MD5 实验报告

16327109 谢昆成

算法原理

MD5即Message-Digest Algorithm 5 (信息-摘要算法5)，是一种Hash算法，为Merkel 模型的一种。

模型结构

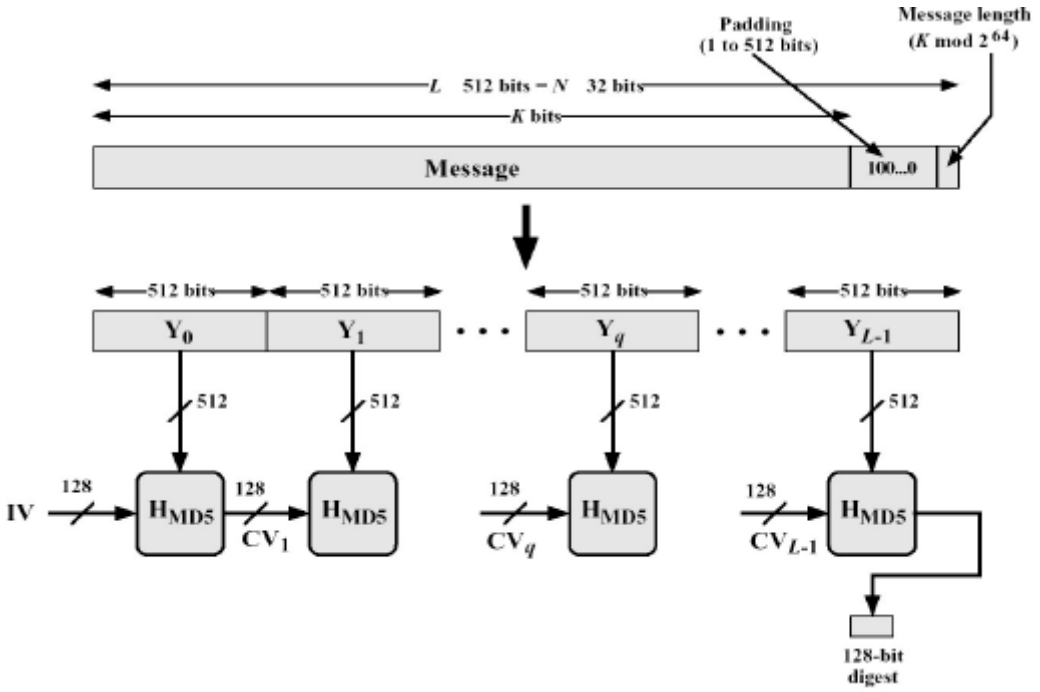
1. 把原始消息M 分成一些固定长度的块Yi ；
2. 最后一块padding 并使其包含消息M 的长度；
3. 设定初始值CV_0 ；
4. 采用压缩函数f，$CV_i = f(CV_{i-1}, Y_{i-1})$ ；
5. 最后一个CV_i 为hash 值。

MD5 使用little-endian (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。

总体结构

MD5 算法基本流程



填充padding

1. 在长度为 *K* bits 的原始消息数据尾部填充长度为 *P* bits 的标识100...0，1<=P<=512 (即至少要填充1个bit)，使得填充后的消息位数为：*K* + P= 448 (mod 512).

   注意到当 *K* = 448 (mod 512) 时，需要 *P*= 512.

2. 再向上述填充好的消息尾部附加 *K* 值的低64位(即 *K* mod 2^64)， 最后得到一个长度位数为 *K* + *P* + 64 = 0 (mod 512) 的消息。

分块

1. 把填充后的消息结果分割为L 个512-bit 分组：Y0, Y1, ..., YL-1。
2. 分组结果也可表示成N 个32-bit 字M0, M1, ..., MN-1，N = L*16。

初始化

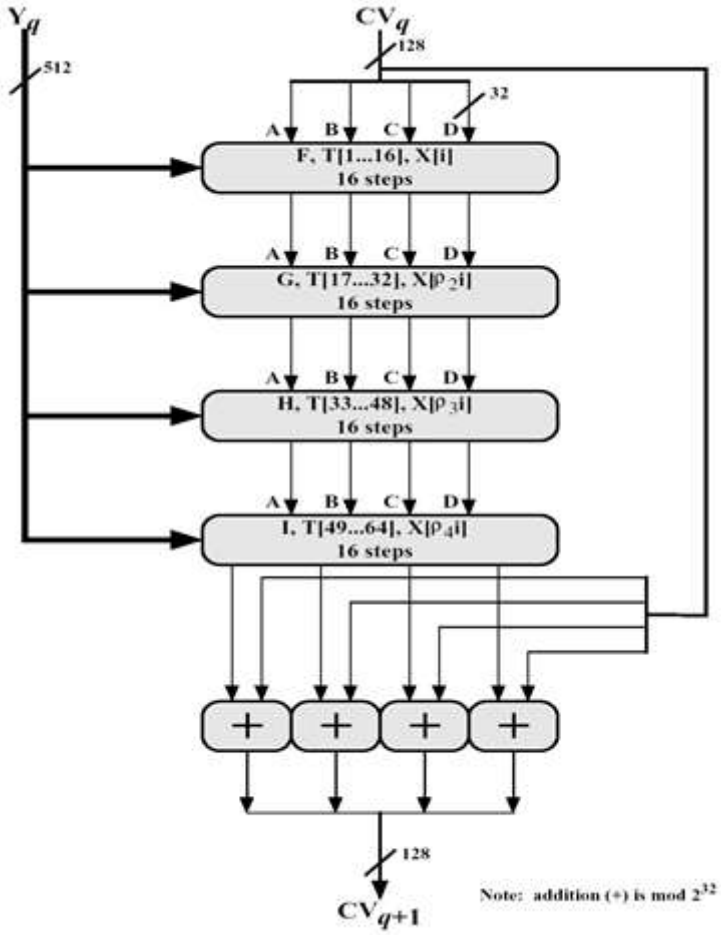
1. 初始化一个128-bit 的MD 缓冲区，记为CV_q，表示成4个32-bit寄存器(A, B, C, D)；$CV_0 = IV$。迭代在MD 缓冲区进行，最后一步的128-bit 输出即为算法结果。

总控流程

1. 以512-bit 消息分组为单位，每一分组Y_q (q = 0, 1, ..., L-1) 经过4个循环的压缩算法，表示为： CV0 = IV CVi = HMD5(CVi-1 , Yi)
2. 输出结果：MD = CVL .

