The Libgcrypt Reference Manual

Version 1.1.5

The Libgcrypt Reference Manual: Version 1.1.5

Copyright © 2000 by Free Software Foundation, Inc.

Please direct questions, bug reports, or suggestions concerning this manual to the mailing list `<gnupg-doc@gnupg.org>`.

This manual may be redistributed under the terms of the GNU General Public License (http://www.gnu.org/copyleft/gpl.html).

Table of Contents

I. Libgcrypt Reference Pages

gcry_cipher_open

Name

`gcry_cipher_open, gcry_cipher_close` — create and destroy an encryption context

Synopsis

```
#include <gcrypt.h>

GCRY_CIPHER_HD gcry_cipher_open(int algo, int mode, unsigned in flags);
void gcry_cipher_close(GCRY_CIPHER_HD h);
```

Description

`gcry_cipher_open` creates the context required for most of the cipher functions.

gcry_cipher_ctl

Name

`gcry_cipher_ctl, gcry_cipher_setkey, gcry_cipher_setiv, gcry_cipher_setiv` — control cipher functions

Synopsis

```
#include <gcrypt.h>

int gcry_cipher_ctl(GCRY_CIPHER_HD h, int cmd, void *buffer, size_t buflen);
```

Description

`gcry_cipher_ctl` controls various aspects of the cipher module and specific cipher contexts. A couple of macros may be used for convenience: gcry_cipher_setkey(h,k,l) gcry_cipher_setiv(h,k,l) gcry_cipher_sync(h)

gcry_cipher_info

Name

`gcry_cipher_info` — return information about a cipher context

Synopsis

```
#include <gcrypt.h>
```

`int` **`gcry_cipher_info`**`(GCRY_CIPHER_HD` *`h`*`, int` *`what`*`);`

Description

`gcry_cipher_info` is used to retrieve various information about a cipher context or the cipher module in general. Currently no information is available.

gcry_cipher_algo_name

Name

`gcry_cipher_algo_name` — return the name of a cipher algorithm

Synopsis

```
#include <gcrypt.h>
```

```
const char *gcry_cipher_algo_name(int algo);
```

Description

`gcry_cipher_algo_name` returns a string with the name of the cipher algorithm *algo*. If the algorithm is not known or an other error occured, an empty string is return. This function will never return NULL.

gcry_cipher_map_name

Name

`gcry_cipher_map_name` — return an algorithm identifier for string

Synopsis

```
#include <gcrypt.h>

int gcry_cipher_map_name(const char *name);
```

Description

`gcry_cipher_map_name` returns the algorithm identifier for the cipher algorithm described by the string *name*. If this algorith is not available 0 is returned.

gcry_cipher_encrypt

Name

`gcry_cipher_encrypt, gcry_cipher_decrypt` — encrypt data or decrypt data

Synopsis

```
      #include <gcrypt.h>

int gcry_cipher_encrypt(GCRY_CIPHER_HD h, unsigned char out, size_t outsize,
unsigned char in, size_t insize);
int gcry_cipher_decrypt(GCRY_CIPHER_HD h, unsigned char out, size_t outsize,
unsigned char in, size_t insize);
```

Description

`gcry_cipher_encrypt` is used to encrypt the data. the complemetary function `gcry_cipher_decrypt` is be used to decrypt the data. The calling convention for both functions is the same.

These functions can either work in place or with two buffers. Overlapping buffers are not allowed.

gcry_md_open

Name

`gcry_md_open, gcry_md_enable, gcry_md_close` — create and destroy a message digest context

Synopsis

```
      #include <gcrypt.h>

GCRY_MD_HD gcry_md_open(int algo, unsigned int flags);
void gcry_md_enable(GCRY_MD_HD h, int algo);
void gcry_md_close(GCRY_MD_HD h);
```

Description

`gcry_md_open` creates the context required for the message digest functions. The hash algorithm may optionally be specified. It is possible to use these functions as MAC functons; therefore the flag `GCRY_MD_FLAG_HMAC` must be given along with the hash functions. Other MAC algorithms than HMAC are currently not supported. The key for the MAC must be set using the `gcry_md_setkey` function. `gcry_md_close` releases all resources associated with the context. `gcry_md_enable` may be used to enable hash algorithms. This function may be used multiple times to create a hash context for multiple algorithms. Adding an already enabled algorithm has no effect. A algorithm must be enabled prios to calculate hash algorithms.

gcry_md_copy

Name

`gcry_md_copy` — create and copy of a message digest context

Synopsis

```
#include <gcrypt.h>
```

```
GCRY_MD_HD gcry_md_copy(GCRY_MD_HD h);
```

Description

`gcry_md_copy` creates an excat copy of the given context. This is useful to calculate hashes with a common initial part of the plaintext.

gcry_md_reset

Name

`gcry_md_reset` — reset a message digest context

Synopsis

```
#include <gcrypt.h>
```

```
void gcry_md_reset(GCRY_MD_HD h);
```

Description

`gcry_md_reset` is used to reuse a message context without the overhead of an open and close operation.

gcry_md_ctl

Name

`gcry_md_ctl, gcry_md_final, gcry_md_setkey` — perform special operations on a digest context

Synopsis

```
#include <gcrypt.h>
```

```
int gcry_md_ctl(GCRY_MD_HD h, int cmd, unsigned char * buffer, size_t
buflen);
```

Description

`gcry_md_ctl` is a multi-purpose function to control the behaviour of all gcry_md functions or one instance of it.

