开发环境是eclipse+jkd1.8

认证跳表的实现过程：

准备算法：

Hash函数——MD5码

定义四个初始变量：

**static** **final** **long** ***A***=0x67452301L;

**static** **final** **long** ***B***=0xefcdab89L;

**static** **final** **long** ***C***=0x98badcfeL;

**static** **final** **long** ***D***=0x10325476L;

一个初始函数:

g(i,x,y,z);其中g(0,x,y,z)=F(x,y,z)=(x&y)|((~x)&z);

g(1,x,y,z)=G(x,y,z)=(x&z)|((~z)&y);g(2,x,y,z)=H(x,y,z)=x^y^z;

g(3,x,y,z)=I(x,y,z)=y^(x|(~z));

移位数组s = {{7,12,17,22},{5,9,14,20},{4,11,16,23},{6,10,15,21}};

结果集result={A,B,C,D}

四种常用操作（)）

1. 其中a=b+((a+F(b,c,d)++)<<<s)
2. 其中a=b+((a+G(b,c,d)++)<<<s)
3. 其中a=b+((a+H(b,c,d)++)<<<s)
4. 其中a=b+((a+I(b,c,d)++)<<<s)

具体算法

数据填充fileFunction(String message)：

1：将message转化为字节数组byte，然后利用字节数组byte的长度算出需要填充的字节数x。

2：添加1和若干0直到长度等于x，计算此时的在计算message的二进制的低64位，此时再将此填充到byte数组中。

3：将处理好的byte数组按512长度分为若干组，并把每个组按64字节转化为long型，记每一组为。

数据处理dataHanding(long[] data)：

输入长度为16的,对其进行4轮操作。第一轮进行16次

第二轮进行16次…。

其中是代表第数据块的16进制数。

最后将16进制数转化为字符串进行拼接。

其中关键算法dataHanding(long[] data)的代码如下：

a = result[0], b = result[1], c = result[2], d = result[3]

for i=0:3 step 1

for j=0:15 step 1

result[0] += g(i,result[1], result[2], result[3])

+data[K[i][j]]+)

result[0] = result[1]

+ (result[0]<< ***S***[n][i] | (result[0] >>> (32 - ***S***[n][i])));

end

change(result[0],result[1], result[2] result[3])

end

result[0] += a;

result[1] += b;

result[2] += c;

result[3] += d;

测试：

分别对”hello”和”认证”两个字符串进行加密：

java程序：

hello：5D41402ABC4B2A76B9719D911017C592

认证: B7158A42CA1C2845B3C08111982B0494

在线MD5码加密（[https://md5jiami.51240.com](https://md5jiami.51240.com/)）结果：

hello：5D41402ABC4B2A76B9719D911017C592

认证: B7158A42CA1C2845B3C08111982B494

认证跳表实现

建立跳表结构：

跳表节点的结构如下

{

Long key;//节点的key

Object value;//节点原始的Hash值

SkipListNode[] forword;//用于存储前节点

SkipListNode[] back;//用于存储后节点

}

实现跳表的常用操作：算法见第四章

init();//初始化跳表

add\_Node();//增加节点

find();

delete\_Node();

change\_Node();