10.1.2.3.5 Generating Pseudorandom Bits Using Hash_DRBG (...)

The following process or its equivalent **shall** be used to generate pseudorandom bits. Let **Hash (...)** be the Approved hash function to be used.

Hash_DRBG (...):

Input: integer (*state_handle*, *requested_no_of bits*, *requested_strength*, *additional_input, prediction_resistance_request_flag, mode*).

Output: string *status*, bitstring *pseudorandom_bits*.

Process:

1. If ((*state_handle* > *max_no_of_states*) or (*state* (*state_handle*) = {*Null*, *Null*, 0, 0, 0, 0, *Null*})), then **Return** ("State not available for the *state_handle*", *Null*).
2. Set up the *state* values, e.g., *V* = *state*(*state_handle*).*V*, *C* = *state*(*state_handle*).*C*, *reseed_counter* = *state*(*state_handle*).*reseed_counter*, *strength* = *state*(*state_handle*).*strength*, *seedlen* = *state*(*state_handle*).*seedlen*, *prediction_resistance_flag* = *state*(*state_handle*).*prediction_resistance_flag*, *old_transformed_entropy_input* = *state(state_handle).transformed_entropy_input*.

   Comment: If *reseed_counter* ≥ *reseed_interval*, then reseeding could not be done in step 12 (below) during the previous call.

3. If (*requested_no_of_bits* > *max_length*) then **Return** ("Too many bits requested", *Null*).
4. If (*requested_strength* > *strength*), then **Return** ("Invalid *requested_strength*", *Null*).
5. If ((*prediction_resistance_request_flag* = *Provide_prediction_resistance*) and (*prediction_resistance_flag* = *No_prediction_resistance*)), then **Return** ("Prediction resistance capability not instantiated", *Null*).
6. If ((*reseed_counter* ≥ *reseed_interval*) OR (*prediction_resistance_request_flag* = *Provide_prediction_resistance*)), then

   Comment: If a reseeding capability is not available).

   Return ("DRBG can no longer be used. Please re-instantiate or reseed", *Null*).

   Comment: If a reseeding capability is readily available.

   6.1 *status* = **Reseed_ Hash_DRBG_Instantiation** (*state_handle, additional_input, mode*).

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

   6.3 *additional_input* = *Null*.

7. If (*additional_input* ≠ *Null*), then do

   7.1 *w* = **Hash** (0x02 || *V* || *additional_input*).

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

8. *pseudorandom_bits* = **Hashgen** (*requested_no_of_bits*, *V*).
9. *H* = **Hash** (0x03 || *V*).
10. *V* = (*V* + *H* + *C* + *reseed_counter*) mod $2^{seedlen}$.

11. *reseed_counter* = *reseed_counter* + 1.
12. Update the changed values in the *state*.
    12.1 *state*(*state_handle*).*V* = *V*.
    12.2 *state*(*state_handle*).*reseed_counter* = *reseed_counter*.
13. **Return** ("Success", *pseudorandom_bits*).

Hashgen (...):

Input: integer *requested_no_of_bits*, bitstring *V*.

Output: bitstring *pseudorandom_bits*.

Process:

1. $m = \left\lceil \frac{requested\_no\_of\_bits}{outlen} \right\rceil$.
2. *data* = *V*.
3. *W* = the Null string.
4. For *i* = 1 to *m*
   4.1 w_i = **Hash** (*data*).
   4.2 *W* = *W* || w_i.
   4.3 *data* = *data* + 1.
5. *pseudorandom_bits* = Leftmost (*requested_no_of_bits*) bits of *W*.
6. **Return** (*pseudorandom_bits*).

If an implementation will never use *additional_input*, then the additional_input input parameter may be omitted, and steps 6.3 and 7 may be omitted.
If an implementation does not need the *prediction_resistance_flag*, then the *prediction_resistance_flag* and steps 5 may be omitted..
If an implementation does not have a reseeding capability, then step 6 does not need steps 6.1 - 6.3.

10.1.3.3.6 Generating Pseudorandom Bits Using KHF_DRBG (...)

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

KHF_DRBG(...):

Input: integer (*state_handle*, *requested_no_of_bits*, *requested_strength*, *additional_input*, *prediction_resistance_request_flag*, *mode*).

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

Process:

1. If ((*state_handle* > *max_no_of_states*) or (*state* (*state_handle*)) = {*Null*, *Null*, *Null*, 0, 0, 0, *Null*}), then **Return** ("State not available for the indicated *state_handle*", *Null*).
2. If (**len** (*additional_input* > *max_additional_input_length*), then **Return** ("*additional_input* too long").

   Comment: Get the appropriate *state* values.
3. *V* = *state*(*state_handle*).*V*, K_0 = *state*(*state_handle*).K_0, K_1 = *state*(*state_handle*).K_1, *strength* = *state*(*state_handle*).*strength*, *reseed_counter* = *state* (*state_handle*).*reseed_counter*, *prediction_resistance_flag* = *state* (*state_handle*).*prediction_resistance_flag*, *old_transformed_entropy_bits* = *state*(*state_handle*).*transformed_entropy_bits*.

