10.1.4 HMAC_DRBG (...)

10.1.4.1 Discussion

HMAC_DRBG (...) uses several occurrences of an Approved keyed hash function and an Approved hash function.. The same Approved hash function **shall** be used throughout. The hash function used **shall** meet or exceed the security requirements of the consuming application. Table 1 in Section 10.1.1 specifies the entropy and seed length requirements that **shall** be used for each hash function in order to meet a specified security level.

HMAC_DRBG (...) is specified using an internal functions: **Update (...)**. This function is called during the instantiation, pseudorandom bit generation and reseeding processes to adjust the state when new entropy or additional input is provided.

10.1.4.2 Interaction with HMAC_DRBG (...)

10.1.4.2.1 Instantiating HMAC_DRBG (...)

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

(*status*, *usage_class*) = **Instantiate_Hash_DRBG** (*requested_strength*, *prediction_resistance_flag*, *personalization_string*, *mode*),

as described in Sections 9.6.1 and 10.1.4.3.3.

10.1.4.2.2 Reseeding a HMAC_DRBG (...) Instantiation

When a **HMAC_DRBG (...)** instantiation requires reseeding, the DRBG **shall** be reseeded using the following call:

status = **Reseed_HMAC_DRBG_Instantiation** (*usage_class*, *mode*)

as described in Sections 9.7.2 and 10.1.4.3.4.

10.1.4.2.3 Generating Pseudorandom Bits Using HMAC_DRBG (...)

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

(*status*, *pseudorandom_bits*) = **HMAC_DRBG** (*usage_class*, *requested_no_of_bits*, *requested_strength*, *additional_input*, *prediction_resistance_requested*, *mode*)

as discussed in Sections 9.8.2 and 10.1.4.3.5.

10.1.4.2.4 Removing an HMAC_DRBG (...) Instantiation

An application may request the removal of an **HMAC_DRBG (...)** instantiation using the following call:

status = **Uninstantiate_HMAC_DRBG** (*usage_class*)

as described in Sections 9.X.X and 10.1.4.3.6.

10.1.4.2.5 Self Testing of the HMAC_DRBG (...) Process

An **HMAC_DRBG (...)** implementation is tested at power-up and on demand using the following call:

status = **Self_Test_HMAC_DRBG** ( )

as described in Sections 9.9 and 10.1.4.3.7.

10.1.4.3 Specifications

10.1.4.3.1 General

The instantiation and reseeding of **HMAC_DRBG (...)** consists of obtaining a *seed* with the appropriate amount of entropy. The entropy input is used to derive a *seed*, which is then used to derive elements of the initial *state* of the DRBG. The *state* consists of:

1. The value *V*, which is updated each time another *outlen* bits of output are produced (where *outlen* is the number of output bits in the underlying hash function).
2. The value *K*, which are updated at least once each time the DRBG generates pseudorandom bits.
3. The security *strength* of the DRBG instantiation.
4. A counter (*ctr*) that indicates the number of updates of *V* since new *entropy_input* was obtained whose entropy meets or exceeds the entropy requirement for the security *strength*.
5. A *prediction_resistance_flag* that indicates whether or not a prediction resistance capability is required for the DRBG.
6. (Optional) A transformation of the entropy input using a one-way function for later comparison with new entropy input 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 **HMAC_DRBG (...)** are:

additional_input	Optional additional input.
ctr	A counter that records the number of times that the *state* has been updated since the DRBG instantiation was seeded, reseeded or prediction resistance was obtained.
entropy_input	The bits containing entropy that are used to determine the *seed_material*.
Find_state_space (*mode*)	A function that returns a *usage_class* indicating an available state space. The *mode* indicates whether the request is made during normal operation or during testing.

Get_entropy (*min_entropy*, *outlen*, 2^{32}, *mode*)

