组织：中国互动出版网（http://www.china-pub.com/）

RFC 文档中文翻译计划

（http://www.china-pub.com/compters/emook/aboutemook.htm）

E-mail：ouyang@china-pub.com

译者：（未名 — 向你致敬）

译文发布时间：2001-11-7

修改： Hinker Liu

Email：hinkerliu@163.com

修改时间：2011-01-22

重新编辑说明：

原翻译在 《第三节 MD5 算法描述》 的 《3.4 Step 4：处理 16 个字的数据》中和 《第五节 MD4 和 MD5 的区别》 中缺少部分翻译，在网上能找到的所有版本中都没有一个是全的，现已补全这两部分。

该版本还参考了谢超（http://hi.baidu.com/airxiechao）的《MD5 消息摘要算法》部分翻译，以及张裔智、赵毅和汤小斌的《MD5 算法研究》（计算机科学 2008V01．35№．7）。

Network Working Group R. Rivest

Request for Comments: 1321 MIT Laboratory for Computer Science

and RSA Data Security, Inc.

April 1992

MD5 报文摘要算法

（The MD5 Message-Digest Algorithm）

本文地位

本文并非指定一个互联网标准，而是向互联网社区提供信息，本文可以任意传播，不受限制。

致谢：

Don Coppersmith, Burt Kaliski, Ralph Merkle, David Chaum, 和 Noam Nisan 向本文提供极大的帮助，在此本人表示忠心的感谢。

目录

1. 执行简介

本文描述了 MD5 报文摘要算法，此算法将对输入的任意长度的信息进行计算，产生一个 128 位长度的“指纹”或“报文摘要”，假定两个不同的文件产生相同的报文摘要或由给定的报文摘要产生原始信息在计算上是行不通的。MD5 算法适合用在数据签名应用中，在此应用中，一个大的文件必须在类似 RSA 算法的

公用密钥系统中用私人密钥加密前被“压缩”在一种安全模式下。

MD5 算法能在 32 位机器上能以很快的速度运行。另外，MD5 算法不需要任何大型的置换列表。此算法编码很简洁。

MD5 算法是 MD4 报文摘要算法的扩展。MD5 算法稍慢于 MD4 算法，但是在设计上比 MD4 算法更加“保守”。设计 MD5 是因为 MD4 算法被采用的速度太快，以至于还无法证明它的正确性，因为 MD4 算法速度非常快，它处在遭受成功秘密攻击的“边缘”。MD5 后退了一步，它舍弃了一些速度以求更好的安全性。它集中了不同的评论家提出的建议，并采取了一些附加的优化措施。它被放在公共的地方以求公众的评论意见，它可能当作一个标准被采纳。

作为基于 OSI 的应用，MD5 的对象标识符是：

md5 OBJECT IDENTIFIER ::=

iso(1) member-body(2) US(840) rsadsi(113549) digestAlgorithm(2) 5}

在 X.509 类型 AlgorithmIdentifier [3]中，MD5 算法参数应该包括 NULL 类型。

2. 术语和符号

本文中一个“字”是 32 位，一个“字节”是 8 位。一系列位串可看成是一系列字节的普通形式，其中的连续的 8 位看成一个字节，高位在前，同理一系列字节串可看成是一系列 32 位的字，其中每个连续的 4 个字节当作一个字，低位在前。

我们定义 x_i 代表“x 减去 i"。如果下划线右边的是一个表达式，则用括号括住，如：x_{i+1}。同样我们用^代表求幂，这样 x^i 则代表 x 的 i 次幂。

符号“＋”代表字的加，X <<< s 代表 32 位的值 X 循环左移 s 位，not(X)代表 X 的按位补运算，X v Y 表示 X 和 Y 的按位或运算，X xor Y 代表 X 和 Y 的按位异或运算，XY 代表 X 和 Y 的按位与运算。

3. MD5 算法描述

我们假设有一个 b 位长度的输入信息，希望产生它的报文摘要，此处 b 是一个非负整数，b 也可能是 0，不一定必须是 8 的整数倍，它可能是任意大的长度。我们设想信号的比特流如下所示：

m_0 m_1 … m_{b-1}

下面的 5 步计算信息的报文摘要。

3.1 第一步：添加填充位

MD5 算法是对输入的数据进行补位，使得如果数据位长度 LEN 对 512 求余的结果是 448。即数据扩展至 K*512+448 位。即 K*64+56 个字节，K 为整数。补位操作始终要执行，即使数据长度 LEN 对 512 求余的结果已是 448。

