Happy Key: HPKE implementation (RFC9180)

https://github.com/sftcd/happykey

Generated by Doxygen 1.9.1

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 3

Data Structure Documentation

3.1 OSSL_HPKE_SUITE Struct Reference

ciphersuite combination

`#include <hpke.h>`

Data Fields

- uint16_t kem_id
  Key Encryption Method id.
- uint16_t kdf_id
  Key Derivation Function id.
- uint16_t aead_id
  AEAD alg id.

3.1.1 Detailed Description

ciphersuite combination

The documentation for this struct was generated from the following file:

- hpke.h

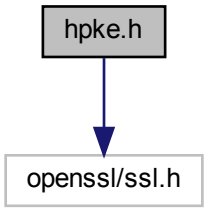
Chapter 4

File Documentation

4.1 hpke.h File Reference

APIs and data structures for HPKE (RFC9180).

`#include <openssl/ssl.h>`
Include dependency graph for hpke.h:



Data Structures

- struct OSSL_HPKE_SUITE
  ciphersuite combination

Macros

- #define OSSL_HPKE_MODE_BASE 0
  Base mode
- #define OSSL_HPKE_MODE_PSK 1
  Pre-shared key mode.
- #define OSSL_HPKE_MODE_AUTH 2
  Authenticated mode.

- #define OSSL_HPKE_MODE_PSKAUTH 3
  PSK+authenticated mode.
- #define OSSL_HPKE_KEM_ID_RESERVED 0x0000
  not used
- #define OSSL_HPKE_KEM_ID_P256 0x0010
  NIST P-256.
- #define OSSL_HPKE_KEM_ID_P384 0x0011
  NIST P-256.
- #define OSSL_HPKE_KEM_ID_P521 0x0012
  NIST P-521.
- #define OSSL_HPKE_KEM_ID_25519 0x0020
  Curve25519.
- #define OSSL_HPKE_KEM_ID_448 0x0021
  Curve448.
- #define OSSL_HPKE_KDF_ID_RESERVED 0x0000
  not used
- #define OSSL_HPKE_KDF_ID_HKDF_SHA256 0x0001
  HKDF-SHA256.
- #define OSSL_HPKE_KDF_ID_HKDF_SHA384 0x0002
  HKDF-SHA384.
- #define OSSL_HPKE_KDF_ID_HKDF_SHA512 0x0003
  HKDF-SHA512.
- #define OSSL_HPKE_AEAD_ID_RESERVED 0x0000
  not used
- #define OSSL_HPKE_AEAD_ID_AES_GCM_128 0x0001
  AES-GCM-128.
- #define OSSL_HPKE_AEAD_ID_AES_GCM_256 0x0002
  AES-GCM-256.
- #define OSSL_HPKE_AEAD_ID_CHACHA_POLY1305 0x0003
  Chacha20-Poly1305.
- #define OSSL_HPKE_AEAD_ID_EXPORTONLY 0xFFFF
  export-only fake ID
- #define OSSL_HPKE_MODESTR_BASE "base"
  base mode (1)
- #define OSSL_HPKE_MODESTR_PSK "psk"
  psk mode (2)
- #define OSSL_HPKE_MODESTR_AUTH "auth"
  sender-key pair auth (3)
- #define OSSL_HPKE_MODESTR_PSKAUTH "pskauth"
  psk+sender-key pair (4)
- #define OSSL_HPKE_KEMSTR_P256 "P-256"
  KEM id 0x10.
- #define OSSL_HPKE_KEMSTR_P384 "P-384"
  KEM id 0x11.
- #define OSSL_HPKE_KEMSTR_P521 "P-521"
  KEM id 0x12.
- #define OSSL_HPKE_KEMSTR_X25519 SN_X25519
  KEM id 0x20.
- #define OSSL_HPKE_KEMSTR_X448 SN_X448
  KEM id 0x21.
- #define OSSL_HPKE_KDFSTR_256 "hkdf-sha256"

KDF id 1.

