

Happy Key: HPKE implementation (draft-irtf-cfrg-hpke)

Generated by Doxygen 1.8.13

Contents

1	Data Structure Index	1
1.1	Data Structures	1
2	File Index	3
2.1	File List	3
3	Data Structure Documentation	5
3.1	hpke_suite_t Struct Reference	5
4	File Documentation	7
4.1	hpke.c File Reference	7
4.2	hpke.h File Reference	7
4.2.1	Detailed Description	9
4.2.2	Function Documentation	9
4.2.2.1	hpke_ah_decode()	9
4.2.2.2	hpke_enc()	9
4.3	hpke_main.c File Reference	10
4.3.1	Detailed Description	10
4.3.2	Macro Definition Documentation	10
4.3.2.1	FINDLAB	11
4.4	hpke_tv.c File Reference	11
4.4.1	Detailed Description	11
4.5	hpke_tv.h File Reference	11
4.5.1	Detailed Description	11
	Index	13

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

hpke_suite_t	5
--	---

Chapter 2

File Index

2.1 File List

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

hpke.c	An OpenSSL-based HPKE implementation following draft-irtf-cfrg-hpke	7
hpke.h	This has the data structures and prototypes (both internal and external) for an OpenSSL-based HPKE implementation following draft-irtf-cfrg-hpke	7
hpke_main.c	An OpenSSL-based HPKE implementation following draft-irtf-cfrg-hpke	10
hpke_tv.c	Implementation related to test vectors for HPKE	11
hpke_tv.h	Header file related to test vectors for HPKE	11

Chapter 3

Data Structure Documentation

3.1 hpke_suite_t Struct Reference

Data Fields

- uint16_t [kem_id](#)
Key Encryption Method id.
- uint16_t [kdf_id](#)
Key Derivation Function id.
- uint16_t [aead_id](#)
Authenticated Encryption with Associated Data id.

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

- [hpke.h](#)

Chapter 4

File Documentation

4.1 hpke.c File Reference

An OpenSSL-based HPKE implementation following draft-irtf-cfrg-hpke.

```
#include <stddef.h>
#include <stdint.h>
#include <string.h>
#include <openssl/ssl.h>
#include <openssl/rand.h>
#include <openssl/kdf.h>
#include <openssl/evp.h>
#include <openssl/params.h>
#include "hpke.h"
Include dependency graph for hpke.c:
```

4.2 hpke.h File Reference

This has the data structures and prototypes (both internal and external) for an OpenSSL-based HPKE implementation following draft-irtf-cfrg-hpke.

This graph shows which files directly or indirectly include this file:

Data Structures

- struct [hpke_suite_t](#)

Macros

- #define `HPKE_MAXSIZE` (640*1024)
640k is more than enough for anyone (using this program:-)
- #define `HPKE_MODE_BASE` 0
Base mode (all that we support for now)
- #define `HPKE_MODE_PSK` 1
Pre-shared key mode.
- #define `HPKE_MODE_AUTH` 2
Authenticated mode.
- #define `HPKE_MODE_PSK_AUTH` 3
PSK+authenticated mode.
- #define `HPKE_KEM_ID_RESERVED` 0x0000
not used
- #define `HPKE_KEM_ID_P256` 0x0001
NIST P-256.
- #define `HPKE_KEM_ID_25519` 0x0002
Curve25519.
- #define `HPKE_KEM_ID_P521` 0x0003
NIST P-521.
- #define `HPKE_KEM_ID_448` 0x0004
Curve448.
- #define `HPKE_KDF_ID_RESERVED` 0x0000
not used
- #define `HPKE_KDF_ID_HKDF_SHA256` 0x0001
HKDF-SHA256.
- #define `HPKE_KDF_ID_HKDF_SHA512` 0x0002
HKDF-SHA512.
- #define `HPKE_AEAD_ID_RESERVED` 0x0000
not used
- #define `HPKE_AEAD_ID_AES_GCM_128` 0x0001
AES-GCM-128.
- #define `HPKE_AEAD_ID_AES_GCM_256` 0x0002
AES-GCM-256.
- #define `HPKE_AEAD_ID_CHACHA_POLY1305` 0x0003
Chacha20-Poly1305.
- #define `HPKE_SUITE_DEFAULT` { `HPKE_KEM_ID_25519`, `HPKE_KDF_ID_HKDF_SHA256`, `HPKE_AEAD_ID_AES_GCM_128` }

Functions

- int `hpke_ah_decode` (size_t ahlen, const char *ah, size_t *blen, unsigned char **buf)
decode ascii hex to a binary buffer
- int `hpke_enc` (unsigned int mode, `hpke_suite_t` suite, size_t publen, unsigned char *pub, size_t clearlen, unsigned char *clear, size_t aadlen, unsigned char *aad, size_t infolen, unsigned char *info, size_t *senderpublen, unsigned char *senderpub, size_t *cipherlen, unsigned char *cipher)
- int `hpke_dec` (unsigned int mode, `hpke_suite_t` suite, size_t privlen, unsigned char *priv, size_t enclen, unsigned char *enc, size_t cipherlen, unsigned char *cipher, size_t aadlen, unsigned char *aad, size_t *clearlen, unsigned char *clear)

4.2.1 Detailed Description

This has the data structures and prototypes (both internal and external) for an OpenSSL-based HPKE implementation following draft-irtf-cfrg-hpke.

