**信息安全与技术作业二**

**MD5算法**

**17343105 田皓**

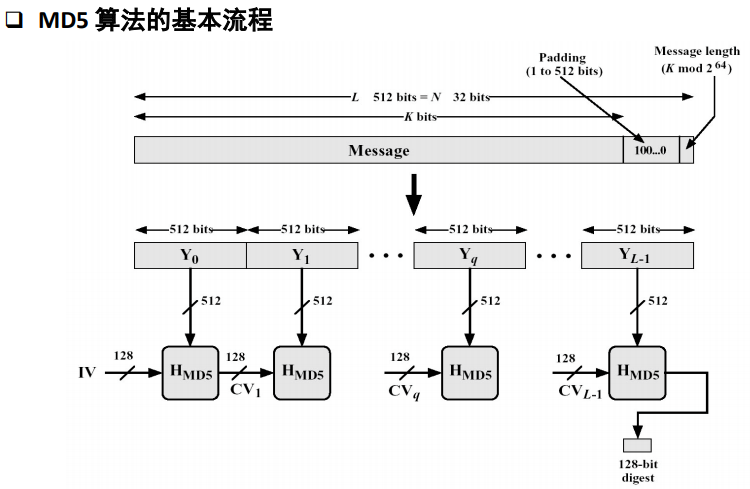
## 原理概述

MD5 即 Message-Digest Algorithm 5 (信息-摘要算法 5)

* MD4 (1990)、MD5(1992, RFC 1321) 由 Ron Rivest 发明，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 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。
* MD5 不是足够安全的。
* Hans Dobbertin 在1996年找到了两个不同的 512-bit 块,它们 在 MD5 计算下产生相同的 hash 值。
* 至今还没有真正找到两个不同的消息，它们的 MD5 的 hash 值相等。

## 

## 总体结构



## 模块分解

1. 填充padding

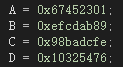
* 在长度为 K bits 的原始消息数据尾部填充长度为 P bits 的标识 100…0，1P 512 (即至少要填充1个bit)，使得填充后的消息位 数为：K + P  448 (mod 512). ◌ 注意到当 K  448 (mod 512) 时，需要 P= 512.
* 再向上述填充好的消息尾部附加 K 值的低64位 (即 K mod 264)， 最后得到一个长度位数为 K + P + 64  0 (mod 512) 的消息。

1. 分块

把填充后的结果分为L个512位的分组。

1. 初始化缓存区

初始化一个128bit的缓冲区，表示为四个32bit的寄存器ABCD。

A ：01 23 45 67

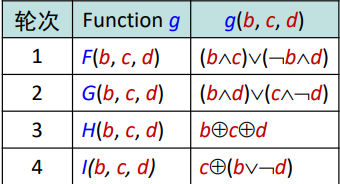
B： 89 ab cd ef

C： fe dc ba 98

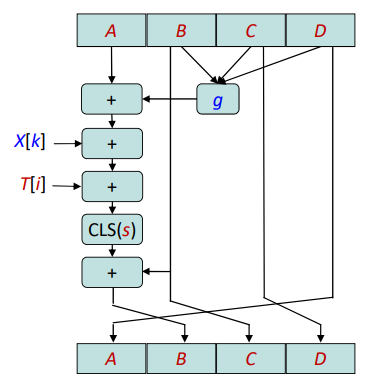
D：76 54 32 10

1. 循环压缩

首先定义四个轮函数g：



1. unsigned **int** F(unsigned **int** b, unsigned **int** c, unsigned **int** d) {
2. **return** (b & c) | ((~b) & d);
3. }
5. unsigned **int** G(unsigned **int** b, unsigned **int** c, unsigned **int** d) {
6. **return** (b & d) | (c & (~d));
7. }
9. unsigned **int** H(unsigned **int** b, unsigned **int** c, unsigned **int** d) {
10. **return** b ^ c ^ d;
11. }
13. unsigned **int** I(unsigned **int** b, unsigned **int** c, unsigned **int** d) {
14. **return** c ^ (b | (~d));
15. }

* 每一次迭代运算逻辑 对A迭代：A <= B + ((A + g(B,C,D) + X[k] + T[i])) <<< S[i]
* 缓冲区循环轮换：(B,C,D,A）<=（A,B,C,D）
* 其中：
  + A,B,C,D代表MD5缓冲区当前的数值。
  + g为轮函数，1-16轮迭代使用F函数，17-32轮迭代使用G函数，33-48轮迭 代使用H函数，49-64轮迭代使用I函数。
  + S[i]，32位输入循环左移 (CLS) S[i] 位，S表为规定值。
  + X[k]，当前处理消息分组的第k个32位字，X[k]由第n轮迭代对应的顺序表决定。
  + T[i]，T表的第i项的值。
* 每轮循环中的一次迭代运算
* 每轮循环第j次迭代使用的X[K]：
* i = j - 1
* f轮：k = i;
* g轮：k = (1 + 5 \* i) % 16;
* h轮：k = (5 + 3 \* i) % 16;
* i轮：k = (7 \* i) % 16;
* T表和S表

1. **const** unsigned **int** T[] = {
2. 0xd76aa478,0xe8c7b756,0x242070db,0xc1bdceee,
3. 0xf57c0faf,0x4787c62a,0xa8304613,0xfd469501,
4. 0x698098d8,0x8b44f7af,0xffff5bb1,0x895cd7be,
5. 0x6b901122,0xfd987193,0xa679438e,0x49b40821,
6. 0xf61e2562,0xc040b340,0x265e5a51,0xe9b6c7aa,
7. 0xd62f105d,0x02441453,0xd8a1e681,0xe7d3fbc8,
8. 0x21e1cde6,0xc33707d6,0xf4d50d87,0x455a14ed,
9. 0xa9e3e905,0xfcefa3f8,0x676f02d9,0x8d2a4c8a,
10. 0xfffa3942,0x8771f681,0x6d9d6122,0xfde5380c,
11. 0xa4beea44,0x4bdecfa9,0xf6bb4b60,0xbebfbc70,
12. 0x289b7ec6,0xeaa127fa,0xd4ef3085,0x04881d05,
13. 0xd9d4d039,0xe6db99e5,0x1fa27cf8,0xc4ac5665,
14. 0xf4292244,0x432aff97,0xab9423a7,0xfc93a039,
15. 0x655b59c3,0x8f0ccc92,0xffeff47d,0x85845dd1,
16. 0x6fa87e4f,0xfe2ce6e0,0xa3014314,0x4e0811a1,
17. 0xf7537e82,0xbd3af235,0x2ad7d2bb,0xeb86d391
18. };
20. **const** unsigned **int** s[] = {
21. 7,12,17,22,7,12,17,22,7,12,17,22,7,12,17,22,
22. 5,9,14,20,5,9,14,20,5,9,14,20,5,9,14,20,
23. 4,11,16,23,4,11,16,23,4,11,16,23,4,11,16,23,
24. 6,10,15,21,6,10,15,21,6,10,15,21,6,10,15,21
25. };

**结果验证**

借python的hashlib库的md5方法来验证结果。