具体补位操作：补一个 1，然后补 0 至满足上述要求。总共最少要补一位，最多补 512 位。

3.2 第二步：补数据长度

用一个 64 位的数字表示数据的原始长度 b（信息的长度是添加填充位前的长度），将 b 添加到上一步的结果后。当遇到 b 大于 2^64 这种极少遇到的情况时，那么只取 b 的低 64 位。（这些位被添加为两个 32 位字，并按照以前的规定首先添加到到低位字。）

这时，数据就被填补成长度为 512 位的倍数（后面是填充位和长度 b）。也就是说，此时的数据长度是 16 个（32 位）字的整数倍数。用 M[0 ... N-1]表示此时的数据，其中的 N 是 16 的倍数。

3.3 第三步：初始化 MD 缓冲器

用一个四个字的缓冲器（A，B，C，D）来计算报文摘要，A,B,C,D 分别是 32 位的寄存器，初始化使用的是十六进制表示的数字

字 A：0X01234567

字 B：0X89abcdef

字 C：0Xfedcba98

字 D：0X76543210

3.4 第四步：处理 16 个字的数据

首先定义 4 个辅助函数，每个函数的输入是三个 32 位的字，输出是一个 32 位的字。

```
F(X,Y,Z) = XY v not(X) Z
G(X,Y,Z) = XZ v Y not(Z)
H(X,Y,Z) = X xor Y xor Z
```

```
I(X,Y,Z) = Y xor (X v not(Z))
```

F 是这样处理每一位的：如果 X=1，那么结果为 Y，否则结果为 Z。如果对应 X，Y，Z 中的每一位是独立、均匀的，那么结果中的每一位也是独立、均匀的。

函数 G，H，I 和函数 F 相似，因为他们采取“按位平行”，从 X，Y 和 Z 的位来产生它们的输出，在这样的方式，如果 X、Y 和 Z 的相应位是独立和均匀的，那么 G(X,Y,Z)、 H(X,Y,Z)和 I(X,Y,Z)的每一位也是独立和均匀的。请注意函数 H 是其输入函数的逐位“异或”或“奇偶校验”（Parity）。

这一步中使用一个 64 元素的常数组 T[1 ... 64]，它由 sin 函数构成，T[i]表示数组中的第 i 个元素，它的值等于经过 4294967296 次 abs(sin(i))后的值的整数部分（其中 i 是弧度 ）。T[i]为 32 位整数用 16 进制表示，数组元素在附录中给出。