压缩函数HMD5

1. HMD5 从CV 输入128位，从消息分组输入512位，完成4轮循环后，输出128位，用于下 一轮输入的CV 值。
   - 每轮循环分别固定不同的生成函数F, G, H, I，结合指定的T 表元素T[] 和消息分组的不同部分X[] 做16次迭代运算，生成下一轮循环的输入。
   - 4轮循环共有64次迭代运算。



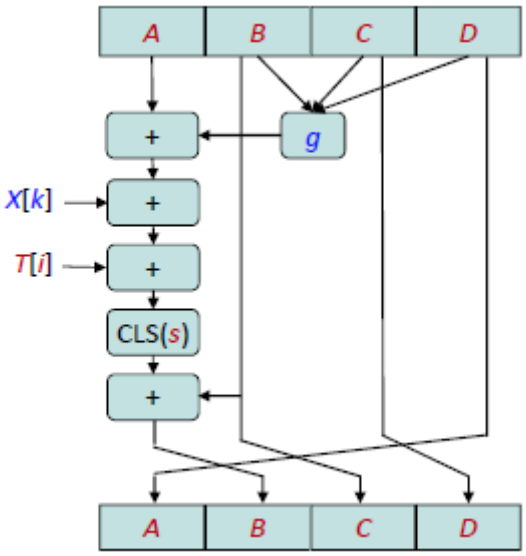
2. 4轮循环中使用的生成函数(轮函数) g 是一个32位非线性逻辑函数，在相应各轮 的定义如下：

轮次	Function *g*	*g*(*b*, *c*, *d*)
1	*F*(*b*, *c*, *d*)	$(b \wedge c) \vee (\neg b \wedge d)$
2	*G*(*b*, *c*, *d*)	$(b \wedge d) \vee (c \wedge \neg d)$
3	*H*(*b*, *c*, *d*)	$b \oplus c \oplus d$
4	*I*(*b*, *c*, *d*)	$c \oplus (b \vee \neg d)$

3. 每轮循环中的一次迭代运算逻辑 (1) 对A 迭代：a <- b + ((a + g(b, c, d) + X[k] + T[i]) <<<s) (2) 缓冲区(A, B, C, D) 作循环轮换：(B, C, D, A) <- (A, B, C, D)

   X[k] : 当前处理消息分组q 的第k 个(k = 0..15) 32位字，即M_{q*16+k}。

   T[i] : T 表的第i 个元素，32位字；T 表总共有64个元素，也称为加法常数。



模块分解

根据模块的总体结构，我们可以将MD5算法分解为以下部分：

1. 数据定义，包括X、T等数据
2. MD5算法框架，文件读写等
3. 最后的特殊处理，填充（可读取到最后时填充）
4. 循环压缩算法结构CYCLE
5. 压缩算法F、G、H、I

数据结构

采用面向对象的设计方法。将总体算法封装成MD5类。

将数据和算法分离，用数组定义算法中的数据。

用unsigned char 代表一个字节。

通过模块分解，将MD5分解为`md5`,`process`,`finalProcess`,`padding`,`llToChar`等函数。

追求最少冗余代码，尽可能简单实现。

运行结果

将运行结果与网站MD5算法结果对比。

短文件测试

test1.txt仅有

```
Hello, world!
```

结果

```
00000000: 6cd3 556d eb0d a54b
00000008: ca06 0b4c 3947 9839
```

网址MD5结果

字符串	Hello, world!
16位 小写	eb0da54bca060b4c
16位 大写	EB0DA54BCA060B4C
32位 小写	6cd3556deb0da54bca060b4c39479839
32位 大写	6CD3556DEB0DA54BCA060B4C39479839

结果正确

长文件测试

test2.txt中有275个字符，上千位

结果

```
00000000: 1b70 3e20 489a fa69
00000008: 17d5 8072 c00a 9a79
```

网址MD5结果

字符串	Over the past five months or so, a series of violent crimes in Hong Kong have seriously trampled the rule of law and social order, undermined Hong Kong's prosperity and stability, and challenged the bottom line of the "one country, two systems" principle, the statement read.
16位 小写	489afa6917d58072
16位 大写	489AFA6917D58072
32位 小写	1b703e20489afa6917d58072c00a9a79
32位 大写	1B703E20489AFA6917D58072C00A9A79

结果正确

编译运行方法

进入src/目录

编译

```
g++ main.cpp -o main
```

运行

```
 main inputFilename outputFilename
```

如

```
main ../test/test1.txt ../result/result1.txt
```

```
main ../test/test2.txt ../result/result2.txt
```

C++源代码

md5.h

main.cpp

实验体会

编程时遇到几个bug，记录如下

- 填充的最后64位P即信息长度是以位为单位计算，自己曾一度以字节来计数，导致错误
- 小端存放，填充的P也如此
- 算法流程不熟悉，每4轮循环后要将A、B、C、D分别与CV_q相加，得到CV_{q+1}