**电子科技大学信息与软件工程学院**

**标准实验报告**

**（实验）课程名称信息安全导论**

**电子科技大学教务处制表**

电子科技大学

**实 验 报 告**

**学生姓名： 贾怀宇 学 号： 2020090916007 指导教师：赵洋**

**实验地点：信软楼400 实验时间：2022年10月24日**

1. **实验室名称：**

**信软楼400**

**二、实验项目名称：简单加解密算法设计与实现**

**三、实验学时： 4 学时**

**（一）实验目的**

密码学是信息安全研究领域的基础学科，理解加解密算法的基本原理是系统学习密码学理论的前提条件。通过设计和实现简单的加解密算法，可以加深对导论课程中相关基本概念的理解，为后续的学习奠定基础。

**（二）实验内容**

（1）简单替代加解密算法实现；

（2）简单移位加解密算法实现。

**四、实验原理**

替代和移位是古典密码体制中常见的两种加解密模式。替换加密保留了明文中字符的排列顺序，通过将原始字符替代为其他字符的方式来隐藏原始的信息。古典密码体制中的替代加密可以分为单表替代和多表替代，单表替代中原始字符只有一种替代方式，如凯撒密码中固定用当前字母后的第三个字母替换原字母，多表替换则存在不同的替换方式，如维吉尼亚密码。不同于替代加密，移位加密只是调整了明文中字符的位置，字符本身并没有改变，如借助工具实现的斯巴达手杖，以及表格换位法等。古典密码体制因受到计算能力的限制，目前看来都不具备基本的安全性，但替代和移位始终是实现加解密变换的基本思路。

数学理论的发展和计算机的出现使加解密变换的复杂度大幅度提高，借助计算机强大的计算能力能够轻松地融合替代和移位技术，组合成新的加密算法。例如，DES（Data Encryption Standard，标准加密标准）将原文分组，每组8个字母，分别进行16轮移位和替换操作。

1. **简单替代加解密算法实现**

凯撒密码是最早出现的一种替代加解密算法，其基本思路是先将字母进行编码，a=0，b=1，……，z=25，选择一个正整数*k*()，加密过程为计算，其中m为被加密的字母，c为加密后得到的字母；解密过程为计算。

1. **简单移位加解密算法实现**

斯巴达手杖是一种借助工具实现的移位加密方法，其基本原理是将书写信息的纸条（或羊皮）缠绕在特定的木棍上，然后自上到下写上明文信息，最后将字条从木棍上取下，构成明文的字母顺序就被打乱，当纸条重新被缠绕回相同的木棍时字母顺序被还原，从而可以读取明文信息。假定木棍的截面为一个正六边型，字条缠绕方向和文字书写方向为自上而下，自右向左，简单的示意如图1所示。

图1(a)纸条缠绕后书写的信息 图1(b)展开后的纸条

模拟斯巴达手杖的加密过程，假定木棍为3-6边的正多变型，可以看做将明文按列写入矩阵，然后按行输出到数组，通过一个逆过程可以重新得到明文。

**五、实验器材（设备、元器件）**

（一）学生每人一台PC，安装WindowsXP/2000以上操作系统。

（二）个人PC安装C、C++或JAVA程序开发环境。

**六、实验步骤**

**（一）简单替代加解密算法实现**

以凯撒密码为替代加解密算法原型完成：

1. 实现加解密参数的随机选择函数REP\_KeyGen ( );
2. 实现加密函数int REP\_Encrypt(short key, char\*plaintext, int plength, char\* ciphertext)
3. 实现解密函数int REP\_Decrypt(short key, char\*ciphertext, int clength, char\* plaintext)
4. 利用上述函数编写一个简单的替代加解密例程。

**（二）简单移位加解密算法实现**

模拟斯巴达手杖为替代加解密算法原型完成：

1. 实现加解密参数的随机选择函数SHIFT\_KeyGen ( );
2. 实现加密函数int SHIFT\_Encrypt(short key, char\* plaintext, int plength, char\* ciphertext,)
3. 实现解密函数int SHIFT\_Decrypt(short key, char\* ciphertext, int clength, char\* plaintext)
4. 利用上述函数编写一个简单的移位加解密例程。

