MD5 算法详解

陈志扬 15331046

MD5 即 Message-Digest Algorithm 5 (信息-摘要算法 5)，是信息摘要的一种实现，它使用 little-endian，可以从任意长度的明文字符串，以 512 位长进行分组，生成四个 32 位数据，最后联合起来生成 128 位的哈希值，即 128 位的二进制数，也就是 32 位的十六进制数。

MD5 算法的基本过程为：求余、取余、调整长度、与链接变量进行循环运算、得出结果。

MD5 算法底层原理：

简单概括起来，MD5 算法的过程分为四步：**处理原文，设置初始值，循环加工，拼接结果。**

第一步：处理原文

首先，我们计算出原文长度（bit）对 512 求余的结果，如果不等于 448，就需要填充原文使得原文对 512 求余的结果等于 448.

填充的方法为第一位填充 1，其余位填充 0。填充完后，信息的长度就是 512*N+448.

之后，用剩余的位置（512-448=64 位）记录原文的真正长度，把长度的二进制补在最后。这样处理后的信息长度就是 512*（N+1）。

```
/**
 * @Initialization the md5 object, processing another message block,
 * and updating the context.
 *
 * @param {input} the input message.
 *
 * @param {length} the number btye of message.
 *
 */
void MD5::encryptUnsignedChar(const unsigned char* input, size_t length) {
    unsigned int index, partLen;
    size_t i;

    /* Compute number of bytes mod 64 */
    index = static_cast<unsigned int>((count[0] >> 3) & 0x3f);
    /* update number of bits */
    if ((count[0] += (static_cast<unsigned int>(length) << 3))
        < (static_cast<unsigned int>(length) << 3)) {
        count[1]++;
    }
    count[1] +=  (static_cast<unsigned int>(length) >> 29);

    partLen = 64 - index;

    /* transform as many times as possible. */
    if (length >= partLen) {
        memcpy(&buffer[index], input, partLen);
        processOfMD5(buffer);
        for (i = partLen; i + 63 < length; i += 64) {
            processOfMD5(&input[i]);
        }
        index = 0;
    }
    else {
        i = 0;
    }
    /* Buffer remaining input */
    memcpy(&buffer[index], &input[i], length - i);
}
```

第二步：设置初始值

MD5 的哈希结果长度为 128 位，按每 32 位分成一组共 4 组。这 4 组结果是由 4 个初始值 A、B、C、D 经过不断演变得到。MD5 的官方实现中，A、B、C、D 的初始值如下（16 进制）：

A=0x67452301

B=0xefcdab89

C=0x98badcfe

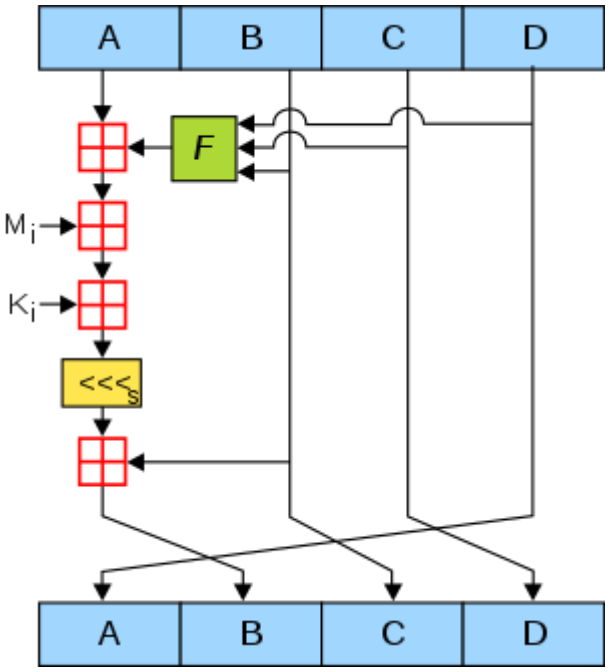
D=0x10325476

```
/**
 * Args：空
 * Func：初始化链接变量
 * Return：空
 */
void MD5::MD5_init() {
    count[0] = 0;
    count[1] = 0;

    state[0] = 0x67452301;
    state[1] = 0xefcdab89;
    state[2] = 0x98badcfe;
    state[3] = 0x10325476;
}
```