具体过程如下：

```
/* 处理数据原文 */
For i = 0 to N/16-1 do
/*每一次，把数据原文存放在16个元素的数组x中. */
For j = 0 to 15 do
Set X[j] to M[i*16+j].
end  /* J循环的结束 */
/* Save A as AA, B as BB, C as CC, and D as DD. */
AA = A
BB = B
CC = C
DD = D
/* 第1轮*/
/* 以 [abcd k s i]表示如下操作
a = b + ((a + F(b,c,d) + X[k] + T[i]) <<< s). */
/* Do the following 16 operations. */
[ABCD 0 7 1] [DABC 1 12 2] [CDAB 2 17 3] [BCDA 3 22 4]
[ABCD 4 7 5] [DABC 5 12 6] [CDAB 6 17 7] [BCDA 7 22 8]
[ABCD 8 7 9] [DABC 9 12 10] [CDAB 10 17 11] [BCDA 11 22 12]
[ABCD 12 7 13] [DABC 13 12 14] [CDAB 14 17 15] [BCDA 15 22 16]
/* 第2轮。*/
/* 以 [abcd k s i]表示如下操作
```

```
a = b + ((a + G(b,c,d) + X[k] + T[i]) <<< s). */
/* Do the following 16 operations. */
[ABCD 1 5 17] [DABC 6 9 18] [CDAB 11 14 19] [BCDA 0 20 20]
[ABCD 5 5 21] [DABC 10 9 22] [CDAB 15 14 23] [BCDA 4 20 24]
[ABCD 9 5 25] [DABC 14 9 26] [CDAB 3 14 27] [BCDA 8 20 28]
[ABCD 13 5 29] [DABC 2 9 30] [CDAB 7 14 31] [BCDA 12 20 32]
/* 第3轮*/
/* 以 [abcd k s i]表示如下操作
a = b + ((a + H(b,c,d) + X[k] + T[i]) <<< s). */
/* Do the following 16 operations. */
[ABCD 5 4 33] [DABC 8 11 34] [CDAB 11 16 35] [BCDA 14 23 36]
[ABCD 1 4 37] [DABC 4 11 38] [CDAB 7 16 39] [BCDA 10 23 40]
[ABCD 13 4 41] [DABC 0 11 42] [CDAB 3 16 43] [BCDA 6 23 44]
[ABCD 9 4 45] [DABC 12 11 46] [CDAB 15 16 47] [BCDA 2 23 48]
/* 第4轮*/
/* 以 [abcd k s i]表示如下操作
  a = b + ((a + I(b,c,d) + X[k] + T[i]) <<< s). */
/* Do the following 16 operations. */
[ABCD 0 6 49] [DABC 7 10 50] [CDAB 14 15 51] [BCDA 5 21 52]
[ABCD 12 6 53] [DABC 3 10 54] [CDAB 10 15 55] [BCDA 1 21 56]
[ABCD 8 6 57] [DABC 15 10 58] [CDAB 6 15 59] [BCDA 13 21 60]
[ABCD 4 6 61] [DABC 11 10 62] [CDAB 2 15 63] [BCDA 9 21 64]
/* 然后进行如下操作 */
A = A + AA
B = B + BB
C = C + CC
D = D + DD
end /* 结束对 I 的循环*/
```

3.5 第五步：输出结果

报文摘要的产生后的形式为：A，B，C，D。也就是低位字节 A 开始，高位字节 D 结束。

现在完成了对 MD5 的描述，在附录中给出了 C 形式的程序。

4. 摘要

MD5 算法实现很容易，它提供了任意长度的信息的“指纹”（或称为报文摘要）。据推测要实现两个不同的报文产生相同的摘要需要 2^64 次的操作，要恢复给定摘要的报文则需要 2^128 次操作。

为寻找缺陷，MD5 算法已经过非常细致的检查。最后的结论是还需要相关

的更好的算法和更进一步的安全分析。

5. MD4 和 MD5 的区别

以下是 MD5 和 MD4 的不同点：

1. 加上了第四轮循环；
2. 每一步增加了一个唯一的常数值；
3. 第二轮中的函数 g 从(XY v XZ v YZ)变成了(XZ v Y not(Z))，以减少 G 函数的均衡性。
4. 第一步加上了上一步的结果，这将引起更快的“雪崩效应”；
5. 改变了第二轮和第三轮中访问消息子分组的次序，使其更不相似；
6. 近似优化了每一轮中的循环左移位移量以实现更快的“雪崩效应”，各轮的位移量互不相同。

6. 参考文献

[1] Rivest, R., "The MD4 Message Digest Algorithm", RFC 1320, MIT and RSA Data Security, Inc., April 1992.

[2] Rivest, R., "The MD4 message digest algorithm", in A.J. Menezes and S.A. Vanstone, editors, Advances in Cryptology - CRYPTO '90 Proceedings, pages 303-311, Springer-Verlag, 1991.

[3] CCITT Recommendation X.509 (1988), "The Directory - Authentication Framework."

7. 附录 A—参考应用程序

本附录包括以下文件：摘自 RSAREF: A Cryptographic Toolkit for Privacy-Enhanced Mail:

global.h	-- 全局头文件
md5.h	-- MD5 头文件
md5c.c	-- MD5 源代码

要得到更多的 RSAREF 信息，请发 e-mai 到: <rsaref@rsa.com>.