	A function that acquires a string of bits from an entropy input source. *min_entropy* indicates the minimum amount of entropy to be provided in the returned bits; *outlen* indicates the minimum number of bits to return; 2^{32} indicates the maximum number of bits that may be returned; *mode* is used to indicate whether the bits are to be obtained during normal operation or during testing. See Section 9.6.2.
K	A value in the state that is updated when the DRBG generates pseudorandom bits.
max_no_of_states	The maximum number of states and instantiations that an implementation can handle.
max_updates	The maximum number of *state* updates allowed for the DRBG instantiation from one seeding, reseeding or prediction resistance operation.
min_entropy	The minimum amount of entropy to be provided in the *entropy_input*.
mode	An indication of whether a process is to be conducted for normal operations or for testing. *mode* = 1 = *Normal_operation* indicates that normal operation is required; *mode* = 2 = *Fixed_1* indicates that a predetermined value is to be used during instantiation, *mode* = 3 = *Fixed_2* indicates that a predetermined value is to be used during reseeding, mode = 4 = *Failure* indicates that a failure indication is to be returned.
N	The number of bytes in the hash function output block.
old_transformed_entropy_input	The *transformed_entropy_input* from the previous acquisition of *entropy_input* (e.g., used during reseeding).
outlen	The number of bits in the hash function output block.
personalization_string	A string that may be used to personalize a DRBG instantiation.
prediction_resistance_flag	Indicates whether or not prediction resistance is to be provided upon request during an instantiation. 1 = *Allow_prediction_resistance*: requests for prediction resistance will be handled; 0 = *No_prediction_resistance*: requests for prediction resistance will return an error indication.
prediction_resistance_requested	Indicates whether or not prediction resistance is required during the actual generation of pseudorandom bits. 1 =

	Provide_prediction_resistance: prediction resistance required; 0 = *No_prediction_resistance*: prediction resistance not required.
pseudorandom_bits	The string of *pseudorandom_bits* that are generated during a single "call" to the **KHF_DRBG (...)** process.
requested_no_of_bits	The number of pseudorandom bits to be generated.
requested_strength	The security strength to be provided for the pseudorandom bits to be obtained from the DRBG.
seed_material	The data used as the *seed.*
state(*usage_class*)	An array of *states* for different DRBG instantiations. A *state* is carried between calls to the DRBG. In the following specifications, the state for a *usage_class* is defined as *state*(*usage_class*) = {*V*, K_0, K_1, *strength*, *ctr*, *prediction_resistance_flag*, *transformed_entropy_input*}. A particular element of the *state* is specified as *state*(*usage_class*).*element*; e.g., *state*(*usage_class*).*V*.
status	The status returned from a function call, where *status* = "Success" or an indication of failure. Failure messages are: 1. Invalid *requested_strength.* 2. Cannot support prediction resistance. 3. No available *state* space. 4. Failure indication returned by the *entropy_input* source. 5. State not available for the indicated *usage_class.* 6. *Entropy_input* source failure. 7. KHF_DRBG can no longer be used. Please re-instantiate or reseed. 8. Too many bits requested. 9. Prediction resistance capability not instantiated.
strength	The security strength provided by the DRBG instantiation.
temp	A temporary value.
transformed_entropy_input	A one-way transformation of the *entropy_input* for the DRBG.
usage_class	The usage class of a DRBG instantiation. Used as a pointer to an instantiation's *state* values.

V	A value in the *state* that is updated whenever pseudorandom bits are generated.

10.1.4.3.2 Internal Function : The Update Function

The **Update (...)** function updates the internal state of the HMAC_DRBG (...) using *seed_material*.

Update (...):

Input: string (*seed_material*, *K*, *V*).

Output: string (K, V).

Process:

1. *K* = **HMAC** (*K*, V || 0x00 || *seed_material*).
2. *V* = **HMAC** (*K*, *V*).
3. *K* = **HMAC** (*K*, *V* || 0x01 || *seed_material*).
4. *V* = **HMAC** (*K*, *V*).
5. *ctr* = *ctr* + 2.
6. **Return** (*K*, *V*).

10.1.4.3.3 Instantiation of HMAC_DRBG(...)

The following process or its equivalent **shall** be used to initially instantiate the **HMAC_DRBG (...)** process. Let **HMAC (...)** be the Approved keyed hash function that is based on an Approved hash function, and let **Hash (...)** be that hash function. Let *outlen* be the output length of the hash function in bits, and let *N* be the output length of the hash function in bytes.

Instantiate_HMAC_DRBG (...):

Input: integer (*requested_strength*, *prediction_resistance_flag*, *personalization_string*, *mode*).

Output: string *status,* integer *usage_class*.

Process:

1. If (*requested_strength* > the maximum security *strength* that can be provided by the hash function (see Table 1)), then **Return** ("Invalid *requested_strength*", 0).
2. If (*prediction_resistance_flag* = *Allow_prediction_resistance*) and prediction resistance cannot be supported, then **Return** ("Cannot support prediction resistance", 0).

   Comment: Find state space.
3. (*status*, *usage_class*) = **Find_state_space** (*mode*).
4. If (*status* = "Failure"), then **Return** ("No available state space", 0).

   Comment: Set the *strength* to one of

the five security strengths.