- #define OSSL_HPKE_KDFSTR_384 "hkdf-sha384"

  KDF id 2.
- #define OSSL_HPKE_KDFSTR_512 "hkdf-sha512"

  KDF id 3.
- #define OSSL_HPKE_AEADSTR_AES128GCM LN_aes_128_gcm

  AEAD id 1.
- #define OSSL_HPKE_AEADSTR_AES256GCM LN_aes_256_gcm

  AEAD id 2.
- #define OSSL_HPKE_AEADSTR_CP LN_chacha20_poly1305

  AEAD id 3.
- #define OSSL_HPKE_AEADSTR_EXP "exporter"

  AEAD id 0xff.
- #define OSSL_HPKE_SUITE_DEFAULT

  Suite constants, use this like: OSSL_HPKE_SUITE myvar = OSSL_HPKE_SUITE_DEFAULT;.
- #define **OSSL_HPKE_MAXSIZE** 512

Typedefs

- typedef struct ossl_hpke_ctx_st OSSL_HPKE_CTX

  opaque type for HPKE contexts

Functions

- OSSL_HPKE_CTX * OSSL_HPKE_CTX_new (int mode, OSSL_HPKE_SUITE suite, OSSL_LIB_CTX *libctx, const char *propq)

  context creator
- void OSSL_HPKE_CTX_free (OSSL_HPKE_CTX *ctx)

  free up storage for a HPKE context
- int OSSL_HPKE_CTX_set1_psk (OSSL_HPKE_CTX *ctx, const char *pskid, const unsigned char *psk, size_t psklen)

  set a PSK for an HPKE context
- int OSSL_HPKE_CTX_set1_senderpriv (OSSL_HPKE_CTX *ctx, EVP_PKEY *privp)

  set a sender KEM private key for HPKE
- int OSSL_HPKE_CTX_set1_authpriv (OSSL_HPKE_CTX *ctx, EVP_PKEY *privp)

  set a sender private key for HPKE authenticated modes
- int OSSL_HPKE_CTX_set1_authpub (OSSL_HPKE_CTX *ctx, unsigned char *pub, size_t publen)

  set a public key for HPKE authenticated modes
- int OSSL_HPKE_CTX_get0_seq (OSSL_HPKE_CTX *ctx, uint64_t *seq)

  ask for the state of the sequence of seal/open calls
- int OSSL_HPKE_CTX_set1_seq (OSSL_HPKE_CTX *ctx, uint64_t seq)

  set the sequence value for seal/open calls
- int OSSL_HPKE_sender_seal (OSSL_HPKE_CTX *ctx, unsigned char *enc, size_t *enclen, unsigned char *ct, size_t *ctlen, unsigned char *pub, size_t publen, const unsigned char *info, size_t infolen, const unsigned char *aad, size_t aadlen, const unsigned char *pt, size_t ptlen)

  sender seal function
- int OSSL_HPKE_recipient_open (OSSL_HPKE_CTX *ctx, unsigned char *pt, size_t *ptlen, const unsigned char *enc, size_t enclen, EVP_PKEY *recippriv, const unsigned char *info, size_t infolen, const unsigned char *aad, size_t aadlen, const unsigned char *ct, size_t ctlen)

  recipient open function

- int OSSL_HPKE_CTX_export (OSSL_HPKE_CTX *ctx, unsigned char *secret, size_t secret_len, const unsigned char *label, size_t labellen)

  generate a given-length secret based on context and label
- int OSSL_HPKE_keygen (OSSL_LIB_CTX *libctx, const char *propq, unsigned int mode, OSSL_HPKE_SUITE suite, const unsigned char *ikm, size_t ikmlen, unsigned char *pub, size_t *publen, EVP_PKEY **priv)

  generate a key pair
- int OSSL_HPKE_suite_check (OSSL_HPKE_SUITE suite)

  check if a suite is supported locally
- int OSSL_HPKE_good4grease (OSSL_LIB_CTX *libctx, const char *propq, OSSL_HPKE_SUITE *suite_in, OSSL_HPKE_SUITE *suite, unsigned char *pub, size_t *pub_len, unsigned char *cipher, size_t cipher_len)