附录中还包括：

mddriver.c -- MD2、MD4 和 MD5 的测试驱动程序。

驱动程序默认情况下编译 MD5，但如果在 C 的编译命令行将 MD 参数设成 2 或 4，则也可以编译 MD2 和 MD4

此应用程序是方便使用的，可用在不同的平台上，在特殊的平台上优化它也并不困难，这留给读者作为练习。例如，在“小字节序”平台上，此平台 32 位字的最低地址字节最无意义的字节，并且没有队列限制，在 MD5 变换中的解码的命令调用可以被相应的类型替代。

A1 global.h

```
/* GLOBAL.H - RSAREF 类型和常数*/
/* 当且仅当编译器支持函数原型的声明时，PROTOTYPES 必须被设置一次
如果还没有定义 C 编译器的标记，下面的代码使 PROTOTYPES 置为 0。*/
#ifndef PROTOTYPES
#define PROTOTYPES 0
#endif

/* POINTER 定义成一个普通的指针类型 */
typedef unsigned char *POINTER;

/* UINT2 定义成两字节的字 */
typedef unsigned short int UINT2;

/* UINT4 定一成四字节的字  */
typedef unsigned long int UINT4;

/* PROTO_LIST 的定义依赖于上面 PROTOTYPES 的定义，如果使用了 PROTOTYPES，
那么
PROTO_LIST 返回此列表，否则返回一个空列表。*/
#if PROTOTYPES
#define PROTO_LIST(list) list
#else
#define PROTO_LIST(list) ()
#endif
```

A.2 md5.h

```
/*MD5.H - MD5C.C 头文件*/
/*本软件允许被复制或运用，但必须在所有提及和参考的地方标注"RSA Data Security,
Inc. MD5 Message-Digest Algorithm"，也允许产生或运用派生软件，但必须在所
有提及和参考的地方标明 "derived from the RSA Data Security, Inc. MD5
Message-Digest Algorithm"
    RSA 数据安全公司（RSA Data Security, Inc.）从来没有出于任何特定目的陈
述过关于此软件的可买性和实用性，它提供了"as is",没有表达或暗示过任何理由。
    此声明必须在任何此文件和软件的任何拷贝中保留。*/

/* MD5 context. */
typedef struct
{
  UINT4 state[4];                                   /* state (ABCD) */
  UINT4 count[2];        /* 位数量, 模 2^64 (低位在前) */
  unsigned char buffer[64];                         /* 输入缓冲器 */
} MD5_CTX;

void MD5Init PROTO_LIST ((MD5_CTX *));
void MD5Update PROTO_LIST
  ((MD5_CTX *, unsigned char *, unsigned int));
void MD5Final PROTO_LIST ((unsigned char [16], MD5_CTX *));
```

A.3 md5c.c

```
/* MD5C.C - RSA 数据安全公司，MD5 报文摘要算法*/
/*本软件允许被复制或运用，但必须在所有提及和参考的地方标注"RSA Data Security,
Inc. MD5 Message-Digest Algorithm"也允许产生或运用派生软件，但必须在所有
提及和参考的地方标明 "derived from the RSA Data RSA 数据安全公司（RSA Data
Security, Inc.）从来没有出于任何特定目的陈述过关于此软件的可买性和实用性，它
提供了"as is",没有表达或暗示过任何理由。
    此声明必须在任何此文件和软件的任何拷贝中保留。*/

#include "global.h"
#include "md5.h"

/* Constants for MD5Transform routine.
 */

#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

static void MD5Transform PROTO_LIST ((UINT4 [4], unsigned char [64]));
static void Encode PROTO_LIST
  ((unsigned char *, UINT4 *, unsigned int));
static void Decode PROTO_LIST
  ((UINT4 *, unsigned char *, unsigned int));
static void MD5_memcpy PROTO_LIST ((POINTER, POINTER, unsigned int));
static void MD5_memset PROTO_LIST ((POINTER, int, unsigned int));

static unsigned char PADDING[64] = {
  0x80, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
};

/* F, G, H 和 I 是基本 MD5 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 循环左移 n 位 */
#define ROTATE_LEFT(x, n) (((x) << (n)) | ((x) >> (32-(n))))

/* 循环从加法中分离出是为了防止重复计算*/
#define FF(a, b, c, d, x, s, ac) { \
 (a) += F ((b), (c), (d)) + (x) + (UINT4)(ac); \
 (a) = ROTATE_LEFT ((a), (s)); \

 (a) += (b); \
```

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

/* MD5 初始化. 开始一个 MD5 操作写一个新的 context. */
void MD5Init (context)
MD5_CTX *context;                                        /* context */
{
  context->count[0] = context->count[1] = 0;
  context->state[0] = 0x67452301;
  context->state[1] = 0xefcdab89;
  context->state[2] = 0x98badcfe;
  context->state[3] = 0x10325476;
}

