10.1.3 KHF_DRBG

10.1.3.1 Discussion

KHF_DRBG (...) specifies multiple uses of a 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.

KHF_DRBG (...) is specified using two internal functions: **KHF (...)** and **Update (...)**. Both are called during the instantiation, pseudorandom bit generation and reseeding processes to adjust the state.

10.1.3.2 Interaction with KHF_DRBG

10.1.3.2.1 Instantiating KHF_DRBG (...)

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

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

as described in Sections 9.6.1 and 10.1.3.3.3.

10.1.3.2.2 Reseeding a KHF_DRBG (...) Instantiation

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

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

as described in Sections 9.7.2 and 10.1.3.3.4.

10.1.3.2.3 Generating Pseudorandom Bits Using KHF_DRBG (...)

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

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

as discussed in Sections 9.8.2 10.1:3.3.5.

10.1.3.2.4 Removing a KHF_DRBG (...) Instantiation

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

status = **Uninstantiate_KHF_DRBG** (*usage_class*)

as described in Sections 9.X.X and 10.1.3.3.6.

10.1.3.2.5 Self Testing of the KHF_DRBG (...) Process

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

status = **Self_Test_KHF_DRBG** ( )

as described in Sections 9.9 and 10.1.3.3.7.

10.1.3.3 Specifications

10.1.3.3.1 General

The instantiation and reseeding of **KHF_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*. 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 from the underlying hash function).
2. The values K_0 and K_1, 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 times that pseudorandom bits were generated since the DRBG instantiation was seeded, reseeded or prediction resistance was obtained.
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 or prediction resistance is requested; this value **shall** be present if the DRBG will potentially be reseeded or a prediction resistance capability is required for the instantiation.

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

additional_input — Optional additional input.

ctr — A counter that records the number of times that pseudorandom bits were generated 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^{35}, *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^{35} 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_0, K_1	Values in the state that are updated when the DRBG generates pseudorandom bits.
len(string)	A function returning the number of bits in a string.
M	The number of bytes in the hash function input block.
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. Note that the *mode* = 2 fixed values **shall** be different than the *mode* = 3 fixed values.
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.
*Pad*_0, *Pad*_1	Zero padding used by the **KHF (...)** function.
*Padded*_K_0	K_0 padded with zeros to create *M* bytes.
*Padded*_*V*	*V* padded with zeros to create $M - 9$ bytes.
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. *personalization_string* too long.
4. No available *state* space.
5. Failure indication returned by the *entropy_input* source.
6. State not available for the indicated *usage_class*.
7. *Entropy_input* source failure.
8. HMAC_DRBG can no longer be used. Please re-instantiate or reseed.
9. *additional_input* too long
10. Too many bits requested.

11. 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.

10.1.3.3.2 Internal Functions

10.1.3.3.2.1 The KHF Function

The KHF (...) function is used as a compression function and to distribute the effect of the bits in the input values across the entire output string. Let *N* be the number of bytes of output from the hash function, and *M* be the number of bytes of the hash function input block.

KHF(...):

Input: string (K_0, K_1, V).

Output: string V.

Process:

```
1. $Pad\_0$ = 0x00 00...00.                    Comment: $M - N$ bytes of zeros.
2. $Pad\_1$ = 0x00 00...00.                    Comment: $M - N - 9$ bytes of zeros.
3. $Padded\_K_0 = K_0 \| Pad\_0$.              Comment: Since $K_0$ is $N$ bytes in
                                               length, $Padded\_K_0$ is $M$ bytes long.
4. $Padded\_V = V \| Pad\_1$.                  Comment: Since $V$ is $N$ bytes in
                                               length, $Padded\_V$ is $M$-9 bytes long.
5. $temp = Padded\_V \oplus K_1$.
6. $V$ = Hash ($Padded\_K_0 \| temp$).
7. Return ($V$).
```

10.1.3.3.2.2 The Update Function

The **Update (...)** function updates the internal *state* of the **KHF_DRBG (...)** using the *seed_material*. The *seed_material* can be any input string of 2^{35} bits or less, including the Null string. **Update (...)** makes extensive use of both the **KHF (...)** and the **hash_df (...)** functions described in Sections 10.1.3.3.2.1 and 9.6.4.2, respectively. Let *N* be the output length of the hash function in bytes, and let *M* be the hash function input block size in bytes.

Update (...):

Input: string (*seed_material*, K_0, K_1, V).

Output: string (K_0, K_1, V).

Process:

```
1. temp = the Null string.
2. While ((8 × len (temp)) < $N + M - 9$) do:
    2.1 $V$ = KHF ($K_0$, $K_1$, $V$).
    2.2 temp = temp || $V$.
3. temp = The rightmost (least significant) $N$+$M$-9 bytes of temp.
4. temp = temp ⊕ hash_df (seed_material, $8 \times (N + M - 9)$).
5. $K_0$ = The rightmost $N$ bytes of temp.
6. $K_1$ = The leftmost $M$-9 bytes of temp.
7. $V$ = KHF ($K_0$, $K_1$, $V$).
8. Return ($K_0$, $K_1$, $V$).
```

10.1.3.3.3 Instantiation of KHF_DRBG(...)

The following process or its equivalent **shall** be used to initially instantiate the **KHF_DRBG (...)** process. Let **Hash (...)** be the Approved hash function to be used. Let *outlen* be the output length of that hash function in bits, and let N be the output length of the hash function in bytes. Let M be the hash function input block size in bytes.