  get a (possibly) random suite, public key and ciphertext for GREASErs
- int OSSL_HPKE_str2suite (const char *str, OSSL_HPKE_SUITE *suite)

  map a string to a HPKE suite
- int OSSL_HPKE_expansion (OSSL_HPKE_SUITE suite, size_t *enclen, size_t clearlen, size_t *cipherlen)

  tell the caller how big the cipertext will be
- int OSSL_HPKE_enc (OSSL_LIB_CTX *libctx, const char *propq, unsigned int mode, OSSL_HPKE_SUITE suite, const char *pskid, const unsigned char *psk, size_t psklen, const unsigned char *pub, size_t publen, const unsigned char *authpriv, size_t authprivlen, EVP_PKEY *authpriv_evp, const unsigned char *clear, size_t clearlen, const unsigned char *aad, size_t aadlen, const unsigned char *info, size_t infolen, const unsigned char *seq, size_t seqlen, unsigned char *senderpub, size_t *senderpublen, EVP_PKEY *senderpriv, unsigned char *cipher, size_t *cipherlen)

  HPKE single-shot encryption function.
- int OSSL_HPKE_enc_evp (OSSL_LIB_CTX *libctx, const char *propq, unsigned int mode, OSSL_HPKE_SUITE suite, const char *pskid, const unsigned char *psk, size_t psklen, const unsigned char *pub, size_t publen, const unsigned char *authpriv, size_t authprivlen, EVP_PKEY *authpriv_evp, const unsigned char *clear, size_t clearlen, const unsigned char *aad, size_t aadlen, const unsigned char *info, size_t infolen, const unsigned char *seq, size_t seqlen, const unsigned char *senderpub, size_t senderpublen, EVP_PKEY *senderpriv, unsigned char *cipher, size_t *cipherlen)

  HPKE multi-shot encryption function.
- int OSSL_HPKE_dec (OSSL_LIB_CTX *libctx, const char *propq, unsigned int mode, OSSL_HPKE_SUITE suite, const char *pskid, const unsigned char *psk, size_t psklen, const unsigned char *pub, size_t publen, const unsigned char *priv, size_t privlen, EVP_PKEY *evppriv, const unsigned char *enc, size_t enclen, const unsigned char *cipher, size_t cipherlen, const unsigned char *aad, size_t aadlen, const unsigned char *info, size_t infolen, const unsigned char *seq, size_t seqlen, unsigned char *clear, size_t *clearlen)

  HPKE single-shot decryption function.

4.1.1 Detailed Description

APIs and data structures for HPKE (RFC9180).

4.1.2 Macro Definition Documentation

4.1.2.1 OSSL_HPKE_SUITE_DEFAULT

```
#define OSSL_HPKE_SUITE_DEFAULT
```

Value:

```
    {\
        OSSL_HPKE_KEM_ID_25519, \
        OSSL_HPKE_KDF_ID_HKDF_SHA256, \
        OSSL_HPKE_AEAD_ID_AES_GCM_128 \
    }
```

Suite constants, use this like: OSSL_HPKE_SUITE myvar = OSSL_HPKE_SUITE_DEFAULT;.

4.1.3 Function Documentation

4.1.3.1 OSSL_HPKE_CTX_export()

```
int OSSL_HPKE_CTX_export (
            OSSL_HPKE_CTX * ctx,
            unsigned char * secret,
            size_t secret_len,
            const unsigned char * label,
            size_t labellen )
```

generate a given-length secret based on context and label

Parameters

ctx	is the HPKE context
secret	is the resulting secret that will be of length...
secret_len	is the desired output length
label	is a buffer to provide separation between secrets
labellen	is the length of the above

Returns

1 for good, 0 for error

The context has to have been used already for one encryption or decryption for this to work (as this is based on the negotiated "exporter_secret" estabilshed via the HPKE operation).