/*MD5 分组更新操作. 继续一个 MD5 操作,处理另一个消息分组并更新 context. */
void MD5Update (context, input, inputLen)
MD5_CTX *context;                                        /* context */
unsigned char *input;                                  /* 输入分组*/
unsigned int inputLen;                          /* 输入的分组的长度 */
{
  unsigned int i, index, partLen;

     /* 计算字节数模 64 的值 */
  index = (unsigned int)((context->count[0] >> 3) & 0x3F);

  /* Update number of bits */
  if ((context->count[0] += ((UINT4)inputLen << 3))

   < ((UINT4)inputLen << 3))
```

```
context->count[1]++;
 context->count[1] += ((UINT4)inputLen >> 29);

 partLen = 64 - index;

    /* 按能达到的最大次数转换*/
if (inputLen >= partLen) {
MD5_memcpy
  ((POINTER)&context->buffer[index], (POINTER)input, partLen);
MD5Transform (context->state, context->buffer);

for (i = partLen; i + 63 < inputLen; i += 64)
  MD5Transform (context->state, &input[i]);

index = 0;
 }
 else
i = 0;

    /* 缓冲器保留输入值 */
 MD5_memcpy
((POINTER)&context->buffer[index], (POINTER)&input[i],
 inputLen-i);
}

/* MD5 最终结果. 以一个 MD5 报文摘要操作结束, 写下报文摘要值 */
void MD5Final (digest, context)
unsigned char digest[16];                        /*报文摘要 */
MD5_CTX *context;                                   /* context */
{
  unsigned char bits[8];
  unsigned int index, padLen;

    /* 保存位数值 */
  Encode (bits, context->count, 8);
  index = (unsigned int)((context->count[0] >> 3) & 0x3f);
  padLen = (index < 56) ? (56 - index) : (120 - index);
  MD5Update (context, PADDING, padLen);

  /* 附加长度 (在补位之前) */
  MD5Update (context, bits, 8);

  /* 将 state 存入 digest 中*/
```

```
  Encode (digest, context->state, 16);
  MD5_memset ((POINTER)context, 0, sizeof (*context));
}

/* MD5 基本转换. 转换状态基于分组*/
static void MD5Transform (state, block)
UINT4 state[4];
unsigned char block[64];
{
  UINT4 a = state[0], b = state[1], c = state[2], d = state[3], x[16];

  Decode (x, block, 64);

  /* Round 1 */
  FF (a, b, c, d, x[ 0], S11, 0xd76aa478); /* 1 */
  FF (d, a, b, c, x[ 1], S12, 0xe8c7b756); /* 2 */
  FF (c, d, a, b, x[ 2], S13, 0x242070db); /* 3 */
  FF (b, c, d, a, x[ 3], S14, 0xc1bdceee); /* 4 */
  FF (a, b, c, d, x[ 4], S11, 0xf57c0faf); /* 5 */
  FF (d, a, b, c, x[ 5], S12, 0x4787c62a); /* 6 */
  FF (c, d, a, b, x[ 6], S13, 0xa8304613); /* 7 */
  FF (b, c, d, a, x[ 7], S14, 0xfd469501); /* 8 */
  FF (a, b, c, d, x[ 8], S11, 0x698098d8); /* 9 */
  FF (d, a, b, c, x[ 9], S12, 0x8b44f7af); /* 10 */
  FF (c, d, a, b, x[10], S13, 0xffff5bb1); /* 11 */
  FF (b, c, d, a, x[11], S14, 0x895cd7be); /* 12 */
  FF (a, b, c, d, x[12], S11, 0x6b901122); /* 13 */
  FF (d, a, b, c, x[13], S12, 0xfd987193); /* 14 */
  FF (c, d, a, b, x[14], S13, 0xa679438e); /* 15 */
  FF (b, c, d, a, x[15], S14, 0x49b40821); /* 16 */

