NAME

EVP_KDF−KB − The Key−Based EVP_KDF implementation

DESCRIPTION

The EVP_KDF−KB algorithm implements the Key-Based key derivation function (KBKDF). KBKDF derives a key from repeated application of a keyed MAC to an input secret (and other optional values).

Identity

''KBKDF'' is the name for this implementation; it can be used with the **EVP_KDF_fetch()** function.

Supported parameters

The supported parameters are:

''mode'' (**OSSL_KDF_PARAM_MODE**) <UTF8 string>
: The mode parameter determines which flavor of KBKDF to use − currently the choices are ''counter'' and ''feedback''. ''counter'' is the default, and will be used if unspecified.

''mac'' (**OSSL_KDF_PARAM_MAC**) <UTF8 string>
: The value is either CMAC or HMAC.

''digest'' (**OSSL_KDF_PARAM_DIGEST**) <UTF8 string>
''cipher'' (**OSSL_KDF_PARAM_CIPHER**) <UTF8 string>
''properties'' (**OSSL_KDF_PARAM_PROPERTIES**) <UTF8 string>
''key'' (**OSSL_KDF_PARAM_KEY**) <octet string>
''salt'' (**OSSL_KDF_PARAM_SALT**) <octet string>
"info (**OSSL_KDF_PARAM_INFO**) <octet string>
''seed'' (**OSSL_KDF_PARAM_SEED**) <octet string>
: The seed parameter is unused in counter mode.

''use-l'' (**OSSL_KDF_PARAM_KBKDF_USE_L**) <integer>
: Set to **0** to disable use of the optional Fixed Input data 'L' (see SP800−108). The default value of **1** will be used if unspecified.

''use-separator'' (**OSSL_KDF_PARAM_KBKDF_USE_SEPARATOR**) <integer>
: Set to **0** to disable use of the optional Fixed Input data 'zero separator' (see SP800−108) that is placed between the Label and Context. The default value of **1** will be used if unspecified.

Depending on whether mac is CMAC or HMAC, either digest or cipher is required (respectively) and the other is unused.

The parameters key, salt, info, and seed correspond to KI, Label, Context, and IV (respectively) in SP800−108. As in that document, salt, info, and seed are optional and may be omitted.

''mac'', ''digest'', cipher'' and ''properties" are described in ''PARAMETERS'' in **EVP_KDF** (3).

NOTES

A context for KBKDF can be obtained by calling:

```
EVP_KDF *kdf = EVP_KDF_fetch(NULL, "KBKDF", NULL);
EVP_KDF_CTX *kctx = EVP_KDF_CTX_new(kdf);
```

The output length of an KBKDF is specified via the `keylen` parameter to the **EVP_KDF_derive** (3) function.

Note that currently OpenSSL only implements counter and feedback modes. Other variants may be supported in the future.

EXAMPLES

This example derives 10 bytes using COUNTER−HMAC−SHA256, with KI ''secret'', Label ''label'', and Context ''context''.

```
EVP_KDF *kdf;
EVP_KDF_CTX *kctx;
unsigned char out[10];
OSSL_PARAM params[6], *p = params;

kdf = EVP_KDF_fetch(NULL, "KBKDF", NULL);
kctx = EVP_KDF_CTX_new(kdf);
EVP_KDF_free(kdf);

*p++ = OSSL_PARAM_construct_utf8_string(OSSL_KDF_PARAM_DIGEST,
                                        "SHA2-256", 0);
*p++ = OSSL_PARAM_construct_utf8_string(OSSL_KDF_PARAM_MAC,
                                        "HMAC", 0);
*p++ = OSSL_PARAM_construct_octet_string(OSSL_KDF_PARAM_KEY,
                                         "secret", strlen("secret"));
*p++ = OSSL_PARAM_construct_octet_string(OSSL_KDF_PARAM_SALT,
                                         "label", strlen("label"));
*p++ = OSSL_PARAM_construct_octet_string(OSSL_KDF_PARAM_INFO,
                                         "context", strlen("context"));
*p = OSSL_PARAM_construct_end();
if (EVP_KDF_derive(kctx, out, sizeof(out), params) <= 0)
    error("EVP_KDF_derive");

EVP_KDF_CTX_free(kctx);
```

This example derives 10 bytes using FEEDBACK-CMAC-AES256, with KI "secret", Label "label", and IV "sixteen bytes iv".

```
EVP_KDF *kdf;
EVP_KDF_CTX *kctx;
unsigned char out[10];
OSSL_PARAM params[8], *p = params;
unsigned char *iv = "sixteen bytes iv";

kdf = EVP_KDF_fetch(NULL, "KBKDF", NULL);
kctx = EVP_KDF_CTX_new(kdf);
EVP_KDF_free(kdf);

*p++ = OSSL_PARAM_construct_utf8_string(OSSL_KDF_PARAM_CIPHER, "AES256", 0);
*p++ = OSSL_PARAM_construct_utf8_string(OSSL_KDF_PARAM_MAC, "CMAC", 0);
*p++ = OSSL_PARAM_construct_utf8_string(OSSL_KDF_PARAM_MODE, "FEEDBACK", 0);
*p++ = OSSL_PARAM_construct_octet_string(OSSL_KDF_PARAM_KEY,
                                         "secret", strlen("secret"));
*p++ = OSSL_PARAM_construct_octet_string(OSSL_KDF_PARAM_SALT,
                                         "label", strlen("label"));
*p++ = OSSL_PARAM_construct_octet_string(OSSL_KDF_PARAM_INFO,
                                         "context", strlen("context"));
*p++ = OSSL_PARAM_construct_octet_string(OSSL_KDF_PARAM_SEED,
                                         iv, strlen(iv));
*p = OSSL_PARAM_construct_end();
if (EVP_KDF_derive(kctx, out, sizeof(out), params) <= 0)
    error("EVP_KDF_derive");

EVP_KDF_CTX_free(kctx);
```

CONFORMING TO

NIST SP800−108, IETF RFC 6803, IETF RFC 8009.

SEE ALSO

EVP_KDF (3), **EVP_KDF_CTX_free** (3), **EVP_KDF_CTX_get_kdf_size** (3), **EVP_KDF_derive** (3), ‘‘PARAMETERS’’ in **EVP_KDF** (3)

HISTORY

This functionality was added to OpenSSL 3.0.

COPYRIGHT