TN209

Using the MD5 Hash Library

This technical note describes the Message Digest version 5 (MD5) hashing algorithm. MD5 is a one-way hash algorithm that addresses two main concerns that are created when communicating over a network: authenticity and data integrity. MD5 is fast and simple, yet offers a higher level of security than MD4 and is much more reliable than a checksum.

A Hashing Algorithm

A hashing algorithm transforms an arbitrarily long block of data into a large number. This number (called the hash value, or just hash) has a few useful properties:

- It has no correlation to the original data, and nothing about the original data can be inferred from it.
- Small changes in the original data produce large, essentially random, changes in the hash value.
- Generated hash values are evenly dispersed throughout the space of possible values (i.e., all possible values are equally likely to occur).

Uses of MD5

The MD5 algorithm takes a block of data of arbitrary length and produces a 16-byte hash. For additional security, a block of data can be broken into smaller blocks that are hashed separately. The individual hash values are then put together and hashed as a final step.

MD5 hashes can be used to verify the integrity of a block of data. Hashing data and comparing it with a previously calculated hash value will determine if that data has been changed. An MD5 hash is better than a checksum because it will detect transmission errors that a checksum would not be able to detect.

MD5 hashes can be used to verify the source of the transmitted data. This is done by generating the MD5 hash of a message concatenated with a secret password, and then transmitting the message and the hash. A receiver can then hash the message with the password and see if the hash values match.

MD5 is a one-way hashing algorithm i.e., it does not encrypt data.

Example

The following program, **`md5_test.c`**, is available in **`\Samples\tcpip`**. It demonstrates the MD5 hashing library.

```
#use "md5.lib"
const char string_a[] = "Buy low, sell high.";
const char string_b[] = "Buy low, sell high?";
md5_state_t hash_state;
char hash[16];
void hexprint(char *data, int len);
main()
{
   md5_init(&hash_state);                     // prepare for a new hash
   md5_append(&hash_state, string_a, strlen(string_a));
   md5_finish(&hash_state, hash);             // calculate hash value

   printf("%s\n", string_a);
   printf("--hashes to---\n");
   hexprint(hash, 16);
   printf("\n");
   md5_init(&hash_state);                     // prepare for a new hash
   md5_append(&hash_state, string_b, strlen(string_b));
   md5_finish(&hash_state, hash);             // calculate hash value

   printf("%s\n", string_b);
   printf("--hashes to---\n");
   hexprint(hash, 16);
   printf("\n");
}
void hexprint(char *data, int len)
{
   auto int i;
   for(i = 0;i < len;i++)
   {
   /* "%02x" for lowercase, "%02X" for uppercase hexidecimal letters */
      printf("%02x", data[i]);
   }
}
```

MD5 Library Functions

There are three API functions in the MD5 library (**`MD5.LIB`**). Each function takes as an argument a pointer to a structure of type **`md5_state_t`**. This structure must be instantiated by the user.

md5_init

```
void md5_init(md5_state_t *pms);
```

DESCRIPTION

Initialize the MD5 hash process. Initial values are generated for the structure, and this structure will identify a particular transaction in all subsequent calls to the md5 library.

PARAMETER

`pms`	Pointer to the **`md5_state_t`** structure.

md5_append

```
void md5_append(md5_state_t *pms, char *data, int nbytes);
```

DESCRIPTION

This function will take a buffer and compute the MD5 hash of its contents, combined with all previous data passed to it. This function can be called several times to generate the hash of a large amount of data

PARAMETERS

`pms`	Pointer to the **`md5_state_t`** structure that was initialized by **`md5_init`**.
`data`	Pointer to the data to be hashed.
`nbytes`	Length of the data to be hashed.

md5_finish

```
void md5_finish(md5_state_t *pms, char digest[16]);
```

DESCRIPTION

Completes the hash of all the received data and generates the final hash value.

PARAMETERS

pms Pointer to the **md5_state_t** structure that was initialized by **md5_init.**

digest The 16-byte array that the hash value will be written into.

Z-World, Inc.

2900 Spafford Street
Davis, California 95616-6800
USA

Telephone: (530) 757-3737
Fax: (530) 757-3792

www.zworld.com

Rabbit Semiconductor

2932 Spafford Street
Davis, California 95616-6800
USA

Telephone: (530) 757-8400
Fax: (530) 757-8402

www.rabbitsemiconductor.com