第三步：循环加工

这一步是最复杂的一步，其过程如下图，此图代表了单次 A、B、C、D 值演变的流程。



图中，A、B、C、D 就是哈希值的四个分组。每一次循环都会让旧的 A、B、C、D 产生新的 A、B、C、D。

一共进行多少次循环呢？由处理后的原文长度决定。

假设处理后的原文长度为 L

主循环次数=L/512

每个主循环中包含 512/32*4=64 次子循环

上面这张图所表达的就是单次子循环的流程。

下面对图中其他元素一一解释：

1.绿色 F：代表非线性函数。官方 MD5 所用到的函数有 4 种：

F(X,Y,Z)=(X&Y) | ((~X)&Z)

G(X,Y,Z)=(X&Z) | (Y&(~Z))

H(X,Y,Z)=X^Y^Z

I(X,Y,Z)=Y ^ (X|(~Z))

在主循环下面 64 次子循环中，F、G、H、I 交替使用，第一个 16 次使用 F，第二个 16 次使用 G，第三个 16 次使用 H，第四个 16 次使用 I。

2.红色“田”字：代表相加的意思

3.M_i：是第一步处理后的原文。在第一步中，处理后原文的长度是 512 的整数倍。把原文的每 512 位再分成 16 等份，命名为 M_0~M_{15}，每一等份长度为 32.在 64 次循环中，每 16 次循环，都会交替用到 M_0~M_{15} 之一。

4.K_i：一个常量，在 64 次子循环中，每一次用到的常量都是不同的。

5.黄色的<<<S：左移 S 位，S 的值也是常量。

“流水线”的最后，让计算的结果和 B 相加，取代原先的 B。新 A、B、C、D 的产生可以归纳为：

新 A=原 d

新 B=b+((a+F(b,c,d)+M_i+K_i)<<<s)

新 C=原 b

新 D=原 c

```
/**
 * Args：groups[]表示一个512位（64字节）分组
 * Func：四轮主要操作
 * Return：空
 */
void MD5::processOfMD5(const unsigned char groups[64]) {
    unsigned int a = state[0], b = state[1], c = state[2], d = state[3];
    unsigned int M[16];

    UnsignedCharToUnsignedInt(groups, M, 64);
    // 第一轮循环
    FF(a, b, c, d, M[0], 7, 0xd76aa478);
    FF(d, a, b, c, M[1], 12, 0xe8c7b756);
    FF(c, d, a, b, M[2], 17, 0x242070db);
    FF(b, c, d, a, M[3], 22, 0xc1bdceee);
    FF(a, b, c, d, M[4], 7, 0xf57c0faf);
    FF(d, a, b, c, M[5], 12, 0x4787c62a);
    FF(c, d, a, b, M[6], 17, 0xa8304613);
    FF(b, c, d, a, M[7], 22, 0xfd469501);
    FF(a, b, c, d, M[8], 7, 0x698098d8);
    FF(d, a, b, c, M[9], 12, 0x8b44f7af);
    FF(c, d, a, b, M[10], 17, 0xffff5bb1);
    FF(b, c, d, a, M[11], 22, 0x895cd7be);
    FF(a, b, c, d, M[12], 7, 0x6b901122);
    FF(d, a, b, c, M[13], 12, 0xfd987193);
    FF(c, d, a, b, M[14], 17, 0xa679438e);
    FF(b, c, d, a, M[15], 22, 0x49b40821);
```

```
    // 第二轮循环
    GG(a, b, c, d, M[ 1], 5, 0xf61e2562);
    GG(d, a, b, c, M[ 6], 9, 0xc040b340);
    GG(c, d, a, b, M[11], 14, 0x265e5a51);
    GG(b, c, d, a, M[ 0], 20, 0xe9b6c7aa);
    GG(a, b, c, d, M[ 5], 5, 0xd62f105d);
    GG(d, a, b, c, M[10], 9,  0x2441453);
    GG(c, d, a, b, M[15], 14, 0xd8a1e681);
    GG(b, c, d, a, M[ 4], 20, 0xe7d3fbc8);
    GG(a, b, c, d, M[ 9], 5, 0x21e1cde6);
    GG(d, a, b, c, M[14], 9, 0xc33707d6);
    GG(c, d, a, b, M[ 3], 14, 0xf4d50d87);
    GG(b, c, d, a, M[ 8], 20, 0x455a14ed);
    GG(a, b, c, d, M[13], 5, 0xa9e3e905);
    GG(d, a, b, c, M[ 2], 9, 0xfcefa3f8);
    GG(c, d, a, b, M[ 7], 14, 0x676f02d9);
    GG(b, c, d, a, M[12], 20, 0x8d2a4c8a);
    // 第三轮循环
    HH(a, b, c, d, M[ 5], 4, 0xfffa3942);
    HH(d, a, b, c, M[ 8], 11, 0x8771f681);
    HH(c, d, a, b, M[11], 16, 0x6d9d6122);
    HH(b, c, d, a, M[14], 23, 0xfde5380c);
    HH(a, b, c, d, M[ 1], 4, 0xa4beea44);
    HH(d, a, b, c, M[ 4], 11, 0x4bdecfa9);
    HH(c, d, a, b, M[ 7], 16, 0xf6bb4b60);
    HH(b, c, d, a, M[10], 23, 0xbebfbc70);
    HH(a, b, c, d, M[13], 4, 0x289b7ec6);
    HH(d, a, b, c, M[ 0], 11, 0xeaa127fa);
    HH(c, d, a, b, M[ 3], 16, 0xd4ef3085);
    HH(b, c, d, a, M[ 6], 23,  0x4881d05);
    HH(a, b, c, d, M[ 9], 4, 0xd9d4d039);
    HH(d, a, b, c, M[12], 11, 0xe6db99e5);
    HH(c, d, a, b, M[15], 16, 0x1fa27cf8);
    HH(b, c, d, a, M[ 2], 23, 0xc4ac5665);
```

```
    // 第四轮循环
    II(a, b, c, d, M[ 0], 6, 0xf4292244);
    II(d, a, b, c, M[ 7], 10, 0x432aff97);
    II(c, d, a, b, M[14], 15, 0xab9423a7);
    II(b, c, d, a, M[ 5], 21, 0xfc93a039);
    II(a, b, c, d, M[12], 6, 0x655b59c3);
    II(d, a, b, c, M[ 3], 10, 0x8f0ccc92);
    II(c, d, a, b, M[10], 15, 0xffeff47d);
    II(b, c, d, a, M[ 1], 21, 0x85845dd1);
    II(a, b, c, d, M[ 8], 6, 0x6fa87e4f);
    II(d, a, b, c, M[15], 10, 0xfe2ce6e0);
    II(c, d, a, b, M[ 6], 15, 0xa3014314);
    II(b, c, d, a, M[13], 21, 0x4e0811a1);
    II(a, b, c, d, M[ 4], 6, 0xf7537e82);
    II(d, a, b, c, M[11], 10, 0xbd3af235);
    II(c, d, a, b, M[ 2], 15, 0x2ad7d2bb);
    II(b, c, d, a, M[ 9], 21, 0xeb86d391);

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