 /* Round 2 */
  GG (a, b, c, d, x[ 1], S21, 0xf61e2562); /* 17 */
  GG (d, a, b, c, x[ 6], S22, 0xc040b340); /* 18 */
  GG (c, d, a, b, x[11], S23, 0x265e5a51); /* 19 */
  GG (b, c, d, a, x[ 0], S24, 0xe9b6c7aa); /* 20 */
  GG (a, b, c, d, x[ 5], S21, 0xd62f105d); /* 21 */
  GG (d, a, b, c, x[10], S22,  0x2441453); /* 22 */
  GG (c, d, a, b, x[15], S23, 0xd8a1e681); /* 23 */
  GG (b, c, d, a, x[ 4], S24, 0xe7d3fbc8); /* 24 */
  GG (a, b, c, d, x[ 9], S21, 0x21e1cde6); /* 25 */
  GG (d, a, b, c, x[14], S22, 0xc33707d6); /* 26 */
  GG (c, d, a, b, x[ 3], S23, 0xf4d50d87); /* 27 */
```

```
GG (b, c, d, a, x[ 8], S24, 0x455a14ed); /* 28 */
GG (a, b, c, d, x[13], S21, 0xa9e3e905); /* 29 */
GG (d, a, b, c, x[ 2], S22, 0xfcefa3f8); /* 30 */
GG (c, d, a, b, x[ 7], S23, 0x676f02d9); /* 31 */
GG (b, c, d, a, x[12], S24, 0x8d2a4c8a); /* 32 */

/* Round 3 */
HH (a, b, c, d, x[ 5], S31, 0xfffa3942); /* 33 */
HH (d, a, b, c, x[ 8], S32, 0x8771f681); /* 34 */
HH (c, d, a, b, x[11], S33, 0x6d9d6122); /* 35 */
HH (b, c, d, a, x[14], S34, 0xfde5380c); /* 36 */
HH (a, b, c, d, x[ 1], S31, 0xa4beea44); /* 37 */
HH (d, a, b, c, x[ 4], S32, 0x4bdecfa9); /* 38 */
HH (c, d, a, b, x[ 7], S33, 0xf6bb4b60); /* 39 */
HH (b, c, d, a, x[10], S34, 0xbebfbc70); /* 40 */
HH (a, b, c, d, x[13], S31, 0x289b7ec6); /* 41 */
HH (d, a, b, c, x[ 0], S32, 0xeaa127fa); /* 42 */
HH (c, d, a, b, x[ 3], S33, 0xd4ef3085); /* 43 */
HH (b, c, d, a, x[ 6], S34,  0x4881d05); /* 44 */
HH (a, b, c, d, x[ 9], S31, 0xd9d4d039); /* 45 */
HH (d, a, b, c, x[12], S32, 0xe6db99e5); /* 46 */
HH (c, d, a, b, x[15], S33, 0x1fa27cf8); /* 47 */
HH (b, c, d, a, x[ 2], S34, 0xc4ac5665); /* 48 */

/* Round 4 */
II (a, b, c, d, x[ 0], S41, 0xf4292244); /* 49 */
II (d, a, b, c, x[ 7], S42, 0x432aff97); /* 50 */
II (c, d, a, b, x[14], S43, 0xab9423a7); /* 51 */
II (b, c, d, a, x[ 5], S44, 0xfc93a039); /* 52 */
II (a, b, c, d, x[12], S41, 0x655b59c3); /* 53 */
II (d, a, b, c, x[ 3], S42, 0x8f0ccc92); /* 54 */
II (c, d, a, b, x[10], S43, 0xffeff47d); /* 55 */
II (b, c, d, a, x[ 1], S44, 0x85845dd1); /* 56 */
II (a, b, c, d, x[ 8], S41, 0x6fa87e4f); /* 57 */
II (d, a, b, c, x[15], S42, 0xfe2ce6e0); /* 58 */
II (c, d, a, b, x[ 6], S43, 0xa3014314); /* 59 */
II (b, c, d, a, x[13], S44, 0x4e0811a1); /* 60 */
II (a, b, c, d, x[ 4], S41, 0xf7537e82); /* 61 */
II (d, a, b, c, x[11], S42, 0xbd3af235); /* 62 */
II (c, d, a, b, x[ 2], S43, 0x2ad7d2bb); /* 63 */
II (b, c, d, a, x[ 9], S44, 0xeb86d391); /* 64 */
```

```
  state[0] += a;
  state[1] += b;
  state[2] += c;
  state[3] += d;