4.1.3.2 OSSL_HPKE_CTX_free()

```
void OSSL_HPKE_CTX_free (
            OSSL_HPKE_CTX * ctx )
```

free up storage for a HPKE context

Parameters

ctx	is the pointer to be free'd (can be NULL)

4.1.3.3 OSSL_HPKE_CTX_get0_seq()

```
int OSSL_HPKE_CTX_get0_seq (
            OSSL_HPKE_CTX * ctx,
            uint64_t * seq )
```

ask for the state of the sequence of seal/open calls

Parameters

ctx	is the pointer for the HPKE context
seq	returns the positive integer sequence number

Returns

1 for success, 0 for error

The value returned is the next one to be used when sealing or opening (so as we start at zero this will be 1 after the first successful call to seal or open)

seq is a uint64_t as that's what two other implementations chose

4.1.3.4 OSSL_HPKE_CTX_new()

```
OSSL_HPKE_CTX* OSSL_HPKE_CTX_new (
            int mode,
            OSSL_HPKE_SUITE suite,
            OSSL_LIB_CTX * libctx,
            const char * propq )
```

context creator

Parameters

mode	is the desired HPKE mode
suite	specifies the KEM, KDF and AEAD to use
libctx	is the library context to use
propq	is a properties string for the library

Returns

pointer to new context or NULL if error

4.1.3.5 OSSL_HPKE_CTX_set1_authpriv()

```
int OSSL_HPKE_CTX_set1_authpriv (
            OSSL_HPKE_CTX * ctx,
            EVP_PKEY * privp )
```

set a sender private key for HPKE authenticated modes

Parameters

ctx	is the pointer for the HPKE context
privp	is an EVP_PKEY form of the private key

Returns

1 for success, 0 for error

4.1.3.6 OSSL_HPKE_CTX_set1_authpub()

```
int OSSL_HPKE_CTX_set1_authpub (
            OSSL_HPKE_CTX * ctx,
            unsigned char * pub,
            size_t publen )
```

set a public key for HPKE authenticated modes

Parameters

ctx	is the pointer for the HPKE context
pub	is an buffer form of the public key
publen	is the length of the above

Returns

1 for success, 0 for error

In all these APIs public keys are passed as buffers whereas private keys as passed as EVP_PKEY pointers.

4.1.3.7 OSSL_HPKE_CTX_set1_psk()

```
int OSSL_HPKE_CTX_set1_psk (
            OSSL_HPKE_CTX * ctx,
            const char * pskid,
            const unsigned char * psk,
            size_t psklen )
```

set a PSK for an HPKE context

Parameters

ctx	is the pointer for the HPKE context
pskid	is a string identifying the PSK
psk	is the PSK buffer
psklen	is the size of the PSK

Returns

1 for success, 0 for error

4.1.3.8 OSSL_HPKE_CTX_set1_senderpriv()

```
int OSSL_HPKE_CTX_set1_senderpriv (
            OSSL_HPKE_CTX * ctx,
            EVP_PKEY * privp )
```

set a sender KEM private key for HPKE

Parameters

ctx	is the pointer for the HPKE context
privp	is an EVP_PKEY form of the private key

Returns

1 for success, 0 for error

If no key is set via this API an ephemeral one will be generated in the first seal operation and used until the context is free'd. (Or until a subsequent call to this API replaces the key.) This suits senders who are typically clients.

4.1.3.9 OSSL_HPKE_CTX_set1_seq()

```
int OSSL_HPKE_CTX_set1_seq (
            OSSL_HPKE_CTX * ctx,
            uint64_t seq )
```

set the sequence value for seal/open calls

Parameters

ctx	is the pointer for the HPKE context
seq	set the positive integer sequence number

Returns

1 for success, 0 for error

The next seal or open operation will use this value.