Currently defined values for *cmd* are:

`GCRYCTL_FINALIZE` and the convenience macro `gcry_md_final(a)`

`GCRYCTL_SET_KEY` and the convenience macro `gcry_md_setkey(a)`. This is used to turn these hash functions into MAC functions. The key may be any string of the speicified length. The type of the MAC is determined by special flags set with the open function. NEW: There is now a function to do this

gcry_md_write

Name

`gcry_md_write, gcry_md_putc` — calculate the message digest of a buffer

Synopsis

```
    #include <gcrypt.h>

int gcry_md_write(GCRY_MD_HD h, unsigned char * buffer, size_t buflen);
int gcry_md_putc(GCRY_MD_HD h, int c);
```

Description

`gcry_md_write` is used to actually calulate the message digest of a buffer. This function updates the internal state of the message digest.

`gcry_md_putc` is a macro which is used to update the message digest by one byte. this is the preferred way to calculate a digest if only a few bytes at a time are available.

gcry_md_read

Name

`gcry_md_read` — read out the message digest

Synopsis

```
    #include <gcrypt.h>

unsigned char * gcry_md_read(GCRY_MD_HD h, int algo);
```

Description

`gcry_md_read` returns the message digest after finalizing the calculation. This function may be used as often as required but it will alwas return the same value for one handle. The returned message digest is allocated within the message context and therefore valid until the conext is released. *`algo`* may be given as 0 to return the only enbaled message digest or it may specify one of the enabled algorithms. The function will return NULL if the algorithm has not been enabled.

gcry_md_info

Name

`gcry_md_info` — get information about a handle

Synopsis

```
#include <gcrypt.h>

int gcry_md_info(GCRY_MD_HD h, int what, void * buffer, size_t buflen);
```

Description

`gcry_md_info` returns some information about the handle or an global parameter.

The only defined value for *`what`* is `GCRYCTL_IS_SECURE` to return whether the handle has been allocated in secure memory. Buffer and buflen are not needed in this cases. The convenience macro `gcry_md_is_secure(a)` may be also used for this purpose.

gcry_md_algo_info

Name

`gcry_md_algo_info, gcry_md_test_algo, gcry_md_get_algo_dlen` — get information about an algorithm

Synopsis

```
#include <gcrypt.h>
```

`int` **`gcry_md_algo_info`**`(int` *`algo`*`, int` *`what`*`, void *` *`buffer`*`, size_t` *`buflen`*`);`
`unsigned int` **`gcry_md_get_algo_dlen`**`(int` *`algo`*`);`

Description

`gcry_md_algo_info` returns some information about an algorithm. On error the value -1 is return and a more detailed error description is available with `gcry_errno`.

The defined values for *`what`* are `GCRYCTL_TEST_ALGO` to return whether the algorithm is supported. Buffer and buflen are not needed in this cases. The convenience macro `gcry_md_test_algo(a)` may be used for this purpose.

`GCRYCTL_GET_ASNOID` to return whether the ASN.1 object identifier. IF buffer is specified as NULL, only the required length for the buffer is returned.

`gcry_md_get_algo_dlen` returns the length of the digest for a given algorithm in bytes.

gcry_md_algo_name

Name

`gcry_md_algo_name, gcry_md_map_name` — map algorithm to string

Synopsis

```
#include <gcrypt.h>
```

`const char *` **`gcry_md_algo_name`**`(int` *`algo`*`);`
`int` **`gcry_md_map_name`**`(const char*`*`name`*`);`

Description

These both functions are used to map a string with the algorithm name to the internal algorithm identifier value and vice versa.

`gcry_md_algo_name` never returns NULL even in cases where the algorithm string is not available. Instead a string consisting of a single question mark is returned. Do not use this function to test for the existence of an algorithm.