  MD5_memset ((POINTER)x, 0, sizeof (x));
}

/* 将输入（UINT4）编码输出(unsigned char). 假设 len 是 4 的倍数 */
static void Encode (output, input, len)
unsigned char *output;
UINT4 *input;
unsigned int len;
{
  unsigned int i, j;

  for (i = 0, j = 0; j < len; i++, j += 4) {
 output[j] = (unsigned char)(input[i] & 0xff);
 output[j+1] = (unsigned char)((input[i] >> 8) & 0xff);
 output[j+2] = (unsigned char)((input[i] >> 16) & 0xff);
 output[j+3] = (unsigned char)((input[i] >> 24) & 0xff);
  }
}

/* 将输入(unsigned char)解码输出 (UINT4). 假设 len 是 4 的倍数 */
static void Decode (output, input, len)
UINT4 *output;
unsigned char *input;
unsigned int len;
{
  unsigned int i, j;

  for (i = 0, j = 0; j < len; i++, j += 4)
 output[i] = ((UINT4)input[j]) | (((UINT4)input[j+1]) << 8) |
   (((UINT4)input[j+2]) << 16) | (((UINT4)input[j+3]) << 24);
}
static void MD5_memcpy (output, input, len)
POINTER output;
POINTER input;
unsigned int len;
{
  unsigned int i;
```

```
  for (i = 0; i < len; i++)

 output[i] = input[i];
}
static void MD5_memset (output, value, len)
POINTER output;
int value;
unsigned int len;
{
  unsigned int i;

  for (i = 0; i < len; i++)
 ((char *)output)[i] = (char)value;
}
```

A.4 mddriver.c

```
/* MDDRIVER.C - MD2, MD4 and MD5 测试程序 */
/* RSA 数据安全公司（RSA Data Security, Inc.）从来没有出于任何特定目的陈
述过关于此软件的可买性和实用性，它提供了"as is",没有表达或暗示过任何理由。
此声明必须在任何此文件和软件的任何拷贝中保留。*/
/* 如果没有定义 C 编译标志的值，下面一段设定 MD 缺省值为 MD5 */
#ifndef MD
#define MD MD5
#endif

#include <stdio.h>
#include <time.h>
#include <string.h>
#include "global.h"
#if MD == 2
#include "md2.h"
#endif
#if MD == 4
#include "md4.h"
#endif
#if MD == 5
#include "md5.h"
#endif
```

```
/* 测试分组长度和数量 */
#define TEST_BLOCK_LEN 1000
#define TEST_BLOCK_COUNT 1000

static void MDString PROTO_LIST ((char *));
static void MDTimeTrial PROTO_LIST ((void));
static void MDTestSuite PROTO_LIST ((void));
static void MDFile PROTO_LIST ((char *));
static void MDFilter PROTO_LIST ((void));
static void MDPrint PROTO_LIST ((unsigned char [16]));

#if MD == 2
#define MD_CTX MD2_CTX
#define MDInit MD2Init
#define MDUpdate MD2Update
#define MDFinal MD2Final
#endif
#if MD == 4
#define MD_CTX MD4_CTX
#define MDInit MD4Init
#define MDUpdate MD4Update
#define MDFinal MD4Final
#endif
#if MD == 5
#define MD_CTX MD5_CTX
#define MDInit MD5Init
#define MDUpdate MD5Update
#define MDFinal MD5Final
#endif

/* 主程序.
 变量:
  -sstring          - 摘要字符串
  -t                - 运行时间测试
  -x                - 运行测试脚本
  filename          - 摘要文件
  (none)            - 摘要标准输入
 */
int main (argc, argv)
int argc;

char *argv[];
{
```

```
  int i;

  if (argc > 1)
 for (i = 1; i < argc; i++)
   if (argv[i][0] == '-' && argv[i][1] == 's')
     MDString (argv[i] + 2);
   else if (strcmp (argv[i], "-t") == 0)
     MDTimeTrial ();
   else if (strcmp (argv[i], "-x") == 0)
     MDTestSuite ();
   else
     MDFile (argv[i]);
  else
 MDFilter ();

  return (0);
}

/* 计算字符串的摘要并打印其值 */
static void MDString (string)
char *string;
{
  MD_CTX context;
  unsigned char digest[16];
  unsigned int len = strlen (string);