4.1.3.10 OSSL_HPKE_dec()

```
int OSSL_HPKE_dec (
            OSSL_LIB_CTX * libctx,
            const char * propq,
            unsigned int mode,
            OSSL_HPKE_SUITE suite,
            const char * pskid,
            const unsigned char * psk,
            size_t psklen,
```

```
const unsigned char * pub,
size_t publen,
const unsigned char * priv,
size_t privlen,
EVP_PKEY * evppriv,
const unsigned char * enc,
size_t enclen,
const unsigned char * cipher,
size_t cipherlen,
const unsigned char * aad,
size_t aadlen,
const unsigned char * info,
size_t infolen,
const unsigned char * seq,
size_t seqlen,
unsigned char * clear,
size_t * clearlen )
```

HPKE single-shot decryption function.

Parameters

libctx	is the context to use (normally NULL)
propq	is a properties string
mode	is the HPKE mode
suite	is the ciphersuite to use
pskid	is the pskid string for a PSK mode (can be NULL)
psk	is the psk
psklen	is the psk length
pub	is the encoded public (authentication) key
publen	is the length of the public (authentication) key
priv	is the encoded private key
privlen	is the length of the private key
evppriv	is a pointer to an internal form of private key
enc	is the peer's public value
enclen	is the length of the peer's public value
cipher	is the ciphertext
cipherlen	is the length of the ciphertext
aad	is the encoded additional data
aadlen	is the length of the additional data
info	is the encoded info data (can be NULL)
infolen	is the length of the info data (can be zero)
seq	is the encoded sequence data (can be NULL)
seqlen	is the length of the sequence data (can be zero)
clear	is the encoded cleartext
clearlen	length of the input buffer for cleartext

Returns

1 for success, other for error (error returns can be non-zero)

4.1.3.11 OSSL_HPKE_enc()

```
int OSSL_HPKE_enc (
            OSSL_LIB_CTX * libctx,
            const char * propq,
            unsigned int mode,
            OSSL_HPKE_SUITE suite,
            const char * pskid,
            const unsigned char * psk,
            size_t psklen,
            const unsigned char * pub,
            size_t publen,
            const unsigned char * authpriv,
            size_t authprivlen,
            EVP_PKEY * authpriv_evp,
            const unsigned char * clear,
            size_t clearlen,
            const unsigned char * aad,
            size_t aadlen,
            const unsigned char * info,
            size_t infolen,
            const unsigned char * seq,
            size_t seqlen,
            unsigned char * senderpub,
            size_t * senderpublen,
            EVP_PKEY * senderpriv,
            unsigned char * cipher,
            size_t * cipherlen )
```

HPKE single-shot encryption function.

This function generates an ephemeral ECDH value internally and provides the public component as an output that can be sent to the relevant private key holder along with the ciphertext.

Note that the sender's public value is an output here in contrast to the case of OSSL_HPKE_enc_evp where the sender's public value is an input (along with the sender's private value).

Parameters

libctx	is the context to use (normally NULL)
propq	is a properties string
mode	is the HPKE mode
suite	is the ciphersuite to use
pskid	is the pskid string for a PSK mode (can be NULL)
psk	is the psk
psklen	is the psk length
pub	is the encoded public key
publen	is the length of the public key
authpriv	is the encoded private (authentication) key
authprivlen	is the length of the private (authentication) key
authpriv_evp	is the EVP_PKEY* form of private (authentication) key
clear	is the encoded cleartext
clearlen	is the length of the cleartext
aad	is the encoded additional data
aadlen	is the length of the additional data

Parameters

info	is the encoded info data (can be NULL)
infolen	is the length of the info data (can be zero)
seq	is the encoded sequence data (can be NULL)
seqlen	is the length of the sequence data (can be zero)
senderpub	is the input buffer for sender public key
senderpublen	length of the input buffer for sender's public key
senderpriv	is the sender's private key (if being re-used)
cipher	is the input buffer for ciphertext
cipherlen	is the length of the input buffer for ciphertext

Returns

1 for success, other for error (error returns can be non-zero)

4.1.3.12 OSSL_HPKE_enc_evp()

```
int OSSL_HPKE_enc_evp (
            OSSL_LIB_CTX * libctx,
            const char * propq,
            unsigned int mode,
            OSSL_HPKE_SUITE suite,
            const char * pskid,
            const unsigned char * psk,
            size_t psklen,
            const unsigned char * pub,
            size_t publen,
            const unsigned char * authpriv,
            size_t authprivlen,
            EVP_PKEY * authpriv_evp,
            const unsigned char * clear,
            size_t clearlen,
            const unsigned char * aad,
            size_t aadlen,
            const unsigned char * info,
            size_t infolen,
            const unsigned char * seq,
            size_t seqlen,
            const unsigned char * senderpub,
            size_t senderpublen,
            EVP_PKEY * senderpriv,
            unsigned char * cipher,
            size_t * cipherlen )
```

HPKE multi-shot encryption function.