5. If (*requested_strength* ≤ 80), then *strength* = 80

   Else if (*requested_strength* ≤ 112), then *strength* = 112

   Else (*requested_strength* ≤ 128), then *strength* = 128

   Else (*requested_strength* ≤ 192), then *strength* = 192

   Else *strength* = 256.

   Comment: Get the *entropy_input*.

6. *min_entropy* = **max** (128, *strength*).
7. (*status*, *entropy_input*) = **Get_entropy** (*min_entropy*, *outlen*, 2^{32}, *mode*).
8. If (*status* = "Failure"), then **Return** ("Failure indication returned by the entropy source", 0).

   Comment: Perform a one-way function on the *entropy_input* for later comparison during reseeding.

9. *transformed_entropy_input* = **Hash** (*entropy_input*).
10. *seed_material* = *entropy_input* || *personalization_string*.
11. *K* = 0x00 00...00. Comment: *N* bytes of zeros.
12. *V* = 0x01 01...01. Comment: *N* bytes of ones.
13. *ctr* = 0.
14. (*K*, *V*) = **Update** (*seed_material*, *K*, *V*).
15. *state*(*usage_class*) = {*V*, *K*, *strength*, *ctr*, *prediction_resistance_flag*, *transformed_entropy_input*}.
16. **Return** ("Success", *usage_class*).

If an implementation does not handle all five security strengths, then step 5 must be modified accordingly.

If no *personalization_string* will ever be provided, then the *personalization_string* parameter in the input may be omitted, and step 10 becomes *seed_material* = *entropy_input*.

If an implementation will never be reseeded using the process specified in Section 10.1.4.3.3, then step 9 may be omitted, as well as the *transformed_entropy_input* in the *state* (see step 18).

If an implementation does not need the *prediction_resistance_flag* as a calling parameter (i.e., the **HMAC_DRBG (....)** routine in Section 10.1.2.3.4 either always or never acquires new entropy in step 7), then the *prediction_resistance_flag* in the calling parameters and in the *state* (see step 18) may be omitted, as well as omitting step 2.

10.1.4.3.4 Reseeding a HMAC_DRBG(...) Instantiation

The following or an equivalent process **shall** be used to explicitly reseed the **HMAC_DRBG (...)** process. Let **HMAC (...)** be the Approved keyed hash function that is based on an Approved hash function, and let **Hash (...)** be that hash function. Let *outlen* be the output length of the hash function in bits, and let *N* be the output length of the hash function in bytes.

Reseed_HMAC_DRBG_Instantiation (...):

Input: integer (*usage_class*, *mode*).

Output: string *status*.

Process:

1. If ((*usage_class* > *max_no_of_states*) or (*state* (*usage_class*)) = {Null, Null, 0, 0, 0, Null}), then **Return** ("State not available for the indicated *usage_class*").

   Comment: Get the appropriate *state* values for the indicated *usage_class.*

2. *V* = *state*(*usage_class*).*V*, *K* = *state*(*usage_class*).*K*, *strength* = *state*(*usage_class*).*strength*, *prediction_resistance_flag* = *state(usage_class).prediction_resistance_flag*, *old_transformed_entropy_input* = *state*(*usage_class*).*transformed_entropy_input*.

   Comment: Get the new *entropy_input.*

3. *min_entropy* = **max** (128, *strength*).
4. (*status*, *entropy input*) = **Get_entropy** (*min_entropy*, *outlen*, 2^{32}, *mode*).
5. If (*status* = "Failure"), then **Return** ("Failure indication returned by the *entropy_input* source").

   Comment: Compare the old *entropy_input* with the new *entropy_input.*

6. *transformed_entropy_input* = **Hash** (*entropy_input*).
7. If (*transformed_entropy_input* = *old_transformed_entropy_input*), then **Return** ("*Entropy_input* source failure").
8. *ctr* = 0.
9. (*K*, *V*) = **Update** (*seed_material*, *K*, *V*).
10. *state*(*usage_class*) = {*V*, K_0, K_1, *strength*, *ctr*, *prediction_resistance_flag*, *transformed_entropy_input*}.
11. **Return** ("Success").

10.1.4.3.5 Generating Pseudorandom Bits Using HMAC_DRBG(...)

The following process or an equivalent **shall** be used to generate pseudorandom bits. Let *outlen* be the output length of the hash function in bits, and let *N* be the output length of the hash function in bytes.

HMAC_DRBG(...):

Input: integer (*usage_class*, *requested_no_of_bits*, *requested_strength*, *additional_input*, *prediction_resistance_requested*, *mode*).

