NAME

RIPEMD160_Init, **RIPEMD160_Update**, **RIPEMD160_Final**, **RIPEMD160_End**, **RIPEMD160_File**, **RIPEMD160_FileChunk**, **RIPEMD160_Data** **-** calculate the RIPEMD160 message digest

LIBRARY

Message Digest (MD4, MD5, etc.) Support Library (libmd, -lmd)

SYNOPSIS

#include <sys/types.h>
#include <ripemd.h>

void
RIPEMD160_Init(*RIPEMD160_CTX *context*);

void
RIPEMD160_Update(*RIPEMD160_CTX *context*, *const unsigned char *data*, *unsigned int len*);

void
RIPEMD160_Final(*unsigned char digest[20]*, *RIPEMD160_CTX *context*);

*char **
RIPEMD160_End(*RIPEMD160_CTX *context, char *buf*);

*char **
RIPEMD160_File(*const char *filename, char *buf*);

*char **
RIPEMD160_FileChunk(*const char *filename*, *char *buf*, *off_t offset*, *off_t length*);

*char **
RIPEMD160_Data(*const unsigned char *data, unsigned int len, char *buf*);

DESCRIPTION

The RIPEMD160_ functions calculate a 160-bit cryptographic checksum (digest) for any number of input bytes. A cryptographic checksum is a one-way hash function; that is, it is computationally impractical to find the input corresponding to a particular output. This net result is a "fingerprint" of the input-data, which does not disclose the actual input.

The **RIPEMD160_Init**(), **RIPEMD160_Update**(), and **RIPEMD160_Final**() functions are the core functions. Allocate an *RIPEMD160_CTX*, initialize it with **RIPEMD160_Init**(), run over the data with

RIPEMD160_Update(), and finally extract the result using **RIPEMD160_Final**(), which will also erase the *RIPEMD160_CTX*.

The **RIPEMD160_End**() function is a wrapper for **RIPEMD160_Final**() which converts the return value to a 41-character (including the terminating '\0') ASCII string which represents the 160 bits in hexadecimal.

The **RIPEMD160_File**() function calculates the digest of a file, and uses **RIPEMD160_End**() to return the result. If the file cannot be opened, a null pointer is returned. The **RIPEMD160_FileChunk**() function is similar to **RIPEMD160_File**(), but it only calculates the digest over a byte-range of the file specified, starting at *offset* and spanning *length* bytes. If the *length* parameter is specified as 0, or more than the length of the remaining part of the file, **RIPEMD160_FileChunk**() calculates the digest from *offset* to the end of file. The **RIPEMD160_Data**() function calculates the digest of a chunk of data in memory, and uses **RIPEMD160_End**() to return the result.

When using **RIPEMD160_End**(), **RIPEMD160_File**(), or **RIPEMD160_Data**(), the *buf* argument can be a null pointer, in which case the returned string is allocated with malloc(3) and subsequently must be explicitly deallocated using free(3) after use. If the *buf* argument is non-null it must point to at least 41 characters of buffer space.

ERRORS

The **RIPEMD160_End**() function called with a null buf argument may fail and return NULL if:

[ENOMEM]	Insufficient storage space is available.

The **RIPEMD160_File**() and **RIPEMD160_FileChunk**() may return NULL when underlying open(2), fstat(2), lseek(2), or RIPEMD160_End(3) fail.

SEE ALSO

md4(3), md5(3), sha(3), sha256(3), sha512(3), skein(3)

HISTORY

These functions appeared in FreeBSD 4.0.

AUTHORS

The core hash routines were implemented by Eric Young based on the published RIPEMD160 specification.

BUGS

No method is known to exist which finds two files having the same hash value, nor to find a file with a specific hash value. There is on the other hand no guarantee that such a method does not exist.