This function generates a non-ephemeral ECDH value internally and provides the public and private components as outputs. The public part can be sent to the relevant private key holder along with the ciphertext. The private part can be re-used in subequent calls.

Note that the sender's public value is an input here (as is the sender's private value), in contrast to the case of OSSL_HPKE_enc where the sender's public value is an output.

Parameters

libctx	is the context to use
propq	is a properties string
mode	is the HPKE mode
suite	is the ciphersuite to use
pskid	is the pskid string for a PSK mode (can be NULL)
psk	is the psk
psklen	is the psk length
pub	is the encoded public key
publen	is the length of the public key
authpriv	is the encoded private (authentication) key
authprivlen	is the length of the private (authentication) key
authpriv_evp	is the EVP_PKEY* form of private (authentication) key
clear	is the encoded cleartext
clearlen	is the length of the cleartext
aad	is the encoded additional data
aadlen	is the length of the additional data
info	is the encoded info data (can be NULL)
infolen	is the length of the info data (can be zero)
seq	is the encoded sequence data (can be NULL)
seqlen	is the length of the sequence data (can be zero)
senderpub	is the input buffer for sender public key
senderpublen	length of the input buffer for sender's public key
senderpriv	is the EVP_PKEY* form of sender key pair
cipher	is the input buffer for ciphertext
cipherlen	is the length of the input buffer for ciphertext

Returns

1 for success, other for error (error returns can be non-zero)

4.1.3.13 OSSL_HPKE_expansion()

```
int OSSL_HPKE_expansion (
            OSSL_HPKE_SUITE suite,
            size_t * enclen,
            size_t clearlen,
            size_t * cipherlen )
```

tell the caller how big the cipertext will be

Parameters

suite	is the suite to be used
enclen	points to what'll be enc length
clearlen	is the length of plaintext
cipherlen	points to what'll be ciphertext length

Returns

1 for success, otherwise failure

4.1.3.14 OSSL_HPKE_good4grease()

```
int OSSL_HPKE_good4grease (
            OSSL_LIB_CTX * libctx,
            const char * propq,
            OSSL_HPKE_SUITE * suite_in,
            OSSL_HPKE_SUITE * suite,
            unsigned char * pub,
            size_t * pub_len,
            unsigned char * cipher,
            size_t cipher_len )
```

get a (possibly) random suite, public key and ciphertext for GREASErs

Parameters

libctx	is the context to use (normally NULL)
propq	is a properties string
suite_in	specifies the preferred suite or NULL for a random choice
suite	is the chosen or random suite
pub	a random value of the appropriate length for a sender public value
pub_len	is the length of pub (buffer size on input)
cipher	is a random value of the appropriate length for a ciphertext
cipher_len	is the length of cipher

Returns

1 for success, otherwise failure

If suite_in is provided that will be used (if supported). If suite_in is NULL, a random suite (from those supported) will be selected. In all cases the output pub and cipher values will be appropriate random values for the selected suite.