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

Process:

1. If ((*usage_class* > *max_no_of_states*) or (*state* (*usage_class*)) = {Null, Null, 0, 0, 0, Null}), then **Return** ("State not available for the indicated *usage_class*", Null).

   Comment: Get the appropriate *state* values for the indicated *usage_class*.

2. *V* = *state*(*usage_class*).*V*, *K* = *state*(*usage_class*).*K*, *strength* = *state*(*usage_class*).*strength*, *ctr* = *state*(*usage_class*).*ctr*, *prediction_resistance_flag* = *state*(*usage_class*).*prediction_resistance_flag*, *old_transformed_entropy_bits* = *state*(*usage_class*).*transformed_entropy_bits*.

   Comment: If *ctr* ≥ *max_updates*, then reseeding could not be done in step 14 (below) during the previous call because of no available entropy source.

3. If (*ctr* ≥ *max_updates*), then **Return** ("HMAC_DRBG can no longer be used. Please re-instantiate or reseed.", Null).

4. If (*requested_strength* > *strength*), then **Return** ("Invalid *requested_strength*", Null).

5. If (*requested_no_of_bits* > 2^{35}), then **Return** ("Too many bits requested", Null).

6. If ((*prediction_resistance_requested* = *Provide_prediction_resistance*) and (*prediction_resistance_flag* = *No_prediction_resistance*)), then **Return** ("Prediction resistance capability not instantiated", Null).

7. If (*prediction_resistance_requested* = *Provide_prediction_resistance*), then

   7.1 *min_entropy* = **max** (128, *strength*).

   7.2 (*status*, *entropy_bits*) = **Get_entropy** (*min_entropy*, *outlen*, 2^{32}, *mode*).

   7.3 If (*status* = "Failure"), then **Return** ("Failure indication returned by the *entropy_input* source", Null).

   7.4 *transformed_entropy_input* = **Hash** (*entropy_input*).

7.5 If (*transformed_entropy_input* = *old_transformed_entropy_input*), then **Return** ("Entropy_input source failure", Null).

7.6 *ctr* = 0.

Else

7.7 *entropy_input* = Null.

8. *seed_material* = *entropy_input* || *additional_input*.
9. If (*seed_material* ≠ Null), then (*K*, *V*) = **Update** (*seed_material*, *K*, *V*).
10. *temp* = Null.
11. While (**len** (*temp*) < *requested_no_of_bits*) do:

    11.1 *V* = **HMAC** (*K*, *V*).

    11.2 *temp* = *temp* || *V*.

12. *pseudorandom_bits* = Leftmost (*requested_no_of_bits*) of *temp*.
13. If (*seed_material* ≠ Null), then (*K*, *V*) = **Update** (*seed_material*, *K*, *V*)

    Else

    13.1 *K* = **HMAC** (*K*, *V* || 0x00).

    13.2 *V* = **HMAC** (*K*, *V*).

14. If (*ctr* ≥ *max_updates*), then

    14.1 *status* = **Reseed_HMAC_DRBG** (*usage_class*, *mode*).

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

    14.3 Go to step 16.

15. *state*(*usage_class*) = {*V*, *K*, *strength*, *ctr*, *prediction_resistance_flag*, *transformed_entropy_bits*).
16. **Return** ("Success", *pseudorandom_bits*).

If an implementation will never provide *additional_input*, then the *additional_input* input parameter may be omitted, and step 8 becomes *seed_material* = *entropy_input*.

If an implementation does not need the *prediction_resistance_flag*, then the *prediction_resistance_flag* may be omitted as an input parameter, and step 6 may be omitted. If prediction resistance is never used, then step 7 becomes *entropy_input* = Null.

If an implementation does not have a reseeding capability, then step 14 is omitted, and step 3 takes effect during the next call to the DRBG.

10.1.3.3.6 Removing a KHF_DRBG (...) Instantiation

The following or an equivalent process **shall** be used to remove a **HMAC_DRBG (...)** instantiation:

Uninstantiate_HMAC_DRBG (...):

Input: integer *usage_class*.

Output: string *status*.

Process:

1. If (*usage_class* > *max_no_of_states*), then **Return** ("Invalid *usage_class*").
2. *state*(*usage_class*) = {Null, Null, 0, 0, 0, Null}.
3. **Return** ("Success").

10.1.3.3.7 Self Testing of the HMAC_DRBG (...)

[To be added later]

10.1.3.4 Generator Strength and Attributes

10.1.3.5 Reseeding and Optional Input

Comment [ebb1]: Do we even need these sections any more?