  MDInit (&context);
  MDUpdate (&context, string, len);
  MDFinal (digest, &context);

  printf ("MD%d (\"%s\") = ", MD, string);
  MDPrint (digest);
  printf ("\n");
}

/* 测试计算 TEST_BLOCK_COUNT TEST_BLOCK_LEN-byte
   分组摘要的时间 */
static void MDTimeTrial ()
{
  MD_CTX context;
  time_t endTime, startTime;
  unsigned char block[TEST_BLOCK_LEN], digest[16];
  unsigned int i;
```

```
 printf
("MD%d time trial. Digesting %d %d-byte blocks ...", MD,
 TEST_BLOCK_LEN, TEST_BLOCK_COUNT);

 /* 初始化分组*/
 for (i = 0; i < TEST_BLOCK_LEN; i++)
block[i] = (unsigned char)(i & 0xff);

 /* 开始时钟 */
 time (&startTime);

   /* 摘要分组 */
 MDInit (&context);
 for (i = 0; i < TEST_BLOCK_COUNT; i++)
MDUpdate (&context, block, TEST_BLOCK_LEN);
 MDFinal (digest, &context);

   /* 停止时钟 */
 time (&endTime);

 printf (" done\n");
 printf ("Digest = ");
 MDPrint (digest);
 printf ("\nTime = %ld seconds\n", (long)(endTime-startTime));
 printf
("Speed = %ld bytes/second\n",
 (long)TEST_BLOCK_LEN *
(long)TEST_BLOCK_COUNT/(endTime-startTime));
}

/* 计算一个参考组件串的摘要并打印结果*/
static void MDTestSuite ()
{
 printf ("MD%d test suite:\n", MD);

 MDString ("");
 MDString ("a");
 MDString ("abc");
 MDString ("message digest");
 MDString ("abcdefghijklmnopqrstuvwxyz");
 MDString
```

```
("ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789"
);
  MDString
 ("12345678901234567890123456789012345678901\
234567890123456789012345678901234567890");
}

/*计算一个文件的摘要并打印结果 */
static void MDFile (filename)
char *filename;
{
  FILE *file;
  MD_CTX context;
  int len;
  unsigned char buffer[1024], digest[16];

  if ((file = fopen (filename, "rb")) == NULL)
 printf ("%s can't be opened\n", filename);

  else {
 MDInit (&context);
 while (len = fread (buffer, 1, 1024, file))
   MDUpdate (&context, buffer, len);
 MDFinal (digest, &context);

 fclose (file);

 printf ("MD%d (%s) = ", MD, filename);
 MDPrint (digest);
 printf ("\n");
  }
}

/* 计算标准输入的摘要并打印结果*/
static void MDFilter ()
{
  MD_CTX context;
  int len;
  unsigned char buffer[16], digest[16];

  MDInit (&context);
  while (len = fread (buffer, 1, 16, stdin))
```

```
 MDUpdate (&context, buffer, len);
  MDFinal (digest, &context);

  MDPrint (digest);
  printf ("\n");
}

/* 打印一个16进制的摘要*/
static void MDPrint (digest)
unsigned char digest[16];
{

  unsigned int i;

  for (i = 0; i < 16; i++)
 printf ("%02x", digest[i]);
}
```

A.5 测试组件

MD5 测试组件(驱动程序选项"-x")应打印以下值：

MD5 test suite:

MD5 ("") = d41d8cd98f00b204e9800998ecf8427e

MD5 ("a") = 0cc175b9c0f1b6a831c399e269772661

MD5 ("abc") = 900150983cd24fb0d6963f7d28e17f72

MD5 ("message digest") = f96b697d7cb7938d525a2f31aaf161d0

MD5 ("abcdefghijklmnopqrstuvwxyz") = c3fcd3d76192e4007dfb496cca67e13b

MD5 ("ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789")
=d174ab98d277d9f5a5611c2c9f419d9f

MD5 ("12345678901234567890123456789012345678901234567890123456789012345678
12345678901234567890") = 57edf4a22be3c955ac49da2e2107b67a

8. 安全事项

本文中讨论的安全标准被认为已足够实现很高要求的基于公用密钥系统和MD5 算法的数字签名系统中。

9. 作者地址

Ronald L. Rivest

Massachusetts Institute of Technology

Laboratory for Computer Science

NE43-324

545 Technology Square

Cambridge, MA 02139-1986

Phone: (617) 253-5880

EMail: rivest@theory.lcs.mit.edu