NAME

EVP_PKEY_CTX_set_hkdf_md, EVP_PKEY_CTX_set1_hkdf_salt, EVP_PKEY_CTX_set1_hkdf_key, EVP_PKEY_CTX_add1_hkdf_info, EVP_PKEY_CTX_set_hkdf_mode - HMAC-based Extract-and-Expand key derivation algorithm

SYNOPSIS

```
#include <openssl/kdf.h>

int EVP_PKEY_CTX_set_hkdf_mode(EVP_PKEY_CTX *pctx, int mode);

int EVP_PKEY_CTX_set_hkdf_md(EVP_PKEY_CTX *pctx, const EVP_MD *md);

int EVP_PKEY_CTX_set1_hkdf_salt(EVP_PKEY_CTX *pctx, unsigned char *salt,
                                int saltlen);

int EVP_PKEY_CTX_set1_hkdf_key(EVP_PKEY_CTX *pctx, unsigned char *key,
                               int keylen);

int EVP_PKEY_CTX_add1_hkdf_info(EVP_PKEY_CTX *pctx, unsigned char *info,
                                int infolen);
```

DESCRIPTION

The EVP_PKEY_HKDF algorithm implements the HKDF key derivation function. HKDF follows the "extract-then-expand" paradigm, where the KDF logically consists of two modules. The first stage takes the input keying material and "extracts" from it a fixed-length pseudorandom key K. The second stage "expands" the key K into several additional pseudorandom keys (the output of the KDF).

EVP_PKEY_CTX_set_hkdf_mode() sets the mode for the HKDF operation. There are three modes that are currently defined:

EVP_PKEY_HKDEF_MODE_EXTRACT_AND_EXPAND

This is the default mode. Calling **EVP_PKEY_derive**(3) on an EVP_PKEY_CTX set up for HKDF will perform an extract followed by an expand operation in one go. The derived key returned will be the result after the expand operation. The intermediate fixed-length pseudorandom key K is not returned.

In this mode the digest, key, salt and info values must be set before a key is derived or an error occurs.

EVP_PKEY_HKDEF_MODE_EXTRACT_ONLY

In this mode calling **EVP_PKEY_derive**(3) will just perform the extract operation. The value returned will be the intermediate fixed-length pseudorandom key K.

The digest, key and salt values must be set before a key is derived or an error occurs.

EVP_PKEY_HKDEF_MODE_EXPAND_ONLY

In this mode calling **EVP_PKEY_derive**(3) will just perform the expand operation. The input key should be set to the intermediate fixed-length pseudorandom key K returned from a previous extract operation.

The digest, key and info values must be set before a key is derived or an error occurs.

EVP_PKEY_CTX_set_hkdf_md() sets the message digest associated with the HKDF.

EVP_PKEY_CTX_set1_hkdf_salt() sets the salt to **saltlen** bytes of the buffer **salt**. Any existing value is replaced.

EVP_PKEY_CTX_set1_hkdf_key() sets the key to **keylen** bytes of the buffer **key**. Any existing value is replaced.

EVP_PKEY_CTX_add1_hkdf_info() sets the info value to **infolen** bytes of the buffer **info**. If a value is already set, it is appended to the existing value.

STRING CTRLS

HKDF also supports string based control operations via **EVP_PKEY_CTX_ctrl_str**(3). The **type** parameter "md" uses the supplied **value** as the name of the digest algorithm to use. The **type** parameter "mode" uses the values "EXTRACT_AND_EXPAND", "EXTRACT_ONLY" and "EXPAND_ONLY" to determine the mode to use. The **type** parameters "salt", "key" and "info" use the supplied **value** parameter as a **seed**, **key** or **info** value. The names "hexsalt", "hexkey" and "hexinfo" are similar except they take a hex string which is converted to binary.

NOTES

A context for HKDF can be obtained by calling:

```
 EVP_PKEY_CTX *pctx = EVP_PKEY_CTX_new_id(EVP_PKEY_HKDF, NULL);
```

The total length of the info buffer cannot exceed 2048 bytes in length: this should be more than enough for any normal use of HKDF.

The output length of an HKDF expand operation is specified via the length parameter to the

EVP_PKEY_derive(3) function. Since the HKDF output length is variable, passing a **NULL** buffer as a means to obtain the requisite length is not meaningful with HKDF in any mode that performs an expand operation. Instead, the caller must allocate a buffer of the desired length, and pass that buffer to **EVP_PKEY_derive**(3) along with (a pointer initialized to) the desired length. Passing a **NULL** buffer to obtain the length is allowed when using EVP_PKEY_HKDEF_MODE_EXTRACT_ONLY.

Optimised versions of HKDF can be implemented in an ENGINE.

RETURN VALUES

All these functions return 1 for success and 0 or a negative value for failure. In particular a return value of -2 indicates the operation is not supported by the public key algorithm.

EXAMPLES

This example derives 10 bytes using SHA-256 with the secret key "secret", salt value "salt" and info value "label":

```
 EVP_PKEY_CTX *pctx;
 unsigned char out[10];
 size_t outlen = sizeof(out);
 pctx = EVP_PKEY_CTX_new_id(EVP_PKEY_HKDF, NULL);

 if (EVP_PKEY_derive_init(pctx) <= 0)
     /* Error */
 if (EVP_PKEY_CTX_set_hkdf_md(pctx, EVP_sha256()) <= 0)
     /* Error */
 if (EVP_PKEY_CTX_set1_hkdf_salt(pctx, "salt", 4) <= 0)
     /* Error */
 if (EVP_PKEY_CTX_set1_hkdf_key(pctx, "secret", 6) <= 0)
     /* Error */
 if (EVP_PKEY_CTX_add1_hkdf_info(pctx, "label", 5) <= 0)
     /* Error */
 if (EVP_PKEY_derive(pctx, out, &outlen) <= 0)
     /* Error */
```

CONFORMING TO

RFC 5869

SEE ALSO

EVP_PKEY_CTX_new(3), **EVP_PKEY_CTX_ctrl_str**(3), **EVP_PKEY_derive**(3)

HISTORY

All of the functions described here were converted from macros to functions in OpenSSL 3.0.

COPYRIGHT