libHMAC Documentation

Release 1.0.0

ANSSI

May 17, 2019

CONTENTS

Contents

CHAPTER

ONE

OVERVIEW

The libhmac project aim to implement a HMAC (Hash-based Message Authentication Code) userspace library.

This library also supports the PBKDF2 key derivation function.

This library is full software and does not make use of underlying hardware acceleration. Future work include adding such acceleration modes.

This library uses the external libecc library (https://github.com/ANSSI-FR/libecc) as a building block that provides all the hash functions software implementations.

1.1 Principles

HMAC is a specific MAC function which involves a cryptographic hash function and a cryptographic secret key in order to authenticate data and/or check data integrity.

HMAC design is described here:

https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.198-1.pdf

PBKDF2 key derivation is described here:

https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-132.pdf

CHAPTER

TWO

API

2.1 The HMAC functional API

2.1.1 Initializing the HMAC context

Initialize the HMAC context is made through two main functions:

```
#include "hmac.h"

typedef struct {
     const hash_mapping *hash;
     hash_context in_ctx;
     hash_context out_ctx;
} hmac_context;

int hmac_init(hmac_context *ctx, const uint8_t *hmackey, uint32_t hmackey_len, hash_
↪alg_type hash_type);
```

This library does not require any early_init step as the HMAC implementation is full software.

The HMAC initialization function uses the following arguments:

- **ctx**: the HMAC context. The fields of this context structure are initialized by this function
- **hmackey**: the HMAC secret key
- **hmackey_len**: the HMAC secret key len (in bytes)
- **hash_type**: the HASH algorithm type. This type is one of the supported libecc hash algorithms

Hint: The hash_type is typically one of SHA224, SHA256, SHA384, SHA512, SHA3_224...

Danger: The list of supported hash algorithm depends on the libecc compilation flags

The initialization function returns 0 on SUCCESS, or -1 on failure.

2.1.2 Hashing data

Hashing data can be done through successive calls to the libhmac API. Hashing data is done using the following API:

```
#include "hmac.h"

void hmac_update(hmac_context *ctx, const uint8_t *input, uint32_t ilen);
int hmac_finalize(hmac_context *ctx, uint8_t *output, uint32_t *outlen);
```

All successive hash requests of a given data flow is done using the `hmac_update()` function. The last call **must** be done using the `hmac_finalize()` function.

The HMAC context must be provided to the HMAC API as the libhmac does not keep the current context. This allows the user task to manipulate multiple contexts in the same time if needed.

The `hmac_finalize()` function returns 0 on success or -1 on failure.

2.1.3 Generate HMAC_based derived keys

The libhmac supports PBKDF2 password hash-based derivation function to generate derived keys.

Requesting a PBKDF2 computation is done using the following API:

```
#include "hmac.h"

int hmac_pbkdf2(      hash_alg_type  hash_type,
                const uint8_t       *password,
                      uint32_t       password_len,
                const uint8_t       *salt,
                      uint32_t       salt_len,
                      uint32_t       c,
                      uint32_t       dklen,
                      uint8_t       *output,
                      uint32_t      *outlen);
```