信息安全期中大作业

- 完成一个HMAC-MD5算法的程序设计和实现，包括
  - MD5 算法原理概述; HMAC 算法原理概述;总体结构设计;模 块分解;数据结构设计;C 语言源代码;编译运行结果;验证用例。

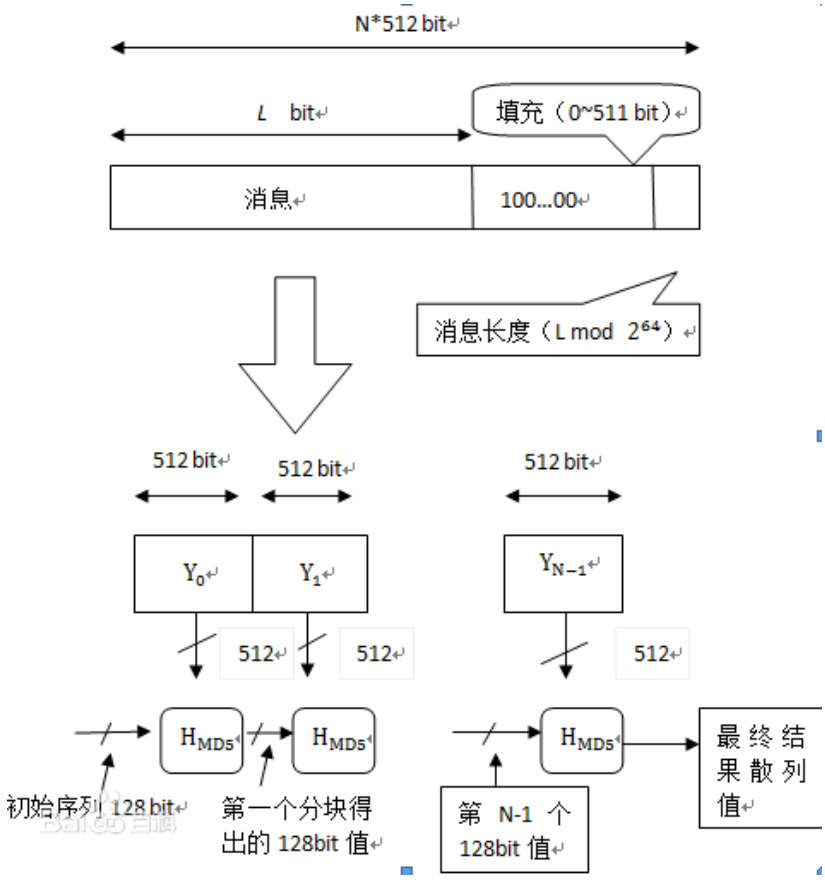
算法原理描述

MD5算法原理概述

MD5即Message-Digest Algorithm5，是广泛使用的Hash算法，用于确保信息传输的完整性和一致性。MD5使用小端模式，输入任意不定长度信息，以512-bit进行分组，生成四个32-bit数据，最后联合输出固定128-bit的信息摘要。MD5算法的基本过程为：填充、分块、缓冲区初始化、循环压缩、得出结果。

MD5算法的原理可简要的叙述为：MD5码以512位分组来处理输入的信息，且每一分组又被划分为16个32位子分组，经过了一系列的处理后，算法的输出由四个32位分组组成，将这四个32位分组级联后将生成一个128位散列值。

总体流程如下图所示，每次的运算都由前一轮的128位结果值和当前的512bit值进行运算



HMAC算法原理概述

HMAC即Hash-based Message Authentication Code，是一种基于Hash函数和密钥进行消息认证的方法。

在HMAC的定义中用到一个密码散列函数H和一个密钥K。假设H是一个能够对明文进行分组循环压缩的散列函数，B为散列函数的明文分组长度（byte），在上述的散列函数中B=64，L为散列函数的输出长度（byte），MD5中L=16，SHA-1中L=20。认证密钥K可以为任意长度，一般密钥长度应大于明文分组的长度，将密钥的第一次散列值作为HMAC真正使用的密钥，密钥的最小推荐长度为Lbytes。

再定义两个不同的固定字符串ipad和opad如下（“i”和“o”表示内部和外部）：

ipad=一个字节（byte）的0x36重复B次；

opad=一个字节（byte）的0x5C重复B次。

若以“text”作为要计算HMAC的明文，则作如下操作：

H（K XOR opad，H（K XOR ipad，text））

也就是说，操作步骤如下：

（1）在密钥K后面填充0，使其成为长度为Bbyte的字符串；如：K是20bytes的字符串，B=64，则要填充44个字节的0x00。

（2）用第一步得到的Bbyte的字符串与ipad[XOR；

（3）将数据流text附加到第（2）步产生的Bbyte字符串后面；

（4）对第（3）产生的数据流用散列函数H计算消息摘要；

（5）用第一步得到的Bbyte的字符串与opad作XOR（按位异或）；

（6）将第（4）生成的消息摘要附加到第（5）步的Bbyte字符串之后；

（7）对第（6）产生的数据流用散列函数H计算消息摘要，作为输出。

实验设计

总体结构设计

测试文件：main.c

MD5相关：md5.h、md5.c

HMAC-MD5：hmac_md5.h、hmac_md5.c

使用HMAC-MD5的接口为：hmac_md5.h中提供的

```
unsigned char* hmac_md5( unsigned char* message, unsigned char* key);
```