**七、实验数据及结果分析**

**（一）简单替代加解密算法实现（C语言实现）**

**头文件解析：**

**#include<stdio.h>**

**#include<string.h> //字符串处理**

**#include<stdlib.h>**

**#include<time.h> //调用时间生成随机数**

1. **实现加解密参数的随机选择函数REP\_KeyGen ( );**

//生成key的函数

int keyGen(){

srand((int)time(0)); //调用time头文件生成随机数

return rand()%25+1; //规定随机数的生成范围 ，凯撒加密算法位移密钥key的区间在(1,25)

}

1. **实现加密函数int REP\_Encrypt(short key, char\*plaintext, int plength, char\* ciphertext)**

//加密函数

void Encryption()

{

printf("please input plaintext：");

gets(plaintext);

printf("ciphertext：");

for(int i=0;plaintext[i]!='**\0**';i++){

if(plaintext[i]>='A'&&plaintext[i]<='Z'){ //如果当前字符是大写，就和‘A’比较

ciphertext[i]=(plaintext[i]-'A'+key)%26+'A';

}

else if(plaintext[i]>='a'&&plaintext[i]<='z'){ //如果当前字符是小写，就和‘a’比较

ciphertext[i]=(plaintext[i]-'a'+key)%26+'a';

}

else

ciphertext[i]=plaintext[i]+key; //非大小写，直接位移

printf("%c",ciphertext[i]); //输出当前字符

}

printf("**\n**");

}

1. **实现解密函数int REP\_Decrypt(short key, char\*ciphertext, int clength, char\* plaintext)**

//解密函数

void Decryption()

{

printf("please input ciphertext：");

gets(ciphertext);

int len=strlen(ciphertext);

printf("plaintext：");

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

{

if(ciphertext[i]>='A'&&ciphertext[i]<='Z'){ //如果当前字符是大写，就和‘A’比较

plaintext[i] = ((ciphertext[i]-'A'-key)%26+26)%26 +'A';

}

else if(ciphertext[i]>='a'&&ciphertext[i]<='z'){ //如果当前字符是小写，就和‘a’比较

plaintext[i]=((ciphertext[i]-'a'-key)%26+26)%26+'a';

}

else

plaintext[i]=ciphertext[i]+key; //非大小写，直接位移

printf("%c",plaintext[i]); //输出当前字符

}

printf("**\n**");

}

1. **利用上述函数编写一个简单的替代加解密例程。**

**凯撒加解密程序：**

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

#include<time.h>

#define MAX 101

char ciphertext[MAX]; //密文数组

char plaintext[MAX]; //明文数组

int key; //密钥

//生成key的函数

int keyGen(){

srand((int)time(0)); //调用time头文件生成随机数

return rand()%25+1; //规定随机数的生成范围 ，凯撒加密算法位移密钥key的区间在(1,25)

}

//加密函数

void Encryption()

{

printf("please input plaintext：");

gets(plaintext);

printf("ciphertext：");

for(int i=0;plaintext[i]!='**\0**';i++){

if(plaintext[i]>='A'&&plaintext[i]<='Z'){ //如果当前字符是大写，就和‘A’比较

ciphertext[i]=(plaintext[i]-'A'+key)%26+'A';

}

else if(plaintext[i]>='a'&&plaintext[i]<='z'){ //如果当前字符是小写，就和‘a’比较

ciphertext[i]=(plaintext[i]-'a'+key)%26+'a';

}

else

ciphertext[i]=plaintext[i]+key; //非大小写，直接位移

printf("%c",ciphertext[i]); //输出当前字符

}

printf("**\n**");

}

//解密函数

void Decryption()

{

printf("please input ciphertext：");

gets(ciphertext);

int len=strlen(ciphertext);

printf("plaintext：");

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

{

if(ciphertext[i]>='A'&&ciphertext[i]<='Z'){ //如果当前字符是大写，就和‘A’比较

plaintext[i] = ((ciphertext[i]-'A'-key)%26+26)%26 +'A';

