

密码学

平时实验报告

（ 2020 / 2021学年 第 二 学期）

题 目： SHA1哈希实验

|  |  |
| --- | --- |
| **专 业** | 信息安全 |
| **组 长 姓 名** |  |
| **组 员 姓 名** |  |
| **班 级 学 号** |  |
| **指 导 教 师** | 李琦 |
| **指 导 单 位** | 信息安全系 |
| **日 期** | 2021年5月7日 |

|  |  |
| --- | --- |
| **成员分工** | |
| 组长（1） | 任远哲，B19031614，写代码和实验报告 |
| **简短评语：**  **指导教师：年月日** | |

1. **课题内容和要求**

**1.实验环境**

实验主机操作系统为Windows 7

**2.实验内容**

将长度小于56byte的字符串，转化成bit串（长度不超过448bit）。对bit串用SHA1算法求其哈希值。

**二、课题需求分析**

1.将需要加密的字符串（长度小于56byte），转化成比特串（长度不超过448bit）。对比特串结果填写一个1和若干个0将其补齐到448bit。然后在比特串最后附加64比特的无符号整数，其值为原始消息的长度。这样就产生了长度为512比特的比特串。

2.初始化5个链接变量：

result[0]=0x67452301;

result[1]=0xEFCDAB89;

result[2]=0x98BADCFE;

result[3]=0x10325476;

result[4]=0xC3D2E1F0;

3.压缩长度为512比特的比特串：采用4轮，每轮20个步骤。不同轮的步函数采用不同的非线性函数。由长度为512比特的比特串生成W[80]（每一个W长度为32bit），定义K[4]，每一个步骤由前面的result状态，计算下一个result状态。第80个步骤结束后那长度为160比特串即为哈希结果。

4.对得到的长度为160比特串，每四个为一组，按16进制格式输出，即为SHA1值。

**完整代码和注释：**

#include <stdio.h>

#include <string.h>

#define ROTL(x,n) (x<<n|x>>(32-n)) //宏定义ROTL函数为循环左移

//循环左移n: (a>>(N-n))|(a>>n),循环右移n: (a<<(N-n))|(a>>n)

static unsigned long M[16],W[80],result[5],bitstring[1000];//unsigned long: 4个字节,即32bit

//M[16],512位，是sha1算法的的处理单位元

//预处理，将字符串转化为比特串(程序的局限性：比特串长度小于448)，并补位(补到448位)+补长度(64位)=512位

static void sha1\_init(char \*input, int len){

int i,n;

for(i=0;i<16;i++){

M[i] = 0;

}

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

n = 24-((i&0x03)<<3);

M[i/4] |= input[i]<<n;

}

n = 24-((i&0x03)<<3); //11000-i&011<<3

M[i/4] |= 0x80<<n; //10000000<<n

M[15] = len\*8;

}

//由 512位 M生成 80个 W，每一个W 32bit

static void generate\_W(void){

int i;

for(i=0;i<16;i++){

W[i]=M[i];

}

for(i=16;i<80;i++){

W[i]=W[i-16]^W[i-14]^W[i-8]^W[i-3];

W[i]=ROTL(W[i],1);

}

}

//由 80个 W计算 result，即对512bit的串 sha1的结果 ，结果为160bit，存在reslt[5]中

static void sha1\_compress(){

int i;

unsigned long a,b,c,d,e,f,k,temp;

result[0]=0x67452301;

result[1]=0xEFCDAB89;

result[2]=0x98BADCFE;

result[3]=0x10325476;

result[4]=0xC3D2E1F0;

a = result[0];

b = result[1];

c = result[2];

d = result[3];

e = result[4];

for(i=0;i<80;i++){

switch(i/20){

case 0:

k=0x5A827999;

f=(b&c)|(~b&d);

break;

case 1:

k=0x6ED9EBA1;

f=b^c^d;

break;

case 2:

k=0x8F1BBCDC;

f=(b&c)|(b&d)|(c&d);

break;

case 3:

k=0xCA62C1D6;

f=b^c^d;

break;

}

temp=ROTL(a,5)+f+e+W[i]+k;

e=d;

d=c;

c=ROTL(b,30);

b=a;

a=temp;

}

result[0]+=a;//模2^32加法 ，unsigned int 类型正好为32bit

result[1]+=b;

result[2]+=c;

result[3]+=d;

result[4]+=e;

}

//input:待校验的数据，len:数据长度(小于56字节,即448bit)

unsigned long\* sha1(char\* input, int len){

sha1\_init(input,len);

generate\_W();

sha1\_compress();

return result;

}

int main(){

char str[10000];

unsigned long\* SHA;

printf("输入:");

gets(str);

SHA=sha1(str,strlen(str));

printf("SHA1=%08X%08X%08X%08X%08X\n",SHA[0],SHA[1],SHA[2],SHA[3],SHA[4]);