`gcry_md_map_name` returns 0 if the algorithm is not known to Libgcrypt.

gcry_md_hash_buffer

Name

`gcry_md_hash_buffer` — fast message calculation

Synopsis

```
#include <gcrypt.h>
```

```
int gcry_md_hash_buffer(int algo, char * digest, const char * buffer, size_t
buflen);
```

Description

`gcry_md_hash_buffer` is a shortcut function to calculate a message digest of a buffer. This function does not require a context and immediatley returns the message digest. `digest` must be string large enough to hold the digest given by algo. This length may be obtained by using the function `gcry_md_get_algo_dlen` but in most cases it will be a statically allocated buffer.

gcry_pk_encrypt

Name

`gcry_pk_encrypt` — do a public key encryption

Synopsis

```
#include <gcrypt.h>
```

`int gcry_pk_encrypt(GCRY_SEXP *result, GCRY_SEXP data, GCRY_SEXP pkey);`

Description

`gcry_pk_encrypt` performs public key encryption operations. The caller has to provide a public key as the S-Exp *pkey* and *data* as a S-Exp with just one MPI in it. The function returns a S-Exp which may be passed tp to pk_decrypt. Later versions of this functions may take more complex input data. Returns: 0 or an errorcode.

```
s_data = (mpi)
s_pkey = key-as-defined-in-sexp_to_key
r_ciph = (enc-val
  (algo
   (param_name1 mpi)
   ...
   (param_namen mpi)
  ))
```

gcry_check_version

Name

`gcry_check_version` — get or check the version of libgcrypt

Synopsis

```
#include <gcrypt.h>
```

`const char *gcry_check_version(const char *req_version);`

Description

`gcry_check_version` checks that the version of the library is at minimum the requested one and return the version string; NULL is returned if the condition is not met. You may pass NULL as reqy_version to simply get the version string back without any checking.

gcry_errno

Name

`gcry_errno, gcry_strerror` — Get the last error

Synopsis

```
      #include <gcrypt.h>

int gcry_errno();
const char *gcry_strerror(intno);
```

Description

Both function are to be used like there Standard-C counterparts. However `gcry_errno` is a function and not just a global variable. If -1 is passed to `gcry_strerror`, `gcry_errno` is implictly used.

gcry_control

Name

`gcry_control` — Multi purpose control function

Synopsis

```
      #include <gcrypt.h>
```

```
int gcry_control(enum gcry_ctl_cmdscmd, ...);
```

Description

This function is used to control various aspects of Libgcrypt FIXME: Explain all commands here.

gcry_set_allocation_handler

Name

`gcry_set_allocation_handler, gcry_set_outofcore_handler` — Use application defined malloc functions

Synopsis

```
#include <gcrypt.h>

void gcry_set_allocation_handler(void *(*alloc_func)(size_t n), void
*(*alloc_secure_func)(size_t n), int (*is_secure_func)(const void *p), void
*(*realloc_func)(void *p, size_t n), void (*free_func)(void *p));
void gcry_set_outofcore_handler(int (*h)( void*, size_t, unsigned int ), void
*opaque ));
```

Description

FIXME

gcry_set_fatalerror_handler

Name

`gcry_set_fatalerror_handler` — change the default fatal error handler

Synopsis

```
    #include <gcrypt.h>

void gcry_set_fatalerror_handler(void (*func)( void *, int, const char*),
void *opaque);
```

Description

At certain places the Libgcrypt may need to call a fatal error fucntion which does terminate the process. To allow an application to do some emergency cleanup, it may register a fatal error handler with the library. This handler is assumed to terminate the application; however if it returns Libgcrypt will abort anyway.

The handler is called with the opaque value registered here, an errorcode from Libgcrypt and some descriptive text string.

gcry_set_gettext_handler

Name

`gcry_set_gettext_handler` — Change the default gettext function

Synopsis

```
    #include <gcrypt.h>

void gcry_set_gettext_handler(const char *(*func)(const char*), void
*opaque);
```

Description

FIXME!!

gcry_set_log_handler

Name

`gcry_set_log_handler` — Change the default logging function

Synopsis

```
      #include <gcrypt.h>

void gcry_set_log_handler(void (*func) (void*, int, const char*, va_list),
void *opaque);
```

Description

Libgcrypt has it;s own logging functions. Applications which need to use their own, should provide a log function to Libgcrypt so that it will use this function instead. Fixme: Describe how this is intended to work.

gcry_malloc

Name

`gcry_malloc`, `gcry_calloc`, `gcry_malloc_secure`, `gcry_calloc_secure`, `gcry_realloc`, `gcry_xmalloc`, `gcry_xcalloc`, `gcry_xmalloc_secure`, `gcry_xcalloc_secure`, `gcry_xrealloc`, `gcry_xstrdup`, `gcry_malloc`, `gcry_malloc` — Change the default logging function

Synopsis

```
      #include <gcrypt.h>

void gcry_set_log_handler(void (*func) (void*, int, const char*, va_list),
void *opaque);
```

Description

Libgcrypt has it;s own logging functions. Applications which need to use their own, should provide a log function to Libgcrypt so that it will use this function instead. Fixme: Describe how this is intended to work.