}

else if(ciphertext[i]>='a'&&ciphertext[i]<='z'){ //如果当前字符是小写，就和‘a’比较

plaintext[i]=((ciphertext[i]-'a'-key)%26+26)%26+'a';

}

else

plaintext[i]=ciphertext[i]+key; //非大小写，直接位移

printf("%c",plaintext[i]); //输出当前字符

}

printf("**\n**");

}

int main()

{

int n,flag=1;

key=keyGen(); //调用keyGen()函数生成随即加密密钥key

printf("key: %d**\n**",key);

while(flag){ //死循环，只有当用户手动输入才结束

printf("please choose（1:Encryption，2:Decryption,3:exit）：");

scanf("%d",&n);

getchar();

switch(n){

case 1:

Encryption(); //加密

break;

case 2:

Decryption(); //解密

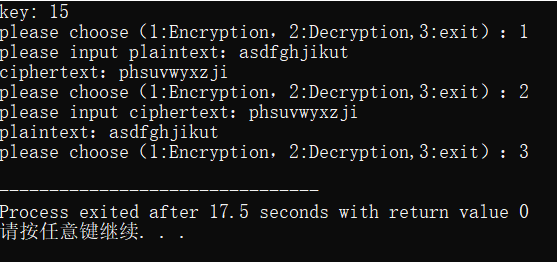
break;

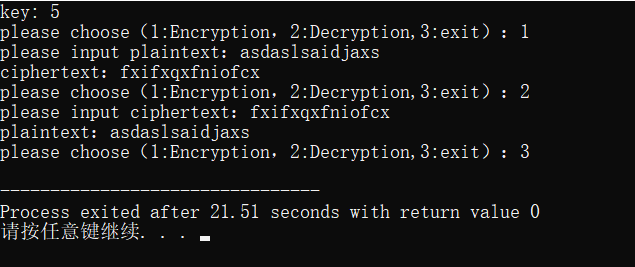
case 3:exit(0); //退出

}

}

}

**实验运行结果：**



**可以看到同一个字符串经过加解密后复原**

**（二）简单移位加解密算法实现**

**（1） 实现加解密参数的随机选择函数SHIFT\_KeyGen ( );**

//斯巴达手杖生成key的函数

int keyGen(){

srand((int)time(0));

return rand()%4+3;

}

**（2） 实现加密函数int SHIFT\_Encrypt(short key, char\* plaintext, int plength, char\* ciphertext,)**

//斯巴达手杖加密

int Encrypt(short key, char\* plaintext, int plength, char\* ciphertext, int\* clength){

int m=plength/key;

int n=plength%key;

int i,j=0,z=0;

while(j<n){

for(i=0;i<m+1;i++,z++){

ciphertext[z]=plaintext[j+i\*key]; //对刚好可以框入正方形的数据进行加密

}

j++;

}

while(j<key){

for(i=0;i<m;i++,z++){

ciphertext[z]=plaintext[j+i\*key]; //对多余部分进行加密

}

j++;

}

\*clength=strlen(ciphertext);

}

**（3） 实现解密函数int SHIFT\_Decrypt(short key, char\* ciphertext, int clength, char\* plaintext)**

//斯巴达手杖解密

int Decrypt(short key, char\* ciphertext, int clength, char\* plaintext, int\* plength){

int m=clength/key;

int n=clength%key;

int i,j=0,z=0;

while(j<n){

for(i=0;i<m+1;i++,z++){

plaintext[j+i\*key]=ciphertext[z]; //对刚好可以框入正方形的数据进行解密

}

j++;

}

while(j<key){

for(i=0;i<m;i++,z++){

plaintext[j+i\*key]=ciphertext[z]; //对多余部分进行解密

}

j++;

}

\*plength=strlen(plaintext);

}

1. **利用上述函数编写一个简单的移位加解密例程。**

**斯巴达手杖加解密程序**

#include<stdio.h>

#include<string.h>

#include<time.h>

#include<stdlib.h>

//斯巴达手杖密码声明

int keyGen();

int Encrypt(short key, char\* plaintext, int plength, char\* ciphertext, int\* clength);

int Decrypt(short key, char\* ciphertext, int clength, char\* plaintext, int\* plength);

int main(){

char plaintext[100],ciphertext[100],plaintext1[100];

printf("请输入您想加密的字符串:");

gets(plaintext);

int plength=strlen(plaintext);

int clength;

int plength1;

int key=keyGen();

printf("key:%d**\n**",key);