```
/**
 * Args：input表示输入字节char数组，output表示输出unsigned int数组，length表示字节长度
 * Func：unsigned char转成unsigned int（左低右高）
 * Return：空
 */
void MD5::UnsignedCharToUnsignedInt(const unsigned char* input, unsigned int* output, size_t length) {
    for (size_t i = 0, j = 0; j < length; i++, j += 4) {
        output[i] = ((static_cast<unsigned int>(input[j]))
                    | ((static_cast<unsigned int>(input[j + 1])) << 8)
                    | ((static_cast<unsigned int>(input[j + 2])) << 16)
                    | ((static_cast<unsigned int>(input[j + 3])) << 24));
    }
}
```

```
/**
 * Args：opNumber表示待左移的数，opBit表示左移的位数
 * Func：完成循环左移操作
 * Return：循环左移后的结果
 */
unsigned int MD5::LeftRotate(unsigned int opNumber, unsigned int opBit) {
    unsigned int left = opNumber;
    unsigned int right = opNumber;
    return (left << opBit) | (right >> (32 - opBit));
}
```

```cpp
void MD5::FF(unsigned int &a, unsigned int b, unsigned int c, unsigned int d, unsigned int Mi, unsigned int s, unsigned int Ti) {
    unsigned int temp = a + F(b, c, d) + Mi + Ti;
    a = b + LeftRotate(temp, s);
}

void MD5::GG(unsigned int &a, unsigned int b, unsigned int c, unsigned int d, unsigned int Mi, unsigned int s, unsigned int Ti) {
    unsigned int temp = a + G(b, c, d) + Mi + Ti;
    a = b + LeftRotate(temp, s);
}

void MD5::HH(unsigned int &a, unsigned int b, unsigned int c, unsigned int d, unsigned int Mi, unsigned int s, unsigned int Ti) {
    unsigned int temp = a + H(b, c, d) + Mi + Ti;
    a = b + LeftRotate(temp, s);
}

void MD5::II(unsigned int &a, unsigned int b, unsigned int c, unsigned int d, unsigned int Mi, unsigned int s, unsigned int Ti) {
    unsigned int temp = a + I(b, c, d) + Mi + Ti;
    a = b + LeftRotate(temp, s);
}
```