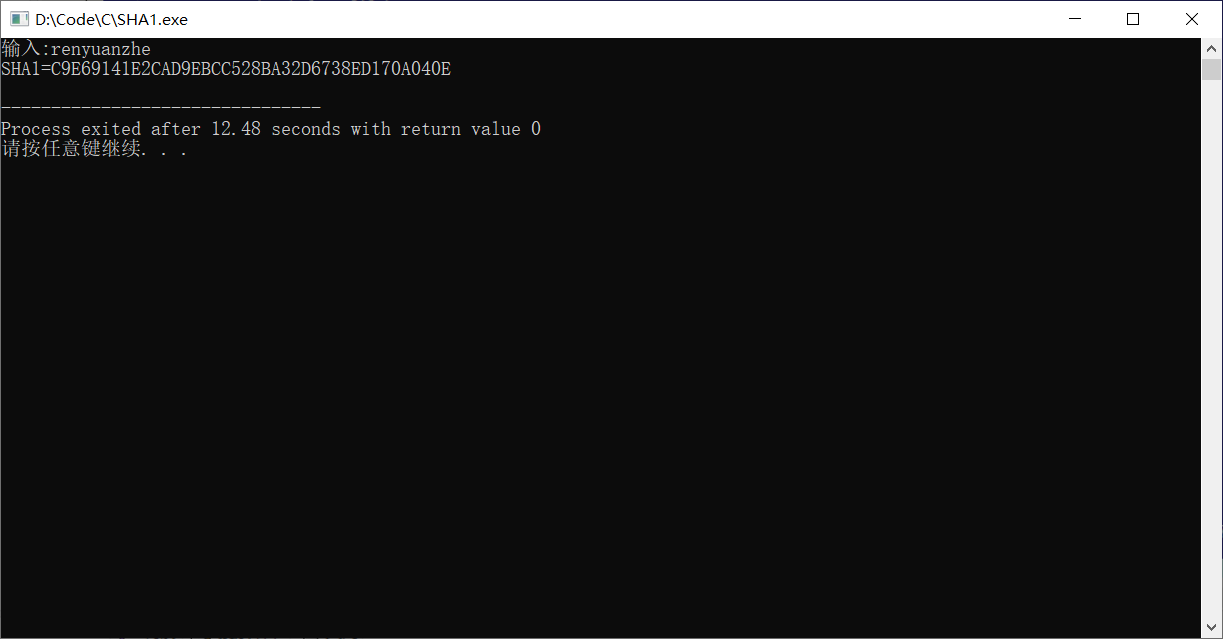
//%08x为整型以16进制方式输出的格式字符串，会把后续对应参数的整型数字，以16进制输出。08的含义为，输出的16进制值占8位，不足部分左侧补0。这里unsigned int 为32位，一个16进制数为 4位，正好8个

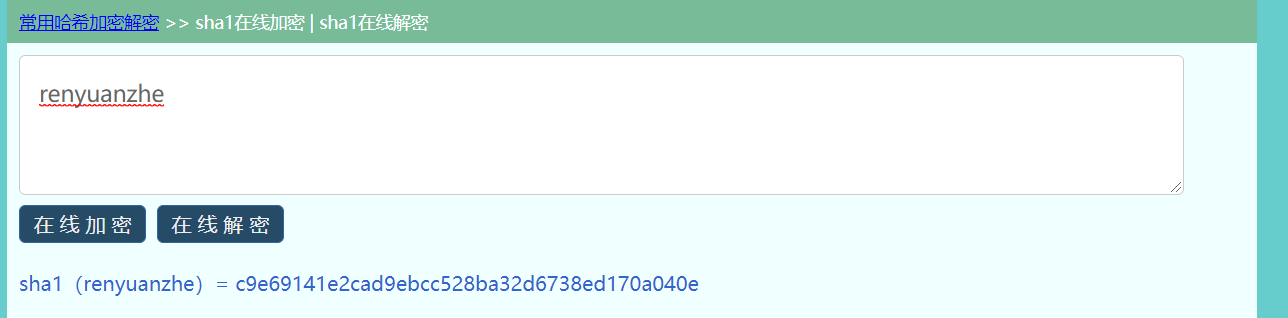
return 0;

}

**三. 运行结果**

我的程序和网络上在线SHA1算法得出相同的结果





**四、课题完成过程中遇到的问题及解决方法**

问题1：循环左移我一开始理解成单纯的w[i]<<n

解决方法：后来在互联网上查询，发现应该是w[i]<<n|w[i]>>(32-n) ，才能达到循环的效果。

问题2：程序中变量的单位必须为unsigned int(=unsigned long)类型。

解决方法：一开始我用int存放长度32比特串，后来发现结果不对，仔细检查发现要使用

无符号类型。无符号类型和有符号类型在计算机中的二进制形式不一样，因此会出错。

**五、总结**

经过本次实验，我深入理解了SHA1算法的各个流程。光看书觉得很简单，但实际写一遍程序会发现存在很多问题和疑惑，所幸现在的互联网发达，很多问题在互联网上都能找到答案。SHA1是一种不可逆的算法，所以不能用于加密，但两个不同的消息经过SHA1基本不会产生同样的消息摘要，所以可以用摘要来校验信息是否正确。SHA-1已经不再视为可抵御有充足资金、充足计算资源的攻击者。2005年，密码分析人员发现了对SHA1的有效攻击方法，这表明该算法可能不够安全，不能继续使用，自2010年以来，许多组织建议用SHA2或SHA3来替换SHA1。