Encrypt(key,plaintext,plength,ciphertext,&clength);

printf("字符串长度是：%d**\n**",plength);

printf("加密后的字符串是：");

puts(ciphertext);

Decrypt(key,ciphertext,clength,plaintext1,&plength1); */\*解密过程,并回复用户以便检验是否解密成功 \*/*

printf("字符串长度是：%d**\n**",clength);

printf("您输入的字符串经加密再解密后：");

puts(plaintext1);

}

//斯巴达手杖生成key的函数

int keyGen(){

srand((int)time(0));

return rand()%4+3;

}

//斯巴达手杖加密

int Encrypt(short key, char\* plaintext, int plength, char\* ciphertext, int\* clength){

int m=plength/key;

int n=plength%key;

int i,j=0,z=0;

while(j<n){

for(i=0;i<m+1;i++,z++){

ciphertext[z]=plaintext[j+i\*key]; //对刚好可以框入正方形的数据进行加密

}

j++;

}

while(j<key){

for(i=0;i<m;i++,z++){

ciphertext[z]=plaintext[j+i\*key]; //对多余部分进行加密

}

j++;

}

\*clength=strlen(ciphertext);

}

 //斯巴达手杖解密

int Decrypt(short key, char\* ciphertext, int clength, char\* plaintext, int\* plength){

int m=clength/key;

int n=clength%key;

int i,j=0,z=0;

while(j<n){

for(i=0;i<m+1;i++,z++){

plaintext[j+i\*key]=ciphertext[z]; //对刚好可以框入正方形的数据进行解密

}

j++;

}

while(j<key){

for(i=0;i<m;i++,z++){

plaintext[j+i\*key]=ciphertext[z]; //对多余部分进行解密

}

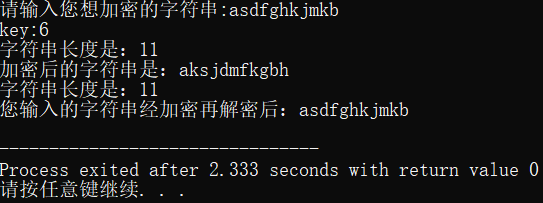
j++;

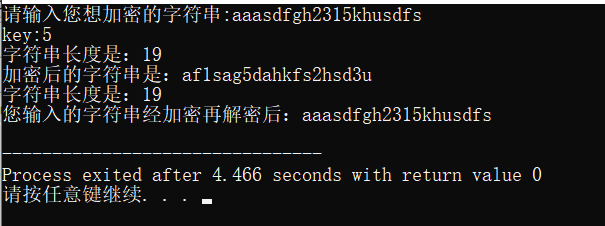
}

\*plength=strlen(plaintext);

}

 实验结果：





 可以看到输入的字符串经斯巴达手杖加密又解密后复原

**八、实验结论、心得体会**

本次实验我们进行了凯撒加解密和斯巴达手杖加解密，并且通过在C语言程序上实现。凯撒加解密的原理是字符的替代，我们选择一个密钥key作为每一个字符所要后移的位置。斯巴达手杖的加解密原理是移位，假定木棍的截面为一个正六边型，字条缠绕方向和文字书写方向为自上而下，自右向左。这两种加密方式是最基本简单的，也是最容易破解的，二者都可以通过尝试密钥进行暴力破解，不过是我们学习密码学的开始，让我们形成了“密钥，加密，解密”的基本思想。

**九、对本实验过程及方法、手段的改进建议**

1.本实验只在C语言程序上实现，还可以选择java或者python等，提高算法的实现速度。

2.key的生成选择了通过调用time头文件随机生成，可以自己编写随机生成函数randomKey()

3.凯撒加密时我们只选择了最基本的位移，也就是让当前字符通过ASCII码向后移动，但针对某些不可转化的字符这种方式存在缺陷。

**报告评分：**

**指导教师签字：**