4.1.3.15 OSSL_HPKE_keygen()

```
int OSSL_HPKE_keygen (
            OSSL_LIB_CTX * libctx,
            const char * propq,
            unsigned int mode,
            OSSL_HPKE_SUITE suite,
            const unsigned char * ikm,
            size_t ikmlen,
            unsigned char * pub,
            size_t * publen,
            EVP_PKEY ** priv )
```

generate a key pair

Parameters

libctx	is the context to use (normally NULL)
propq	is a properties string
mode	is the mode (currently unused)
suite	is the ciphersuite (currently unused)
ikm	is IKM, if supplied
ikmlen	is the length of IKM, if supplied
pub	is the public value
publen	is the size of the public key buffer (exact length on output)
priv	is the private key

Returns

1 for success, other for error (error returns can be non-zero)

Used for entities that will later receive HPKE values to decrypt or that want a private key for an AUTH mode. Currently, only the KEM from the suite is significant here. The `pub` output will typically be published so that others can encrypt to the private key holder using HPKE. (Or authenticate HPKE values from that sender.)

4.1.3.16 OSSL_HPKE_recipient_open()

```
int OSSL_HPKE_recipient_open (
              OSSL_HPKE_CTX * ctx,
              unsigned char * pt,
              size_t * ptlen,
              const unsigned char * enc,
              size_t enclen,
              EVP_PKEY * recippriv,
              const unsigned char * info,
              size_t infolen,
              const unsigned char * aad,
              size_t aadlen,
              const unsigned char * ct,
              size_t ctlen )
```

recipient open function

Parameters

ctx	is the pointer for the HPKE context
pt	is the plaintext
ptlen	is the size the above
enc	is the sender's ephemeral public value
enclen	is the size the above
recippriv	is the EVP_PKEY form of recipient private value
info	is the info parameter
infolen	is the size the above
aad	is the aad parameter
aadlen	is the size the above
ct	is the ciphertext output
ctlen	is the size the above

Returns

1 for success, 0 for error

This can be called once, or multiple, times.

The recipient private key is explicitly set here as recipients are likely to be servers with multiple long(ish) term private keys in memory at once and that may have to e.g. do trial decryptions.

The plaintext output (pt) will be smaller than the ciphertext input for all supported suites.

4.1.3.17 OSSL_HPKE_sender_seal()

```
int OSSL_HPKE_sender_seal (
            OSSL_HPKE_CTX * ctx,
            unsigned char * enc,
            size_t * enclen,
            unsigned char * ct,
            size_t * ctlen,
            unsigned char * pub,
            size_t publen,
            const unsigned char * info,
            size_t infolen,
            const unsigned char * aad,
            size_t aadlen,
            const unsigned char * pt,
            size_t ptlen )
```

sender seal function

Parameters

ctx	is the pointer for the HPKE context
enc	is the sender's ephemeral public value
enclen	is the size the above
ct	is the ciphertext output
ctlen	is the size the above
pub	is the recipient public key octets
publen	is the size the above
recip	is the EVP_PKEY form of recipient public value
info	is the info parameter
infolen	is the size the above
aad	is the aad parameter
aadlen	is the size the above
pt	is the plaintext
ptlen	is the size the above

Returns

1 for success, 0 for error

This can be called once, or multiple, times.

If no KEM private key has been set in the context an ephemeral key will be generated and used for the duration of the context.

The ciphertext buffer (ct) should be big enough to include the AEAD tag generated from encryptions and the `enc` buffer (the ephemeral public key) needs to be big enough for the relevant KEM. `OSSL_HPKE_expansion` can be used to determine the sizes needed.

4.1.3.18 OSSL_HPKE_str2suite()

```
int OSSL_HPKE_str2suite (
            const char * str,
            OSSL_HPKE_SUITE * suite )
```

map a string to a HPKE suite

Parameters

str	is the string value
suite	is the resulting suite

Returns

1 for success, otherwise failure

An example good string is "x25519,hkdf-sha256,aes-128-gcm" Symbols are #define'd for the relevant labels, e.g. OSSL_HPKE_KEMSTR_X25519. Numeric (decimal or hex) values with the relevant IANA codepoint values from RFC9180 may be used, e.g., "0x20,1,1" represents the same suite as the first example.

4.1.3.19 OSSL_HPKE_suite_check()

```
int OSSL_HPKE_suite_check (
            OSSL_HPKE_SUITE suite )
```

check if a suite is supported locally

Parameters

suite	is the suite to check

Returns

1 for success, other for error (error returns can be non-zero)

Index