```cpp
unsigned int MD5::F(unsigned int x, unsigned int y, unsigned int z) {
    return (x & y) | ((~x) & z);
}

unsigned int MD5::G(unsigned int x, unsigned int y, unsigned int z) {
    return (x & z) | (y & (~z));
}

unsigned int MD5::H(unsigned int x, unsigned int y, unsigned int z) {
    return x ^ y ^ z;
}

unsigned int MD5::I(unsigned int x, unsigned int y, unsigned int z) {
    return y ^ (x | (~z));
}
```

第四步：拼接结果

把循环加工最终产生的 A、B、C、D 四个值拼接在一起，转换成字符串即可。

```
/**
 * @Generate md5 digest.
 *
 */
void MD5::final() {
    unsigned char bits[8];
    unsigned int oldState[4], oldCount[2];
    unsigned int index, padLen;

    /* Save current state and count. */
    memcpy(oldState, state, 16);
    memcpy(oldCount, count, 8);

    /* Save number of bits */
    UnsignedIntToUnsignedChar(count, bits, 8);

    /* Pad out to 56 mod 64. */
    index = static_cast<unsigned int>((count[0] >> 3) & 0x3f);
    padLen = (index < 56) ? (56 - index) : (120 - index);
    encryptUnsignedChar(padding, padLen);

    /* Append length (before padding) */
    encryptUnsignedChar(bits, 8);

    /* Store state in digest */
    UnsignedIntToUnsignedChar(state, result, 16);

    /* Restore current state and count. */
    memcpy(state, oldState, 16);
    memcpy(count, oldCount, 8);
}
```

```
/**
 * Args：input表示unsigned int数组, output表示输出字节char数组, length表示输入字节长度
 * Func：unsigned int转成unsigned char
 * Return：空
 */
void MD5::UnsignedIntToUnsignedChar(const unsigned int* input, unsigned char* output, size_t length) {
    for (size_t i = 0, j = 0; j < length; i++, j += 4) {
        output[j] = static_cast<unsigned char>(input[i] & 0xff);
        output[j + 1] = static_cast<unsigned char>((input[i] >> 8) & 0xff);
        output[j + 2] = static_cast<unsigned char>((input[i] >> 16) & 0xff);
        output[j + 3] = static_cast<unsigned char>((input[i] >> 24) & 0xff);
    }
}
```

```cpp
/**
 * Args：input表示生成的信息摘要digest
 * Func：unsigned char转成16进制输出
 * Return：16进制字符串以输出
 */
string MD5::UnsignedCharToHexString(const unsigned char* input) {
    const char charToHex[16] = {'0', '1', '2', '3', '4', '5', '6', '7',
                                '8', '9', 'a', 'b', 'c', 'd', 'e', 'f'};
    string str;
    for (size_t i = 0; i < 16; i++) {
        unsigned int temp = static_cast<unsigned int>(input[i]);
        unsigned int a = temp / 16;
        unsigned int b = temp % 16;
        str.append(1, charToHex[a]);
        str.append(1, charToHex[b]);
    }
    return str;
}
```

运行结果截图：

```
PS C:\Users\linji\Desktop\HW\大三上\信息安全技术\ss2015_15331046_陈志扬_is_assign_2> g++ .\main.cpp -o main
PS C:\Users\linji\Desktop\HW\大三上\信息安全技术\ss2015_15331046_陈志扬_is_assign_2> ./main
If you want to QUIT, please input the character '#'!
Please input the message that you want to encrypt:
123456
The MD5 digest is:
e10adc3949ba59abbe56e057f20f883e
Congratulations! You have done it successfully.

Please input the message that you want to encrypt:
asdfghjkl
The MD5 digest is:
c44a471bd78cc6c2fea32b9fe028d30a
Congratulations! You have done it successfully.

Please input the message that you want to encrypt:
#
You have quit the program!
PS C:\Users\linji\Desktop\HW\大三上\信息安全技术\ss2015_15331046_陈志扬_is_assign_2>
```

和 md5 在线加密解密网站比较可以得到输出结果正确，程序逻辑正确：

我们还可以利用 Python2.X 的 hashlib 包中的 md5()函数直接生成信息摘要：

```python
import hashlib
def md5(str):
    import hashlib
    import types
    if type(str) is types.StringType:
        m = hashlib.md5()
        m.update(str)
        return m.hexdigest()
    else:
        return ''

print md5('123456')
```

运行结果和 C++程序运行所得结果相同：

```
PS C:\Users\linji> & python27 c:/Users/linji/Desktop/HW/大
e10adc3949ba59abbe56e057f20f883e
```