Get the appropriate *state* values in accordance with the indicated *usage_class*, e.g., *V* = *state.V*, *t* = *state.t*, *seedlen* = *state.seedlen*, *prediction_resistance_flag* = *state.prediction_resistance_flag*, *ctr* = *state.ctr*.
5. $m = \left\lceil \frac{requested\_no\_of\_bits}{160} \right\rceil$. Comment: Determine the number of loops needed to generate the requested number of bits.
6. *temp* = the Null string.
7. For *i* = 1 to *m* do:
   Comment: Get additional input in accordance with the *additional_input_flag*.
   7.1 If (*additional_input_flag* = 0), then *additional_input* = 0
   Else do
   7.1.1 (*status*, *additional_input*) = **Get_additional_input** ( ).

7.1.2 If (*status* = "Failure"), then **Return** ("Failure from request for *additional_input*", Null).

7.1.3 If (|| *additional_input* || > *seedlen*), then Return ("*additional_input* too large", Null).

7.2 *data* = (*V* + *additional_input*) mod $2^{seedlen}$.

7.3 *M* = *data* || $0^{512-seedlen}$. Comment: $0^{512-seedlen}$ is a string of (512 - *seedlen*) zero bits.

7.4 Execute the process specified in Section 6.1.2 of FIPS 180-2.

7.5 *returned_bits* = the result from step 7.4 (i.e., *returned_bits* = H_0 || H_1 || H_2 || H_3 || H_4).

7.6 *V* = (1 + *V* + *returned_bits*) mod $2^{seedlen}$.

7.7 *ctr* = *ctr* + 1.

7.8 *temp* = *temp* || *returned_bits*.

8. *pseudorandom_bits* = Leftmost (*requested_no_of_bits*) of (*temp*).

9. If ((*ctr* ≥ *max_updates*) or (*prediction_resistance_flag* = 1)), then

9.1 *status* = **Reseed_SHA1_Hash_DRBG_Instantiation** ([*usage_class*]).

9.2 If (*status* ≠ "Success"), then **Return** (*status*, Null).

Else Update the changed values in the *state*.

9.3 *state.V* = *V*.

9.4 *state.ctr* = *ctr*.

10. **Return** ("Success", *pseudorandom_bits*).

Comment [ebb1]: Page: 64
Is this OK, since the old version didn't count the number of states (i.e., updates) ?

Comment [ebb2]: Page: 64
Does this make any sense for this DRBG ?

10.1.2.3.5 Implementation Considerations

[To be added later]

10.1.2.4 Generator Strength and Attributes

[To be determined]

10.1.2.5 Reseeding

A new *seed* **shall** be generated to reseed the generator [How often? This will determine the value for max_updates].