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

4. If (*requested_strength* > *strength*), then **Return** ("Invalid *requested_strength*", *Null*).
5. If (*requested_no_of_bits* > *max_length*), then **Return** ("Too many bits requested", *Null*).
6. If ((*prediction_resistance_request_flag* = *Provide_prediction_resistance*) and (*prediction_resistance_flag* = *No_prediction_resistance*)), then **Return** ("Prediction resistance capability not instantiated", *Null*).
7. If ((*reseed_counter* ≥ *reseed_interval*) OR (*prediction_resistance_request_flag* = *Provide_prediction_resistance*)), then

   Comment: If reseeding is not available.

   Return ("DRBG can no longer be used. Please re-instantiate or reseed.", *Null*).

   Comment: If reseeding is readily available.

   7.1 *status* = **Reseed_KHF_DRBG** (*state_handle*, *additional_input*, *mode*).

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

   Else

   7.4 If (*additional_input* ≠ *Null*), then (K_0, K_1, V) = **Update** (*additional_input*, K_0, K_1, V).
8. *temp* = *Null*.
9. While (**len** (*temp*) < *requested_no_of_bits*) do:

   9.1 V = **KHF** (K_0, K_1, V).

   9.2 *temp* = *temp* || V.
10. *pseudorandom_bits* = Leftmost (*requested_no_of_bits*) of *temp*.
11. (K_0, K_1, V) = **Update** (*additional_input*, K_0, K_1, V).
12. *reseed_counter* = *reseed_counter* + 1.
13. *state*(*state_handle*) = {V, K_0, K_1, *strength*, *reseed_counter*, *prediction_resistance_flag*, *transformed_entropy_bits*).
14. **Return** ("Success", *pseudorandom_bits*).

If an implementation will never provide *additional_input*, then the *additional_input* input parameter, and steps 7.3 and 7.4 may be omitted, and a *Null* string replaces the *additional_input* in step 7.1 .

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 an implementation does not have a reseeding capability, then steps 7.1-7.3 may be omitted.

10.1.4.3.6 Generating Pseudorandom Bits Using HMAC_DRBG(...)

The following process or an equivalent **shall** be used to generate pseudorandom bits. Let **HMAC (...)** be an Approved HMAC function.

HMAC_DRBG(...):

Input: integer (*state_handle*, *requested_no_of_bits*, *requested_strength*, *additional_input*, *prediction_resistance_request_flag*, *mode*).

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

Process:

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

   Comment: Get the appropriate *state* values.

2. *V* = *state*(*state_handle*).*V*, *K* = *state*(*state_handle*).*K*, *strength* = *state*(*state_handle*).*strength*, *reseed_counter* = *state*(*state_handle*).*reseed_counter*, *prediction_resistance_flag* = *state*(*state_handle*).*prediction_resistance_flag*, *old_transformed_entropy_bits* = *state state_handle*).*transformed_entropy_bits*.
3. If (*requested_strength* > *strength*), then **Return** ("Invalid *requested_strength*", *Null*).
4. If (**len** (*additional_input*) > *max_additional_input_length*), then **Return**("*additional_input* too long.", *Null*)
5. If (*requested_no_of_bits* > *max_length*), then **Return** ("Too many bits requested", *Null*).
6. If ((*prediction_resistance_request_flag* = *Provide_prediction_resistance*) and (*prediction_resistance_flag* = *No_prediction_resistance*)), then **Return** ("Prediction resistance capability not instantiated", *Null*).
7. If ((reseed_counter ≥ reseed_interval) OR (*prediction_resistance_request_flag* = *Provide_prediction_resistance*)), then

   Comment: If reseeding is not available.

   Return ("DRBG can no longer be used. Please re-instantiate or reseed.", *Null*).

   Comment: If reseeding is readily available.

   7.1 *status* = **Reseed_HMAC_DRBG** (*state_handle*, *additional_input*, *mode*).

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

   7.3 *additional_input* = *Null*.

   Else

   7.4 If (*additional_input* ≠ Null), then (*K*, *V*) = **Update** (*additional_input*, *K*, *V*).
8. *temp* = *Null*.
9. While (**len** (*temp*) < *requested_no_of_bits*) do:

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

   9.2 *temp* = *temp* || *V*.
10. *pseudorandom_bits* = Leftmost (*requested_no_of_bits*) of *temp*.
11. (*K*, *V*) = **Update** (*seed_material*,*K*,*V*).
12. *reseed_counter* = *reseed_counter* + 1.

13. *state*(*state_handle*) = {*V*, *K*, *strength*, *reseed_counter*, *prediction_resistance_flag*, *transformed_entropy_bits*).

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

If an implementation will never provide *additional_input*, then the *additional_input* input parameter, and steps 7.3 and 7.4 may be omitted, and a *Null* string replaces the *additional_input* in step 7.1 .
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 an implementation does not have a reseeding capability, then steps 7.1-7.3 may be omitted.