模块分解

- MD5
  - 一些数据

```
#define S11 7
#define S12 12
#define S13 17
#define S14 22
#define S21 5
#define S22 9
#define S23 14
#define S24 20
#define S31 4
#define S32 11
#define S33 16
#define S34 23
#define S41 6
#define S42 10
#define S43 15
#define S44 21

// ABCD
context->reg[0] = 0x67452301;
context->reg[1] = 0xEFCDAB89;
context->reg[2] = 0x98BADCFE;
context->reg[3] = 0x10325476;
```

  - 一些函数

```
/* 四个基础线性函数
 */
#define F(x, y, z) (((x) & (y)) | ((~x) & (z)))
#define G(x, y, z) (((x) & (z)) | ((y) & (~z)))
#define H(x, y, z) ((x) ^ (y) ^ (z))
#define I(x, y, z) ((y) ^ ((x) | (~z)))

/* ROTATE_LEFT 将x左移n个bits
 */
#define ROTATE_LEFT(x, n) (((x) << (n)) | ((x) >> (32-(n))))

/* 四个数据处理函数 FF, GG, HH, II , xj表示第j个数
 */
#define FF(a, b, c, d, xj, s, ac) { \
 (a) += F ((b), (c), (d)) + (xj) + (UINT4)(ac); \
 (a) = ROTATE_LEFT ((a), (s)); \
 (a) += (b); \
```

```
  }
#define GG(a, b, c, d, xj, s, ac) { \
 (a) += G ((b), (c), (d)) + (xj) + (UINT4)(ac); \
 (a) = ROTATE_LEFT ((a), (s)); \
 (a) += (b); \
  }
#define HH(a, b, c, d, xj, s, ac) { \
 (a) += H ((b), (c), (d)) + (xj) + (UINT4)(ac); \
 (a) = ROTATE_LEFT ((a), (s)); \
 (a) += (b); \
  }
#define II(a, b, c, d, xj, s, ac) { \
 (a) += I ((b), (c), (d)) + (xj) + (UINT4)(ac); \
 (a) = ROTATE_LEFT ((a), (s)); \
 (a) += (b); \
  }
```

- MD5接口：

```
void MD5_init(MD5_INFO *);
void MD5_update(MD5_INFO * , unsigned char *, UINT4 );
void MD5_final(MD5_INFO *, unsigned char [16]);
void MD5_transform(unsigned int [4],unsigned char [64]);
void MD5_encode(unsigned char *, UINT4 *, UINT4 );
void MD5_decode(UINT4 *, unsigned char *, UINT4 );
```

- HMAC-MD5

```
unsigned char* hmac_md5( unsigned char* message, unsigned char* key);
```

C语言源代码

在当前目录下的src文件夹内。

测试运行

编译运行结果

```
------- Test1 -------
Msg: Hello world
Key: key
HMAC_MD5: 4650537cb9041dc3a95b7d30415a1eda
```

```
------- Test2 -------
Msg: QWERTYUIOPASDFGHJKLZXCVBNMQWERTYUIOPSDFGHJKZXVBNMASDFGHJKLqwertyui
Key:
12345678901234567890123456789012345678901234567890123456789012345678901234567890
0
HMAC_MD5: 1a141b36fcd3cd83716689bc94130ebf

------- Test3 -------
Msg:
12345678901234567890123456789012345678901234567890123456789012345678901234567890
0
Key: QWERTYUIOPASDFGHJKLZXCVBNMQWERTYUIOPSDFGHJKZXVBNMASDFGHJKLqwertyui
HMAC_MD5: b32cad610612420884114a7df7ae96c7
```

经过这个网站测试：https://tool.oschina.net/encrypt?type=2，结果正确。

明文:

QWERTYUIOPASDFGHJKLZXCVBNMQWERTYUIOPSDFGHJKZXVBNMASDFGHJKLqwertyui

散列/哈希算法:

SHA1 | SHA224 | SHA256 | SHA384 | SHA512 | MD5

HmacSHA1 | HmacSHA224 | HmacSHA256 | HmacSHA384 | HmacSHA512 | HmacMD5

密钥 12345678901234567890123456789 哈希/散列

哈希值:

1a141b36fcd3cd83716689bc94130ebf

123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890

散列/哈希算法:

SHA1 | SHA224 | SHA256 | SHA384 | SHA512

HmacSHA1 | HmacSHA224 | HmacSHA256 | HmacSHA384 | HmacSHA512

密钥 QWERTYUIOPASDFGHJKLZXCVBN 哈希/散列

哈希值:

b32cad610612420884114a7df7ae96c7