I plan to use this for my ESNi-enabled OpenSSL build when the time is right, that's: <https://github.com/sftcd/openssl>

4.2.2 Function Documentation

4.2.2.1 hpke_ah_decode()

```
int hpke_ah_decode (
    size_t ahlen,
    const char * ah,
    size_t * blen,
    unsigned char ** buf )
```

decode ascii hex to a binary buffer

Parameters

<i>ahlen</i>	is the ascii hex string length
<i>ahstr</i>	is the ascii hex string
<i>blen</i>	is a pointer to the returned binary length
<i>buf</i>	is a pointer to the internally allocated binary buffer

Returns

zero for error, 1 for success

4.2.2.2 hpke_enc()

```
int hpke_enc (
    unsigned int mode,
    hpke_suite_t suite,
    size_t publen,
    unsigned char * pub,
    size_t clearlen,
    unsigned char * clear,
    size_t aadlen,
    unsigned char * aad,
    size_t info,
    unsigned char * info,
    size_t * senderpublen,
```

```

    unsigned char * senderpub,
    size_t * cipherlen,
    unsigned char * cipher )

```

< Our error return value - 1 is success

4.3 hpkemain.c File Reference

An OpenSSL-based HPKE implementation following draft-irtf-cfrg-hpke.

```

#include <stddef.h>
#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>
#include <string.h>
#include <getopt.h>
#include <ctype.h>
#include <openssl/evp.h>
#include <openssl/ssl.h>
#include "hpke.h"

```

Include dependency graph for hpkemain.c:

Macros

- #define **HPKE_START_SP** "-----BEGIN SENDERPUB-----"
- #define **HPKE_END_SP** "-----END SENDERPUB-----"
- #define **HPKE_START_CP** "-----BEGIN CIPHERTEXT-----"
- #define **HPKE_END_CP** "-----END CIPHERTEXT-----"
- #define **FINDLAB**(buf, lab, labptr)

Functions

- int [main](#) (int argc, char **argv)
hey it's [main\(\)](#)

4.3.1 Detailed Description

An OpenSSL-based HPKE implementation following draft-irtf-cfrg-hpke.

I plan to use this for my ESNI-enabled OpenSSL build (<https://github.com/sftcd/openssl>) when the time is right.

4.3.2 Macro Definition Documentation

4.3.2.1 FINDLAB

```
#define FINDLAB(
    buf,
    lab,
    labptr )
```

Value:

```
{ \
    labptr=strstr(buf,lab); \
    if (!labptr) { \
        fprintf(stderr,"Error can't find boundary (%s) in file  %s\n",lab,pfname); \
        return(__LINE__); \
    } \
}
```

4.4 hpktv.c File Reference

Implementation related to test vectors for HPKE.

4.4.1 Detailed Description

Implementation related to test vectors for HPKE.

This is compiled in if TESTVECTORS is #define'd, otherwise not.

The overall plan with test vectors is to:

- define data structures here to store the test vectors
- have global variables with the actual data
- have a #ifdef'd command line argument to generate/check a test vector
- have #ifdef'd additional parameters to _enc/_dec functions for doing generation/checking

Source for test vectors is: <https://raw.githubusercontent.com/cfrg/draft-irtf-cfrg-hpke/master/test-vectors.json> A copy from 20191126 is also in this repo in test-vectors.json

4.5 hpktv.h File Reference

Header file related to test vectors for HPKE.

4.5.1 Detailed Description

Header file related to test vectors for HPKE.

This is compiled in if TESTVECTORS is #define'd, otherwise not.

The overall plan with test vectors is to:

- define data structures here to store the test vectors
- have global variables with the actual data
- have a #ifdef'd command line argument to generate/check a test vector
- have #ifdef'd additional parameters to _enc/_dec functions for doing generation/checking

Source for test vectors is: <https://raw.githubusercontent.com/cfrg/draft-irtf-cfrg-hpke/master/test-vectors.json> A copy from 20191126 is also in this repo in test-vectors.json

Index

FINDLAB

hpke_main.c, [10](#)

hpke.c, [7](#)

hpke.h, [7](#)

hpke_ah_decode, [9](#)

hpke_enc, [9](#)

hpke_ah_decode

hpke.h, [9](#)

hpke_enc

hpke.h, [9](#)

hpke_suite_t, [5](#)

hpke_main.c, [10](#)

FINDLAB, [10](#)

hpke_tv.c, [11](#)

hpke_tv.h, [11](#)