Instantiate_KHF_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).
3. If (len(personalization_string)>$2^{35}$), then Return("personalization_string too
   long.")
                                        Comment: Find state space.
4. (status, usage_class) = Find_state_space (mode).
5. If (status = "Failure"), then Return ("No available state space", 0).
                                        Comment: Set the strength to one of
                                        the five security strengths.
6. 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.

7.  min_entropy = max (128, strength).
8.  (status, entropy_input) = Get_entropy (min_entropy, outlen, $2^{35}$, mode).
9.  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.

10. transformed_entropy_input = Hash (entropy_input).

                                          Comment: Set up the working
                                          values.

11. $K_0$ = 0x00 00...00.                 Comment: N bytes of zeroes.
12. $K_1$ = 0x01 01...01.                 Comment: M - 9 bytes of ones.
13. V = 0x02 02...02.                     Comment: N bytes of twos.
14. seed_material = entropy_input || personalization_string.
15. ctr = 0.
16. ($K_0$, $K_1$, V) = Update (seed_material, $K_0$, $K_1$, V).

                                          Comment: Set up the state.

17. state (usage_class) = {V, $K_0$, $K_1$, strength, ctr, prediction_resistance_flag,
    transformed_entropy_input}.
18. Return (“Success”, usage_class).
```

If an implementation does not handle all five security strengths, then step 6 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 13 becomes *seed_material* = *entropy_input*.

If an implementation will never be reseeded using the process specified in Section 10.1.3.3.4, then step 10 may be omitted, as well as the *transformed_entropy_input* in the *state* (see step 17).

If an implementation does not need the *prediction_resistance_flag* as a calling parameter (i.e., the **KHF_DRBG (....)** routine in Section 10.1.3.3.5 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 17) may be omitted, as well as omitting step 2.

10.1.3.3.4 Reseeding a KHF_DRBG(...) Instantiation

The following or an equivalent process **shall** be used to explicitly reseed the **KHF_DRBG (...)** process. Let **Hash (...)** be the Approved hash function to be used; let *outlen* be the output length of that hash function in bits, and let *N* be the output length of the hash function in bytes. Let *M* be the hash function input block size in bytes.

Reseed_KHF_DRBG_Instantiation (...):

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

Output: string *status*.

Process:

1. If ((*usage_class* > *max_no_of_states*) or (*state*(*usage_class*) = {Null, 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_0 = *state*(*usage_class*).K_0, K_1 = *state*(*usage_class*).K_1, *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^{35}, *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").

   Comment: Set up the new working values.

8. *ctr* = 0.
9. (K_0, K_1, *V*) = **Update** (*entropy_input*, K_0, K_1, *V*).

   Comment: Set the state values.

10. *state*(*usage_class*) = {*V*, K_0, K_1, *strength*, *ctr*, *prediction_resistance_flag*, *transformed_entropy_input*}.

```
    10. Return ("Success").
```

10.1.3.3.5 Generating Pseudorandom Bits Using KHF_DRBG (...)

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

KHF_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,
       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, $K_0$ = state(usage_class).$K_0$, $K_1$ =
       state(usage_class).$K_1$, 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 (requested_strength > strength), then Return ("Invalid
       requested_strength", Null).

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

    5. If(len(additional_input)>$2^{35}$), then Return("additional_input too long.")

    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, $2^{35}$,
             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 ($K_0$, $K_1$, V) = Update (seed_material, $K_0$, $K_1$,
   V).
10. If (ctr ≥ max_updates), then
        10.1 status = Reseed_KHF_DRBG (usage_class, mode).
        10.2 If (status ≠ "Success"), then Return (status, Null).
11. temp = Null.
12. While (len (temp) < requested_no_of_bits) do:
    12.1 V = KHF ($K_0$, $K_1$, V).
    12.2 temp = temp || V.
13. pseudorandom_bits = Leftmost (requested_no_of_bits) of temp.
14. ($K_0$, $K_1$, V) = Update (seed_material, $K_0$, $K_1$, V).
15. ctr = ctr + 1
16. state(usage_class) = {V, $K_0$, $K_1$, strength, ctr, prediction_resistance_flag,
    transformed_entropy_bits).
17. 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 10 **shall** be replaced by the following:

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

10.1.3.3.6 Removing a KHF_DRBG (...) Instantiation

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

Uninstantiate_KHF_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, Null, 0, 0, 0, Null}.
3. Return ("Success").
```

10.1.3.3.7 Self Testing of the KHF_DRBG (...)

[To be added later]

10.1.3.4 Generator Strength and Attributes

10.1.3.5 Reseeding and Optional Input